Features

- High Performance, Low Power 32-bit AVR® Microcontroller
  - Compact Single-cycle RISC Instruction Set Including DSP Instruction Set
  - Built-in Floating-Point Processing Unit (FPU)
  - Read-Modify-Write Instructions and Atomic Bit Manipulation
  - Performing 1.49 DMIPS / MHz
    - Up to 91 DMIPS Running at 66 MHz from Flash (1 Wait-State)
    - Up to 49 DMIPS Running at 33 MHz from Flash (0 Wait-State)
  - Memory Protection Unit
- Multi-hierarchy Bus System
  - High-Performance Data Transfers on Separate Buses for Increased Performance
  - 16 Peripheral DMA Channels Improves Speed for Peripheral Communication
- Internal High-Speed Flash
  - 512 Kbytes, 256 Kbytes, 128 Kbytes, 64 Kbytes Versions
  - Single Cycle Access up to 33 MHz
  - FlashVault™ Technology Allows Pre-programmed Secure Library Support for End User Applications
  - Prefetch Buffer Optimizing Instruction Execution at Maximum Speed
  - 100,000 Write Cycles, 15-year Data Retention Capability
  - Flash Security Locks and User Defined Configuration Area
- Internal High-Speed SRAM, Single-Cycle Access at Full Speed
  - 64 Kbytes (512 KB and 256 KB Flash), 32 Kbytes (128 KB Flash), 16 Kbytes (64 KB Flash)
  - 4 Kbytes on the Multi-Layer Bus System (HSB RAM)
- External Memory Interface on AT32UC3C0 Derivatives
  - SDRAM / SRAM Compatible Memory Bus (16-bit Data and 24-bit Address Buses)
- Interrupt Controller
  - Autovectored Low Latency Interrupt Service with Programmable Priority
- System Functions
  - Power and Clock Manager
  - Internal 115KHz (RCSYS) and 8MHz/1MHz (RC8M) RC Oscillators
  - One 32 KHz and Two Multipurpose Oscillators
  - Clock Failure detection
  - Two Phase-Lock-Loop (PLL) allowing Independent CPU Frequency from USB or CAN Frequency
- Windowed Watchdog Timer (WDT)
- Asynchronous Timer (AST) with Real-Time Clock Capability
  - Counter or Calendar Mode Supported
- Frequency Meter (FREQM) for Accurate Measuring of Clock Frequency
- Ethernet MAC 10/100 Mbps interface
  - 802.3 Ethernet Media Access Controller
  - Supports Media Independent Interface (MII) and Reduced MII (RMII)
- Universal Serial Bus (USB)
  - Device 2.0 and Embedded Host Low Speed and Full Speed
  - Flexible End-Point Configuration and Management with Dedicated DMA Channels
  - On-chip Transceivers Including Pull-Ups
- One 2-channel Controller Area Network (CAN)
  - CAN2A and CAN2B protocol compliant, with high-level mailbox system
  - Two independent channels, 16 Message Objects per Channel

32-bit AVR® Microcontroller

AT32UC3C0512C
AT32UC3C0256C
AT32UC3C0128C
AT32UC3C064C
AT32UC3C1512C
AT32UC3C1256C
AT32UC3C1128C
AT32UC3C164C
AT32UC3C2512C
AT32UC3C2256C
AT32UC3C2128C
AT32UC3C264C

Preliminary

32117B–AVR–03/11

- One 4-Channel 20-bit Pulse Width Modulation Controller (PWM)
  - Complementary outputs, with Dead Time Insertion
  - Output Override and Fault Protection
- Two Quadrature Decoders
- One 16-channel 12-bit Pipelined Analog-To-Digital Converter (ADC)
  - Dual Sample and Hold Capability Allowing 2 Synchronous Conversions
  - Single-Ended and Differential Channels, Window Function
- Two 12-bit Digital-To-Analog Converters (DAC), with Dual Output Sample System
- Four Analog Comparators
- Six 16-bit Timer/Counter (TC) Channels
  - External Clock Inputs, PWM, Capture and Various Counting Capabilities
- One Peripheral Event Controller
  - Trigger Actions in Peripherals Depending on Events Generated from Peripherals or from Input Pins
  - Deterministic Trigger
  - 34 Events and 22 Event Actions
- Five Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
  - Independent Baudrate Generator, Support for SPI, LIN, IrDA and ISO7816 interfaces
  - Support for Hardware Handshaking, RS485 Interfaces and Modem Line
- Two Master/Slave Serial Peripheral Interfaces (SPI) with Chip Select Signals
- One Inter-IC Sound (I2S) Controller
  - Compliant with I2S Bus Specification
  - Time Division Multiplexed mode
- Three Master and Three Slave Two-Wire Interfaces (TWI), 400kbit/s I^2C-compatible
- QTouch® Library Support
  - Capacitive Touch Buttons, Sliders, and Wheels
  - QTouch® and QMatrix® Acquisition
- On-Chip Non-intrusive Debug System
  - Nexus Class 2+, Runtime Control, Non-Intrusive Data and Program Trace
  - aWire™ single-pin programming trace and debug interface muxed with reset pin
  - NanoTrace™ provides trace capabilities through JTAG or aWire interface
- 3 package options
  - 64-pin QFN/TQFP (45 GPIO pins)
  - 100-pin TQFP (81 GPIO pins)
  - 144-pin LQFP (123 GPIO pins)
- Two operating voltage ranges:
  - Single 5V Power Supply
  - Single 3.3V Power Supply

1. Description

The AT32UC3C is a complete System-On-Chip microcontroller based on the AVR32UC RISC processor running at frequencies up to 66 MHz. AVR32UC is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density and high performance.

The processor implements a Memory Protection Unit (MPU) and a fast and flexible interrupt controller for supporting modern operating systems and real-time operating systems. Using the Secure Access Unit (SAU) together with the MPU provides the required security and integrity.

Higher computation capabilities are achievable either using a rich set of DSP instructions or using the floating-point instructions.

The AT32UC3C incorporates on-chip Flash and SRAM memories for secure and fast access. For applications requiring additional memory, an external memory interface is provided on AT32UC3C0 derivatives.

The Memory Direct Memory Access controller (MDMA) enables transfers of block of data from memories to memories without processor involvement.

The Peripheral Direct Memory Access (PDCA) controller enables data transfers between peripherals and memories without processor involvement. The PDCA drastically reduces processing overhead when transferring continuous and large data streams.

The AT32UC3C incorporates on-chip Flash and SRAM memories for secure and fast access. The FlashVault technology allows secure libraries to be programmed into the device. The secure libraries can be executed while the CPU is in Secure State, but not read by non-secure software in the device. The device can thus be shipped to end custumers, who are able to program their own code into the device, accessing the secure libraries, without any risk of compromising the proprietary secure code.

The Power Manager improves design flexibility and security. Power monitoring is supported by on-chip Power-On Reset (POR), Brown-Out Detectors (BOD18, BOD33, BOD50). The CPU runs from the on-chip RC oscillators, the PLLs, or the Multipurpose Oscillators. The Asynchronous Timer (AST) combined with the 32 KHz oscillator keeps track of the time. The AST can operate in counter or calendar mode.

The device includes six identical 16-bit Timer/Counter (TC) channels. Each channel can be independently programmed to perform frequency measurement, event counting, interval measurement, pulse generation, delay timing, and pulse width modulation.

The PWM module provides four channels with many configuration options including polarity, edge alignment and waveform non overlap control. The PWM channels can operate independently, with duty cycles set independently from each other, or in interlinked mode, with multiple channels updated at the same time. It also includes safety feature with fault inputs and the ability to lock the PWM configuration registers and the PWM pin assignment.

The AT32UC3C also features many communication interfaces for communication intensive applications. In addition to standard serial interfaces like UART, SPI or TWI, other interfaces like flexible CAN, USB and Ethernet MAC are available. The USART supports different communication modes, like SPI mode and LIN mode.

The Inter-IC Sound Controller (I2SC) provides a 5-bit wide, bidirectional, synchronous, digital audio link with off-chip audio devices. The controller is compliant with the I2S bus specification.

The Full-Speed USB 2.0 Device interface supports several USB Classes at the same time thanks to the rich End-Point configuration. The On-The-GO (OTG) Host interface allows device like a USB Flash disk or a USB printer to be directly connected to the processor.

The media-independent interface (MII) and reduced MII (RMII) 10/100 Ethernet MAC module provides on-chip solutions for network-connected devices.

The Peripheral Event Controller (PEVC) allows to redirect events from one peripheral or from input pins to another peripheral. It can then trigger, in a deterministic time, an action inside a peripheral without the need of CPU. For instance a PWM waveform can directly trigger an ADC capture, hence avoiding delays due to software interrupt processing.

The AT32UC3C features analog functions like ADC, DAC, Analog comparators. The ADC interface is built around a 12-bit pipelined ADC core and is able to control two independent 8-channel or one 16-channel. The ADC block is able to measure two different voltages sampled at the same time. The analog comparators can be paired to detect when the sensing voltage is within or outside the defined reference window.

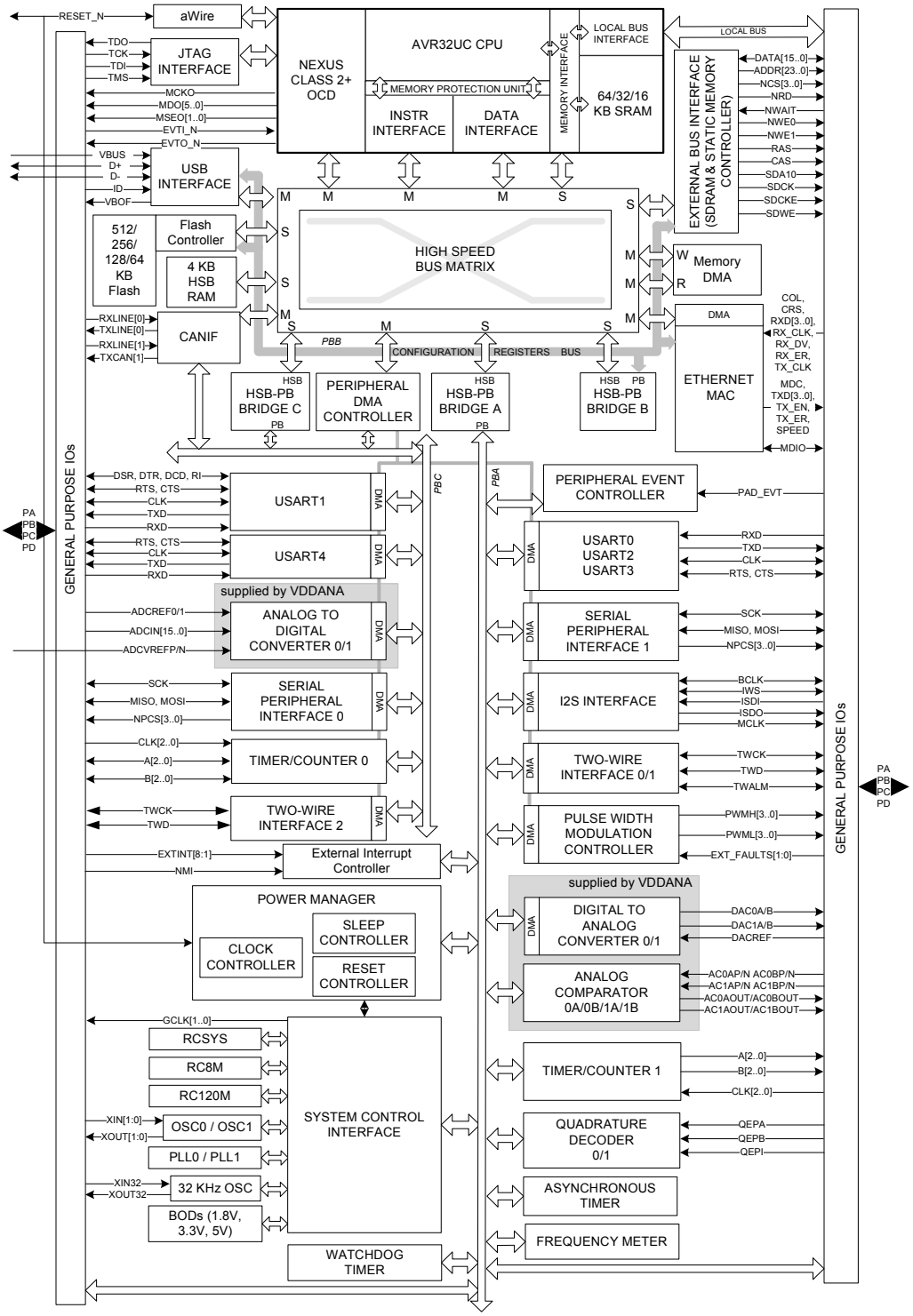
Atmel offers the QTouch library for embedding capacitive touch buttons, sliders, and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and included fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS®) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop, and debug your own touch applications.

AT32UC3C integrates a class 2+ Nexus 2.0 On-Chip Debug (OCD) System, with non-intrusive real-time trace, full-speed read/write memory access in addition to basic runtime control. The Nanotrace interface enables trace feature for aWire- or JTAG-based debuggers. The single-pin aWire interface allows all features available through the JTAG interface to be accessed through the RESET pin, allowing the JTAG pins to be used for GPIO or peripherals.

2. Overview

2.1 Block diagram

Figure 2-1. Block diagram



2.2 Configuration Summary

Table 2-1. Configuration Summary

Feature	AT32UC3C0512C/ AT32UC3C0256C/ AT32UC3C0128C/ AT32UC3C064C	AT32UC3C1512C/ AT32UC3C1256C/ AT32UC3C1128C/ AT32UC3C164C	AT32UC3C2512C/ AT32UC3C2256C/ AT32UC3C2128C/ AT32UC3C264C
Flash	512/256/128/64 KB	512/256/128/64 KB	512/256/128/64 KB
SRAM	64/64/32/16 KB	64/64/32/16 KB	64/64/32/16 KB
HSB RAM	4 KB		
EBI	1	0	0
GPIO	123	81	45
External Interrupts	8	8	8
TWI	3	3	2
USART	5	5	4
Peripheral DMA Channels	16	16	16
Peripheral Event System	1	1	1
SPI	2	2	1
CAN channels	2	2	2
USB	1	1	1
Ethernet MAC 10/100	1 RMII/MII	1 RMII/MII	1 MII only
I2S	1	1	1
Asynchronous Timers	1	1	1
Timer/Counter Channels	6	6	3
PWM channels	4x2		
QDEC	2	2	1
Frequency Meter	1		
Watchdog Timer	1		
Power Manager	1		
Oscillators	PLL 80-240 MHz (PLL0/PLL1) Crystal Oscillator 0.4-20 MHz (OSC0) Crystal Oscillator 32 KHz (OSC32K) RC Oscillator 115 kHz (RCSYS) RC Oscillator 8 MHz (RC8M) RC Oscillator 120 MHz (RC120M)		
	0.4-20 MHz (OSC1)		-
12-bit ADC number of channels	1 16	1 16	1 11
12-bit DAC number of channels	1 4	1 4	1 2

Table 2-1. Configuration Summary

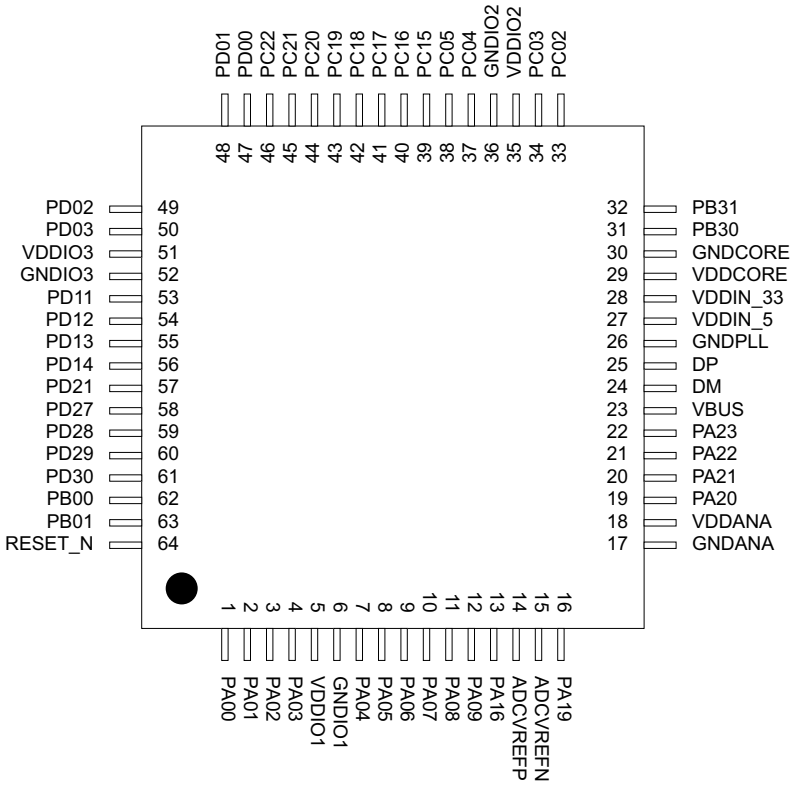
Feature	AT32UC3C0512C/ AT32UC3C0256C/ AT32UC3C0128C/ AT32UC3C064C	AT32UC3C1512C/ AT32UC3C1256C/ AT32UC3C1128C/ AT32UC3C164C	AT32UC3C2512C/ AT32UC3C2256C/ AT32UC3C2128C/ AT32UC3C264C
Analog Comparators	4	4	2
JTAG	1		
aWire	1		
Max Frequency	66 MHz		
Package	LQFP144	TQFP100	TQFP64/QFN64

3. Package and Pinout

3.1 Package

The device pins are multiplexed with peripheral functions as described in Table 3-1 on page 11.

Figure 3-1. QFN64/TQFP64 Pinout



Note: on QFN packages, the exposed pad is unconnected.

Figure 3-2. TQFP100 Pinout

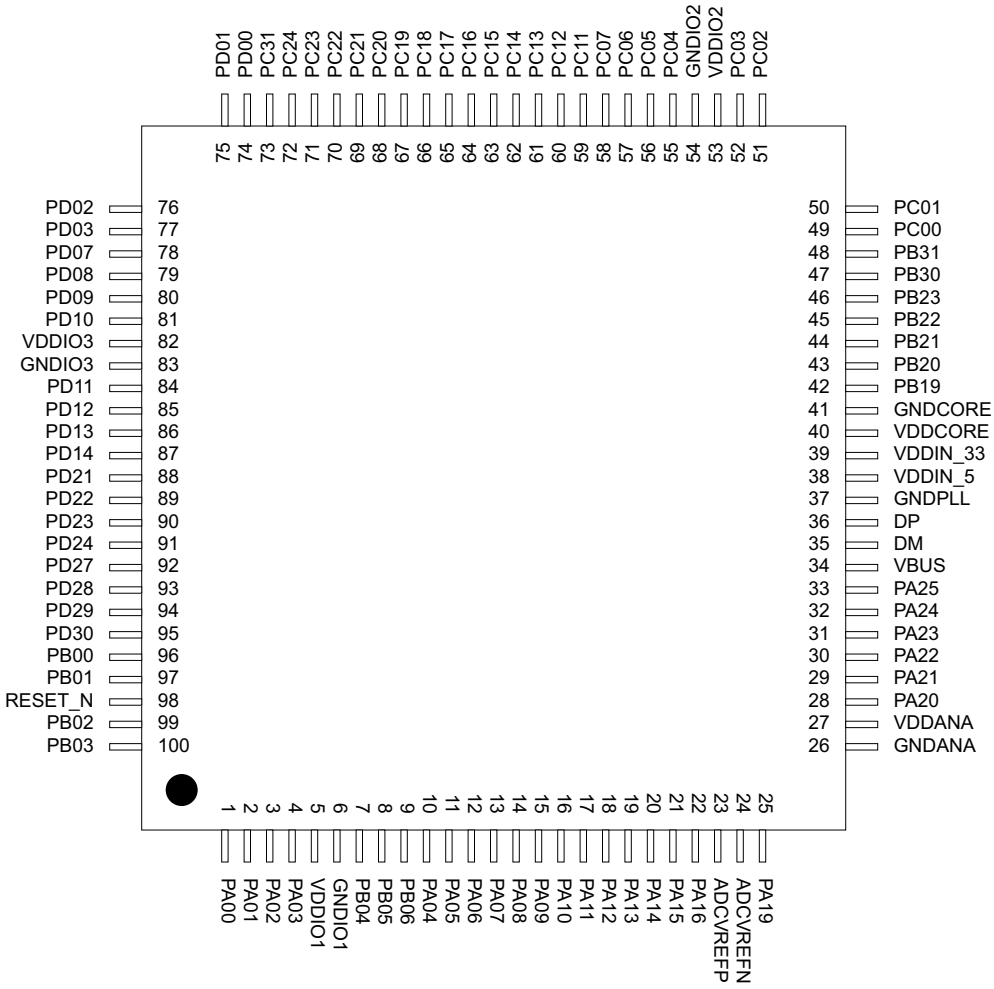
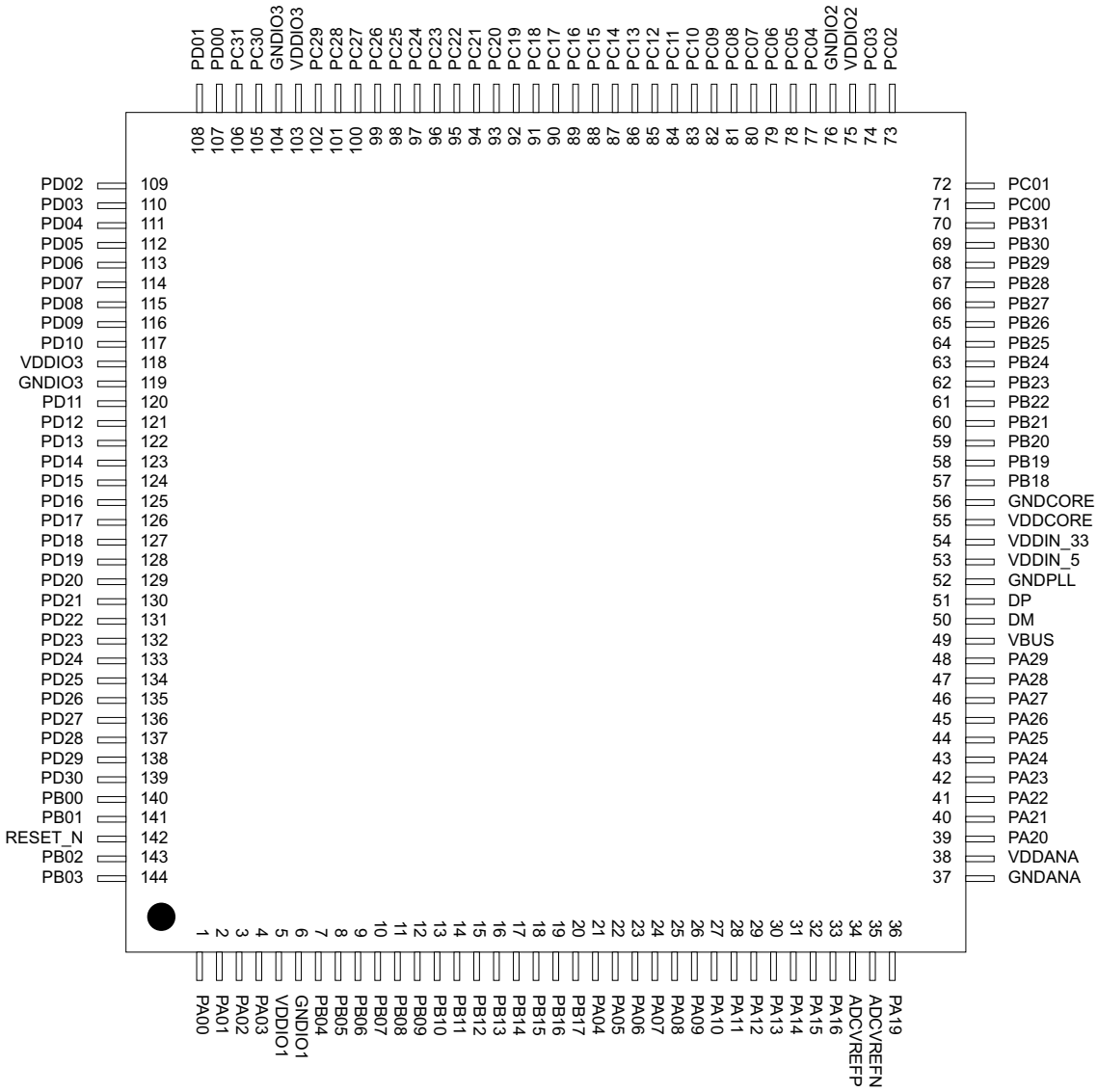


Figure 3-3. LQFP144 Pinout



3.2 Peripheral Multiplexing on I/O lines

3.2.1 Multiplexed Signals

Each GPIO line can be assigned to one of the peripheral functions. The following table describes the peripheral signals multiplexed to the GPIO lines.

Table 3-1. GPIO Controller Function Multiplexing

TQFP / QFN 64	TQFP 100	LQFP 144	PIN	GPIO	Supply	Pin Type (1)	GPIO function					
							A	B	C	D	E	F
1	1	1	PA00	0	VDDIO1	x1/x2		CANIF - TXLINE[1]				
2	2	2	PA01	1	VDDIO1	x1/x2		CANIF - RXLINE[1]	PEVC - PAD_EVT [0]			
3	3	3	PA02	2	VDDIO1	x1/x2	SCIF - GCLK[0]		PEVC - PAD_EVT [1]			
4	4	4	PA03	3	VDDIO1	x1/x2	SCIF - GCLK[1]	EIC - EXTINT[1]				
7	10	21	PA04	4	VDDANA	x1/x2	ADCIN0	USBC - ID	ACIFA0 - ACAOUT			
8	11	22	PA05	5	VDDANA	x1/x2	ADCIN1	USBC - VBOF	ACIFA0 - ACBOUT			
9	12	23	PA06	6	VDDANA	x1/x2	ADCIN2	AC1AP1	PEVC - PAD_EVT [2]			
10	13	24	PA07	7	VDDANA	x1/x2	ADCIN3	AC1AN1	PEVC - PAD_EVT [3]			
11	14	25	PA08	8	VDDANA	x1/x2	ADCIN4	AC1BP1	EIC - EXTINT[2]			
12	15	26	PA09	9	VDDANA	x1/x2	ADCIN5	AC1BN1				
	16	27	PA10	10	VDDANA	x1/x2	ADCIN6	EIC - EXTINT[4]	PEVC - PAD_EVT [13]			
	17	28	PA11	11	VDDANA	x1/x2	ADCIN7	ADCREF1	PEVC - PAD_EVT [14]			
	18	29	PA12	12	VDDANA	x1/x2	AC1AP0	SPI0 - NPCS[0]	AC1AP0 or DAC1A			
	19	30	PA13	13	VDDANA	x1/x2	AC1AN0	SPI0 - NPCS[1]	ADCIN15			
	20	31	PA14	14	VDDANA	x1/x2	AC1BP0	SPI1 - NPCS[0]				
	21	32	PA15	15	VDDANA	x1/x2	AC1BN0	SPI1 - NPCS[1]	AC1BN0 or DAC1B			
13	22	33	PA16	16	VDDANA	x1/x2	ADCREF0		DACREF			
14	23	34	ADC REFP									
15	24	35	ADC REFN									

Table 3-1. GPIO Controller Function Multiplexing

TQFP / QFN 64	TQFP 100	LQFP 144	PIN	GPIO	Supply	Pin Type (1)	GPIO function					
							A	B	C	D	E	F
16	25	36	PA19	19	VDDANA	x1/x2	ADCIN8	EIC - EXTINT[1]				
19	28	39	PA20	20	VDDANA	x1/x2	ADCIN9	AC0AP0	AC0AP0 or DAC0A			
20	29	40	PA21	21	VDDANA	x1/x2	ADCIN10	AC0BN0	AC0BN0 or DAC0B			
21	30	41	PA22	22	VDDANA	x1/x2	ADCIN11	AC0AN0	PEVC - PAD_EVT [4]		MACB - SPEED	
22	31	42	PA23	23	VDDANA	x1/x2	ADCIN12	AC0BP0	PEVC - PAD_EVT [5]		MACB - WOL	
	32	43	PA24	24	VDDANA	x1/x2	ADCIN13	SPI1 - NPCS[2]				
	33	44	PA25	25	VDDANA	x1/x2	ADCIN14	SPI1 - NPCS[3]	EIC - EXTINT[0]			
		45	PA26	26	VDDANA	x1/x2	AC0AP1	EIC - EXTINT[1]				
		46	PA27	27	VDDANA	x1/x2	AC0AN1	EIC - EXTINT[2]				
		47	PA28	28	VDDANA	x1/x2	AC0BP1	EIC - EXTINT[3]				
		48	PA29	29	VDDANA	x1/x2	AC0BN1	EIC - EXTINT[0]				
62	96	140	PB00	32	VDDIO1	x1	USART0 - CLK	CANIF - RXLINE[1]	EIC - EXTINT[8]	PEVC - PAD_EVT [10]		
63	97	141	PB01	33	VDDIO1	x1		CANIF - TXLINE[1]		PEVC - PAD_EVT [11]		
	99	143	PB02	34	VDDIO1	x1		USBC - ID	PEVC - PAD_EVT [6]	TC1 - A1		
	100	144	PB03	35	VDDIO1	x1		USBC - VBOF	PEVC - PAD_EVT [7]			
	7	7	PB04	36	VDDIO1	x1/x2	SPI1 - MOSI	CANIF - RXLINE[0]	QDEC1 - QEPI		MACB - TXD[2]	
	8	8	PB05	37	VDDIO1	x1/x2	SPI1 - MISO	CANIF - TXLINE[0]	PEVC - PAD_EVT [12]	USART3 - CLK	MACB - TXD[3]	
	9	9	PB06	38	VDDIO1	x2/x4	SPI1 - SCK		QDEC1 - QEPA	USART1 - CLK	MACB - TX_ER	
		10	PB07	39	VDDIO1	x1/x2	SPI1 - NPCS[0]	EIC - EXTINT[2]	QDEC1 - QEPB		MACB - RX_DV	
		11	PB08	40	VDDIO1	x1/x2	SPI1 - NPCS[1]	PEVC - PAD_EVT [1]	PWM - PWML[0]		MACB - RXD[0]	
		12	PB09	41	VDDIO1	x1/x2	SPI1 - NPCS[2]		PWM - PWMH[0]		MACB - RXD[1]	
		13	PB10	42	VDDIO1	x1/x2	USART1 - DTR	SPI0 - MOSI	PWM - PWML[1]			

Table 3-1. GPIO Controller Function Multiplexing

TQFP / QFN 64	TQFP 100	LQFP 144	PIN	GPIO	Supply	Pin Type (1)	GPIO function					
							A	B	C	D	E	F
		14	PB11	43	VDDIO1	x1/x2	USART1 - DSR	SPI0 - MISO	PWM - PWMH[1]			
		15	PB12	44	VDDIO1	x1/x2	USART1 - DCD	SPI0 - SCK	PWM - PWML[2]			
		16	PB13	45	VDDIO1	x1/x2	USART1 - RI	SPI0 - NPCS[0]	PWM - PWMH[2]		MACB - RX_ER	
		17	PB14	46	VDDIO1	x1/x2	USART1 - RTS	SPI0 - NPCS[1]	PWM - PWML[3]		MACB - MDC	
		18	PB15	47	VDDIO1	x1/x2	USART1 - CTS	USART1 - CLK	PWM - PWMH[3]		MACB - MDIO	
		19	PB16	48	VDDIO1	x1/x2	USART1 - RXD	SPI0 - NPCS[2]	PWM - EXT_ FAULTS[0]		CANIF - RXLINE[0]	
		20	PB17	49	VDDIO1	x1/x2	USART1 - TXD	SPI0 - NPCS[3]	PWM - EXT_ FAULTS[1]		CANIF - TXLINE[0]	
		57	PB18	50	VDDIO2	x1/x2	TC0 - CLK2		EIC - EXTINT[4]			
	42	58	PB19	51	VDDIO2	x1/x2	TC0 - A0	SPI1 - MOSI	IISC - ISDO		MACB - CRS	
	43	59	PB20	52	VDDIO2	x1/x2	TC0 - B0	SPI1 - MISO	IISC - ISDI	ACIFA1 - ACAOUT	MACB - COL	
	44	60	PB21	53	VDDIO2	x2/x4	TC0 - CLK1	SPI1 - SCK	IISC - IMCK	ACIFA1 - ACBOUT	MACB - RXD[2]	
	45	61	PB22	54	VDDIO2	x1/x2	TC0 - A1	SPI1 - NPCS[3]	IISC - ISCK	SCIF - GCLK[0]	MACB - RXD[3]	
	46	62	PB23	55	VDDIO2	x1/x2	TC0 - B1	SPI1 - NPCS[2]	IISC - IWS	SCIF - GCLK[1]	MACB - RX_CLK	
		63	PB24	56	VDDIO2	x1/x2	TC0 - CLK0	SPI1 - NPCS[1]				
		64	PB25	57	VDDIO2	x1/x2	TC0 - A2	SPI1 - NPCS[0]	PEVC - PAD_EVT [8]			
		65	PB26	58	VDDIO2	x2/x4	TC0 - B2	SPI1 - SCK	PEVC - PAD_EVT [9]		MACB - TX_EN	
		66	PB27	59	VDDIO2	x1/x2	QDEC0 - QEPA	SPI1 - MISO	PEVC - PAD_EVT [10]	TC1 - CLK0	MACB - TXD[0]	
		67	PB28	60	VDDIO2	x1/x2	QDEC0 - QEPB	SPI1 - MOSI	PEVC - PAD_EVT [11]	TC1 - B0	MACB - TXD[1]	
		68	PB29	61	VDDIO2	x1/x2	QDEC0 - QEPI	SPI0 - NPCS[0]	PEVC - PAD_EVT [12]	TC1 - A0		
31	47	69	PB30	62	VDDIO2	x1						
32	48	70	PB31	63	VDDIO2	x1						
	49	71	PC00	64	VDDIO2	x1/x2	USBC - ID	SPI0 - NPCS[1]	USART2 - CTS	TC1 - B2	CANIF - TXLINE[1]	
	50	72	PC01	65	VDDIO2	x1/x2	USBC - VBOF	SPI0 - NPCS[2]	USART2 - RTS	TC1 - A2	CANIF - RXLINE[1]	

Table 3-1. GPIO Controller Function Multiplexing

TQFP / QFN 64	TQFP 100	LQFP 144	PIN	GPIO	Supply	Pin Type (1)	GPIO function					
							A	B	C	D	E	F
33	51	73	PC02	66	VDDIO2	x1	TWIMS0 - TWD	SPI0 - NPCS[3]	USART2 - RXD	TC1 - CLK1	MACB - MDC	
34	52	74	PC03	67	VDDIO2	x1	TWIMS0 - TWCK	EIC - EXTINT[1]	USART2 - TXD	TC1 - B1	MACB - MDIO	
37	55	77	PC04	68	VDDIO2	x1	TWIMS1 - TWD	EIC - EXTINT[3]	USART2 - TXD	TC0 - B1		
38	56	78	PC05	69	VDDIO2	x1	TWIMS1 - TWCK	EIC - EXTINT[4]	USART2 - RXD	TC0 - A2		
	57	79	PC06	70	VDDIO2	x1	PEVC - PAD_EVT [15]	USART2 - CLK	USART2 - CTS	TC0 - CLK2	TWIMS2 - TWD	TWIMS0 - TWALM
	58	80	PC07	71	VDDIO2	x1	PEVC - PAD_EVT [2]	EBI - NCS[3]	USART2 - RTS	TC0 - B2	TWIMS2 - TWCK	TWIMS1 - TWALM
		81	PC08	72	VDDIO2	x1/x2	PEVC - PAD_EVT [13]	SPI1 - NPCS[1]	EBI - NCS[0]		USART4 - TXD	
		82	PC09	73	VDDIO2	x1/x2	PEVC - PAD_EVT [14]	SPI1 - NPCS[2]	EBI - ADDR[23]		USART4 - RXD	
		83	PC10	74	VDDIO2	x1/x2	PEVC - PAD_EVT [15]	SPI1 - NPCS[3]	EBI - ADDR[22]			
	59	84	PC11	75	VDDIO2	x1/x2	PWM - PWMH[3]	CANIF - RXLINE[1]	EBI - ADDR[21]	TC0 - CLK0		
	60	85	PC12	76	VDDIO2	x1/x2	PWM - PWML[3]	CANIF - TXLINE[1]	EBI - ADDR[20]	USART2 - CLK		
	61	86	PC13	77	VDDIO2	x1/x2	PWM - PWMH[2]	EIC - EXTINT[7]		USART0 - RTS		
	62	87	PC14	78	VDDIO2	x1/x2	PWM - PWML[2]	USART0 - CLK	EBI - SDCKE	USART0 - CTS		
39	63	88	PC15	79	VDDIO2	x1/x2	PWM - PWMH[1]	SPI0 - NPCS[0]	EBI - SDWE	USART0 - RXD	CANIF - RXLINE[1]	
40	64	89	PC16	80	VDDIO2	x1/x2	PWM - PWML[1]	SPI0 - NPCS[1]	EBI - CAS	USART0 - TXD	CANIF - TXLINE[1]	
41	65	90	PC17	81	VDDIO2	x1/x2	PWM - PWMH[0]	SPI0 - NPCS[2]	EBI - RAS	IISC - ISDO		USART3 - TXD
42	66	91	PC18	82	VDDIO2	x1/x2	PWM - PWML[0]	EIC - EXTINT[5]	EBI - SDA10	IISC - ISDI		USART3 - RXD
43	67	92	PC19	83	VDDIO3	x1/x2	PWM - PWML[2]	SCIF - GCLK[0]	EBI - DATA[0]	IISC - IMCK		USART3 - CTS
44	68	93	PC20	84	VDDIO3	x1/x2	PWM - PWMH[2]	SCIF - GCLK[1]	EBI - DATA[1]	IISC - ISCK		USART3 - RTS
45	69	94	PC21	85	VDDIO3	x1/x2	PWM - EXT_ FAULTS[0]	CANIF - RXLINE[0]	EBI - DATA[2]	IISC - IWS		
46	70	95	PC22	86	VDDIO3	x1/x2	PWM - EXT_ FAULTS[1]	CANIF - TXLINE[0]	EBI - DATA[3]		USART3 - CLK	
	71	96	PC23	87	VDDIO3	x1/x2	QDEC1 - QEPB	CANIF - RXLINE[1]	EBI - DATA[4]	PEVC - PAD_EVT [3]		

Table 3-1. GPIO Controller Function Multiplexing

TQFP / QFN 64	TQFP 100	LQFP 144	PIN	GPIO	Supply	Pin Type (1)	GPIO function					
							A	B	C	D	E	F
	72	97	PC24	88	VDDIO3	x1/x2	QDEC1 - QEPA	CANIF - TXLINE[1]	EBI - DATA[5]	PEVC - PAD_EVT [4]		
		98	PC25	89	VDDIO3	x1/x2		TC1 - CLK2	EBI - DATA[6]	SCIF - GCLK[0]	USART4 - TXD	
		99	PC26	90	VDDIO3	x1/x2	QDEC1 - QEPI	TC1 - B2	EBI - DATA[7]	SCIF - GCLK[1]	USART4 - RXD	
		100	PC27	91	VDDIO3	x1/x2		TC1 - A2	EBI - DATA[8]	EIC - EXTINT[0]	USART4 - CTS	
		101	PC28	92	VDDIO3	x1/x2	SPI1 - NPCS[3]	TC1 - CLK1	EBI - DATA[9]		USART4 - RTS	
		102	PC29	93	VDDIO3	x1/x2	SPI0 - NPCS[1]	TC1 - B1	EBI - DATA[10]			
		105	PC30	94	VDDIO3	x1/x2	SPI0 - NPCS[2]	TC1 - A1	EBI - DATA[11]			
	73	106	PC31	95	VDDIO3	x1/x2	SPI0 - NPCS[3]	TC1 - B0	EBI - DATA[12]	PEVC - PAD_EVT [5]	USART4 - CLK	
47	74	107	PD00	96	VDDIO3	x1/x2	SPI0 - MOSI	TC1 - CLK0	EBI - DATA[13]	QDEC0 - QEPI	USART0 - TXD	
48	75	108	PD01	97	VDDIO3	x1/x2	SPI0 - MISO	TC1 - A0	EBI - DATA[14]	TC0 - CLK1	USART0 - RXD	
49	76	109	PD02	98	VDDIO3	x2/x4	SPI0 - SCK	TC0 - CLK2	EBI - DATA[15]	QDEC0 - QEPA		
50	77	110	PD03	99	VDDIO3	x1/x2	SPI0 - NPCS[0]	TC0 - B2	EBI - ADDR[0]	QDEC0 - QEPB		
		111	PD04	100	VDDIO3	x1/x2	SPI0 - MOSI		EBI - ADDR[1]			
		112	PD05	101	VDDIO3	x1/x2	SPI0 - MISO		EBI - ADDR[2]			
		113	PD06	102	VDDIO3	x2/x4	SPI0 - SCK		EBI - ADDR[3]			
	78	114	PD07	103	VDDIO3	x1/x2	USART1 - DTR	EIC - EXTINT[5]	EBI - ADDR[4]	QDEC0 - QEPI	USART4 - TXD	
	79	115	PD08	104	VDDIO3	x1/x2	USART1 - DSR	EIC - EXTINT[6]	EBI - ADDR[5]	TC1 - CLK2	USART4 - RXD	
	80	116	PD09	105	VDDIO3	x1/x2	USART1 - DCD	CANIF - RXLINE[0]	EBI - ADDR[6]	QDEC0 - QEPA	USART4 - CTS	
	81	117	PD10	106	VDDIO3	x1/x2	USART1 - RI	CANIF - TXLINE[0]	EBI - ADDR[7]	QDEC0 - QEPB	USART4 - RTS	
53	84	120	PD11	107	VDDIO3	x1/x2	USART1 - TXD	USBC - ID	EBI - ADDR[8]	PEVC - PAD_EVT [6]	MACB - TXD[0]	
54	85	121	PD12	108	VDDIO3	x1/x2	USART1 - RXD	USBC - VBOF	EBI - ADDR[9]	PEVC - PAD_EVT [7]	MACB - TXD[1]	
55	86	122	PD13	109	VDDIO3	x2/x4	USART1 - CTS	USART1 - CLK	EBI - SDCK	PEVC - PAD_EVT [8]	MACB - RXD[0]	
56	87	123	PD14	110	VDDIO3	x1/x2	USART1 - RTS	EIC - EXTINT[7]	EBI - ADDR[10]	PEVC - PAD_EVT [9]	MACB - RXD[1]	

Table 3-1. GPIO Controller Function Multiplexing

TQFP / QFN 64	TQFP 100	LQFP 144	PIN	GPIO	Supply	Pin Type (1)	GPIO function					
							A	B	C	D	E	F
		124	PD15	111	VDDIO3	x1/x2	TC0 - A0	USART3 - TXD	EBI - ADDR[11]			
		125	PD16	112	VDDIO3	x1/x2	TC0 - B0	USART3 - RXD	EBI - ADDR[12]			
		126	PD17	113	VDDIO3	x1/x2	TC0 - A1	USART3 - CTS	EBI - ADDR[13]	USART3 - CLK		
		127	PD18	114	VDDIO3	x1/x2	TC0 - B1	USART3 - RTS	EBI - ADDR[14]			
		128	PD19	115	VDDIO3	x1/x2	TC0 - A2		EBI - ADDR[15]			
		129	PD20	116	VDDIO3	x1/x2	TC0 - B2		EBI - ADDR[16]			
57	88	130	PD21	117	VDDIO3	x1/x2	USART3 - TXD	EIC - EXTINT[0]	EBI - ADDR[17]	QDEC1 - QEPI		
	89	131	PD22	118	VDDIO1	x1/x2	USART3 - RXD	TC0 - A2	EBI - ADDR[18]	SCIF - GCLK[0]		
	90	132	PD23	119	VDDIO1	x1/x2	USART3 - CTS	USART3 - CLK	EBI - ADDR[19]	QDEC1 - QEPA		
	91	133	PD24	120	VDDIO1	x1/x2	USART3 - RTS	EIC - EXTINT[8]	EBI - NWE1	QDEC1 - QEPB		
		134	PD25	121	VDDIO1	x1/x2	TC0 - CLK0	USBC - ID	EBI - NWE0		USART4 - CLK	
		135	PD26	122	VDDIO1	x1/x2	TC0 - CLK1	USBC - VBOF	EBI - NRD			
58	92	136	PD27	123	VDDIO1	x1/x2	USART0 - TXD	CANIF - RXLINE[0]	EBI - NCS[1]	TC0 - A0	MACB - RX_ER	
59	93	137	PD28	124	VDDIO1	x1/x2	USART0 - RXD	CANIF - TXLINE[0]	EBI - NCS[2]	TC0 - B0	MACB - RX_DV	
60	94	138	PD29	125	VDDIO1	x1/x2	USART0 - CTS	EIC - EXTINT[6]	USART0 - CLK	TC0 - CLK0	MACB - TX_CLK	
61	95	139	PD30	126	VDDIO1	x1/x2	USART0 - RTS	EIC - EXTINT[3]	EBI - NWAIT	TC0 - A1	MACB - TX_EN	

Note: 1. Refer to "Electrical Characteristics" on page 1249 for a description of the electrical properties of the pin types used.

See Section 3.3 for a description of the various peripheral signals.

3.2.2 Peripheral Functions

Each GPIO line can be assigned to one of several peripheral functions. The following table describes how the various peripheral functions are selected. The last listed function has priority in case multiple functions are enabled on the same pin.

Table 3-2. Peripheral Functions

Function	Description
GPIO Controller Function multiplexing	GPIO and GPIO peripheral selection A to F
Nexus OCD AUX port connections	OCD trace system

Table 3-2. Peripheral Functions

Function	Description
aWire DATAOUT	aWire output in two-pin mode
JTAG port connections	JTAG debug port
Oscillators	OSC0, OSC32

3.2.3 Oscillator Pinout

The oscillators are not mapped to the normal GPIO functions and their muxings are controlled by registers in the System Control Interface (SCIF). Please refer to the SCIF chapter for more information about this.

Table 3-3. Oscillator pinout

QFN64/ TQFP64 pin	TQFP100 pin	LQFP144 pin	Pad	Oscillator pin
31	47	69	PB30	xin0
	99	143	PB02	xin1
62	96	140	PB00	xin32
32	48	70	PB31	xout0
	100	144	PB03	xout1
63	97	141	PB01	xout32

3.2.4 JTAG port connections

If the JTAG is enabled, the JTAG will take control over a number of pins, irrespectively of the I/O Controller configuration.

Table 3-4. JTAG pinout

QFN64/ TQFP64 pin	TQFP100 pin	LQFP144 pin	Pin name	JTAG pin
2	2	2	PA01	TDI
3	3	3	PA02	TDO
4	4	4	PA03	TMS
1	1	1	PA00	TCK

3.2.5 Nexus OCD AUX port connections

If the OCD trace system is enabled, the trace system will take control over a number of pins, irrespectively of the GPIO configuration. Two different OCD trace pin mappings are possible,

depending on the configuration of the OCD AXS register. For details, see the AVR32UC Technical Reference Manual.

Table 3-5. Nexus OCD AUX port connections

Pin	AXS=0	AXS=1	AXS=2
EVTI_N	PA08	PB19	PA10
MDO[5]	PC05	PC31	PB06
MDO[4]	PC04	PC12	PB15
MDO[3]	PA23	PC11	PB14
MDO[2]	PA22	PB23	PA27
MDO[1]	PA19	PB22	PA26
MDO[0]	PA09	PB20	PA19
EVTO_N	PD29	PD29	PD29
MCKO	PD13	PB21	PB26
MSEO[1]	PD30	PD08	PB25
MSEO[0]	PD14	PD07	PB18

3.2.6 Other Functions

The functions listed in Table 3-6 are not mapped to the normal GPIO functions. The aWire DATA pin will only be active after the aWire is enabled. The aWire DATAOUT pin will only be active after the aWire is enabled and the 2_PIN_MODE command has been sent.

Table 3-6. Other Functions

QFN64/ TQFP64 pin	TQFP100 pin	LQFP144 pin	Pad	Oscillator pin
64	98	142	RESET_N	aWire DATA
3	3	3	PA02	aWire DATAOUT

3.3 Signals Description

The following table give details on the signal name classified by peripherals.

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power				
VDDIO1 VDDIO2 VDDIO3	I/O Power Supply	Power Input		4.5V to 5.5V or 3.0V to 3.6 V
VDDANA	Analog Power Supply	Power Input		4.5V to 5.5V or 3.0V to 3.6 V

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
VDDIN_5	1.8V Voltage Regulator Input	Power Input		Power Supply: 4.5V to 5.5V or 3.0V to 3.6 V
VDDIN_33	USB I/O power supply	Power Output/ Input		Capacitor Connection for the 3.3V voltage regulator or power supply: 3.0V to 3.6 V
VDDCORE	1.8V Voltage Regulator Output	Power output		Capacitor Connection for the 1.8V voltage regulator
GNDIO1 GNDIO2 GNDIO3	I/O Ground	Ground		
GNDANA	Analog Ground	Ground		
GNDCORE	Ground of the core	Ground		
GNDPLL	Ground of the PLLs	Ground		
Analog Comparator Interface - ACIFA0/1				
AC0AN1/AC0AN0	Negative inputs for comparator AC0A	Analog		
AC0AP1/AC0AP0	Positive inputs for comparator AC0A	Analog		
AC0BN1/AC0BN0	Negative inputs for comparator AC0B	Analog		
AC0BP1/AC0BP0	Positive inputs for comparator AC0B	Analog		
AC1AN1/AC1AN0	Negative inputs for comparator AC1A	Analog		
AC1AP1/AC1AP0	Positive inputs for comparator AC1A	Analog		
AC1BN1/AC1BN0	Negative inputs for comparator AC1B	Analog		
AC1BP1/AC1BP0	Positive inputs for comparator AC1B	Analog		
ACAOUT/ACBOUT	analog comparator outputs	output		
ADC Interface - ADCIFA				
ADCIN[15:0]	ADC input pins	Analog		
ADCREF0	Analog positive reference 0 voltage input	Analog		
ADCREF1	Analog positive reference 1 voltage input	Analog		
ADCVREFP	Analog positive reference connected to external capacitor	Analog		

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
ADCVREFN	Analog negative reference connected to external capacitor	Analog		
Auxiliary Port - AUX				
MCKO	Trace Data Output Clock	Output		
MDO[5:0]	Trace Data Output	Output		
MSEO[1:0]	Trace Frame Control	Output		
EVTI_N	Event In	Output	Low	
EVTO_N	Event Out	Output	Low	
aWire - AW				
DATA	aWire data	I/O		
DATAOUT	aWire data output for 2-pin mode	I/O		
Controller Area Network Interface - CANIF				
RXLINE[1:0]	CAN channel rxline	I/O		
TXLINE[1:0]	CAN channel txline	I/O		
DAC Interface - DACIFB0/1				
DAC0A, DAC0B	DAC0 output pins of S/H A	Analog		
DAC1A, DAC1B	DAC output pins of S/H B	Analog		
DACREF	Analog reference voltage input	Analog		
External Bus Interface - EBI				
ADDR[23:0]	Address Bus	Output		
CAS	Column Signal	Output	Low	
DATA[15:0]	Data Bus	I/O		
NCS[3:0]	Chip Select	Output	Low	
NRD	Read Signal	Output	Low	
NWAIT	External Wait Signal	Input	Low	
NWE0	Write Enable 0	Output	Low	
NWE1	Write Enable 1	Output	Low	
RAS	Row Signal	Output	Low	
SDA10	SDRAM Address 10 Line	Output		

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
SDCK	SDRAM Clock	Output		
SDCKE	SDRAM Clock Enable	Output		
SDWE	SDRAM Write Enable	Output	Low	
External Interrupt Controller - EIC				
EXTINT[8:1]	External Interrupt Pins	Input		
NMI_N = EXTINT[0]	Non-Maskable Interrupt Pin	Input	Low	
General Purpose Input/Output - GPIOA, GPIOB, GPIOC, GPIOD				
PA[29:19] - PA[16:0]	Parallel I/O Controller GPIOA	I/O		
PB[31:0]	Parallel I/O Controller GPIOB	I/O		
PC[31:0]	Parallel I/O Controller GPIOC	I/O		
PD[30:0]	Parallel I/O Controller GPIOD	I/O		
Inter-IC Sound (I2S) Controller - IISC				
IMCK	I2S Master Clock	Output		
ISCK	I2S Serial Clock	I/O		
ISDI	I2S Serial Data In	Input		
ISDO	I2S Serial Data Out	Output		
IWS	I2S Word Select	I/O		
JTAG				
TCK	Test Clock	Input		
TDI	Test Data In	Input		
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		
Ethernet MAC - MACB				
COL	Collision Detect	Input		
CRS	Carrier Sense and Data Valid	Input		
MDC	Management Data Clock	Output		
MDIO	Management Data Input/Output	I/O		
RXD[3:0]	Receive Data	Input		

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
RX_CLK	Receive Clock	Input		
RX_DV	Receive Data Valid	Input		
RX_ER	Receive Coding Error	Input		
SPEED	Speed	Output		
TXD[3:0]	Transmit Data	Output		
TX_CLK	Transmit Clock or Reference Clock	Input		
TX_EN	Transmit Enable	Output		
TX_ER	Transmit Coding Error	Output		
WOL	Wake-On-LAN	Output		
Peripheral Event Controller - PEVC				
PAD_EVT[15:0]	Event Input Pins	Input		
Power Manager - PM				
RESET_N	Reset Pin	Input	Low	
Pulse Width Modulator - PWM				
PWMH[3:0] PWML[3:0]	PWM Output Pins	Output		
EXT_FAULT[1:0]	PWM Fault Input Pins	Input		
Quadrature Decoder- QDEC0/QDEC1				
QEPA	QEPA quadrature input	Input		
QEPB	QEPB quadrature input	Input		
QEPI	Index input	Input		
System Controller Interface- SCIF				
XIN0, XIN1, XIN32	Crystal 0, 1, 32K Inputs	Analog		
XOUT0, XOUT1, XOUT32	Crystal 0, 1, 32K Output	Analog		
GCLK0 - GCLK1	Generic Clock Pins	Output		
Serial Peripheral Interface - SPI0, SPI1				
MISO	Master In Slave Out	I/O		
MOSI	Master Out Slave In	I/O		

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
NPCS[3:0]	SPI Peripheral Chip Select	I/O	Low	
SCK	Clock	Output		
Timer/Counter - TC0, TC1				
A0	Channel 0 Line A	I/O		
A1	Channel 1 Line A	I/O		
A2	Channel 2 Line A	I/O		
B0	Channel 0 Line B	I/O		
B1	Channel 1 Line B	I/O		
B2	Channel 2 Line B	I/O		
CLK0	Channel 0 External Clock Input	Input		
CLK1	Channel 1 External Clock Input	Input		
CLK2	Channel 2 External Clock Input	Input		
Two-wire Interface - TWIMS0, TWIMS1, TWIMS2				
TWALM	SMBus SMBALERT	I/O	Low	Only on TWIMS0, TWIMS1
TWCK	Serial Clock	I/O		
TWD	Serial Data	I/O		
Universal Synchronous Asynchronous Receiver Transmitter - USART0, USART1, USART2, USART3, USART4				
CLK	Clock	I/O		
CTS	Clear To Send	Input	Low	
DCD	Data Carrier Detect	Input	Low	Only USART1
DSR	Data Set Ready	Input	Low	Only USART1
DTR	Data Terminal Ready	Output	Low	Only USART1
RI	Ring Indicator	Input	Low	Only USART1
RTS	Request To Send	Output	Low	
RXD	Receive Data	Input		
TXD	Transmit Data	Output		
Universal Serial Bus Device - USB				
DM	USB Device Port Data -	Analog		

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
DP	USB Device Port Data +	Analog		
VBUS	USB VBUS Monitor and OTG Negociation	Analog Input		
ID	ID Pin of the USB Bus	Input		
VBOF	USB VBUS On/off: bus power control port	output		

3.4 I/O Line Considerations

3.4.1 JTAG pins

The JTAG is enabled if TCK is low while the RESET_N pin is released. The TCK, TMS, and TDI pins have pull-up resistors when JTAG is enabled. The TCK pin always have pull-up enabled during reset. The TDO pin is an output, driven at VDDIO1, and has no pull-up resistor. The JTAG pins can be used as GPIO pins and muxed with peripherals when the JTAG is disabled. Please refer to Section 3.2.4 for the JTAG port connections.

3.4.2 RESET_N pin

The RESET_N pin integrates a pull-up resistor to VDDIO1. As the product integrates a power-on reset cell, the RESET_N pin can be left unconnected in case no reset from the system needs to be applied to the product.

The RESET_N pin is also used for the aWire debug protocol. When the pin is used for debugging, it must not be driven by external circuitry.

3.4.3 TWI pins

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with inputs with spike-filtering. When used as GPIO-pins or used for other peripherals, the pins have the same characteristics as GPIO pins.

3.4.4 GPIO pins

All I/O lines integrate programmable pull-up and pull-down resistors. Most I/O lines integrate drive strength control, see Table 3-1. Programming of this pull-up and pull-down resistor or this drive strength is performed independently for each I/O line through the GPIO Controllers.

After reset, I/O lines default as inputs with pull-up/pull-down resistors disabled. After reset, output drive strength is configured to the lowest value to reduce global EMI of the device.

When the I/O line is configured as analog function (ADC I/O, AC inputs, DAC I/O), the pull-up and pull-down resistors are automatically disabled.

4. Processor and Architecture

Rev: 2.1.2.0

This chapter gives an overview of the AVR32UC CPU. AVR32UC is an implementation of the AVR32 architecture. A summary of the programming model, instruction set, and MPU is presented. For further details, see the *AVR32 Architecture Manual* and the *AVR32UC Technical Reference Manual*.

4.1 Features

- **32-bit load/store AVR32A RISC architecture**
  - **15 general-purpose 32-bit registers**
  - **32-bit Stack Pointer, Program Counter and Link Register reside in register file**
  - **Fully orthogonal instruction set**
  - **Privileged and unprivileged modes enabling efficient and secure operating systems**
  - **Innovative instruction set together with variable instruction length ensuring industry leading code density**
  - **DSP extension with saturating arithmetic, and a wide variety of multiply instructions**
- **3-stage pipeline allowing one instruction per clock cycle for most instructions**
  - **Byte, halfword, word, and double word memory access**
  - **Multiple interrupt priority levels**
- **MPU allows for operating systems with memory protection**
- **FPU enables hardware accelerated floating point calculations**
- **Secure State for supporting FlashVault™ technology**

4.2 AVR32 Architecture

AVR32 is a new, high-performance 32-bit RISC microprocessor architecture, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption and high code density. In addition, the instruction set architecture has been tuned to allow a variety of microarchitectures, enabling the AVR32 to be implemented as low-, mid-, or high-performance processors. AVR32 extends the AVR family into the world of 32- and 64-bit applications.

Through a quantitative approach, a large set of industry recognized benchmarks has been compiled and analyzed to achieve the best code density in its class. In addition to lowering the memory requirements, a compact code size also contributes to the core's low power characteristics. The processor supports byte and halfword data types without penalty in code size and performance.

Memory load and store operations are provided for byte, halfword, word, and double word data with automatic sign- or zero extension of halfword and byte data. The C-compiler is closely linked to the architecture and is able to exploit code optimization features, both for size and speed.

In order to reduce code size to a minimum, some instructions have multiple addressing modes. As an example, instructions with immediates often have a compact format with a smaller immediate, and an extended format with a larger immediate. In this way, the compiler is able to use the format giving the smallest code size.

Another feature of the instruction set is that frequently used instructions, like add, have a compact format with two operands as well as an extended format with three operands. The larger format increases performance, allowing an addition and a data move in the same instruction in a

single cycle. Load and store instructions have several different formats in order to reduce code size and speed up execution.

The register file is organized as sixteen 32-bit registers and includes the Program Counter, the Link Register, and the Stack Pointer. In addition, register R12 is designed to hold return values from function calls and is used implicitly by some instructions.

4.3 The AVR32UC CPU

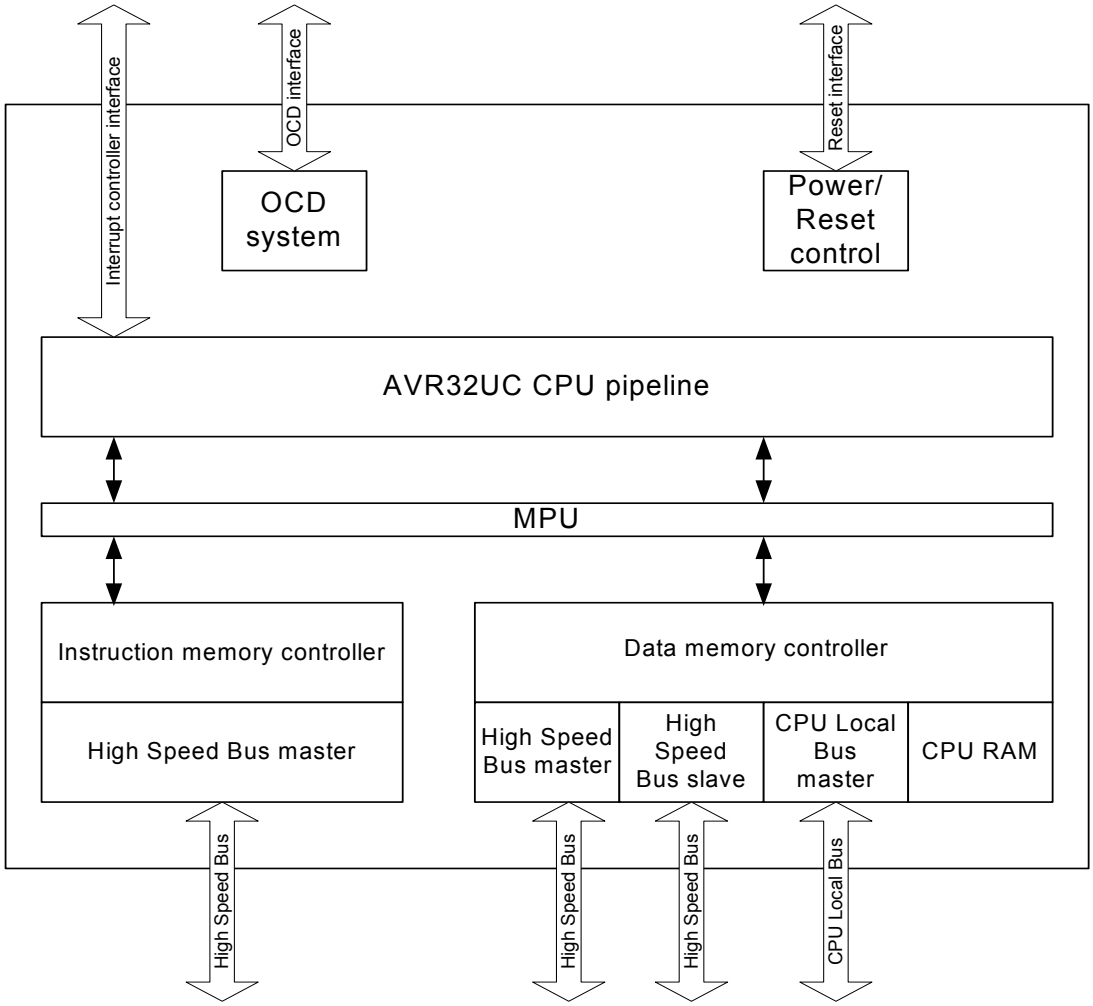
The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced On-Chip Debug (OCD) system, no caches, and a Memory Protection Unit (MPU). A hardware Floating Point Unit (FPU) is also provided through the coprocessor instruction space. Java acceleration hardware is not implemented.

AVR32UC provides three memory interfaces, one High Speed Bus master for instruction fetch, one High Speed Bus master for data access, and one High Speed Bus slave interface allowing other bus masters to access data RAMs internal to the CPU. Keeping data RAMs internal to the CPU allows fast access to the RAMs, reduces latency, and guarantees deterministic timing. Also, power consumption is reduced by not needing a full High Speed Bus access for memory accesses. A dedicated data RAM interface is provided for communicating with the internal data RAMs.

A local bus interface is provided for connecting the CPU to device-specific high-speed systems, such as floating-point units and I/O controller ports. This local bus has to be enabled by writing a one to the LOCEN bit in the CPUCR system register. The local bus is able to transfer data between the CPU and the local bus slave in a single clock cycle. The local bus has a dedicated memory range allocated to it, and data transfers are performed using regular load and store instructions. Details on which devices that are mapped into the local bus space is given in the CPU Local Bus section in the Memories chapter.

Figure 4-1 on page 27 displays the contents of AVR32UC.

Figure 4-1. Overview of the AVR32UC CPU



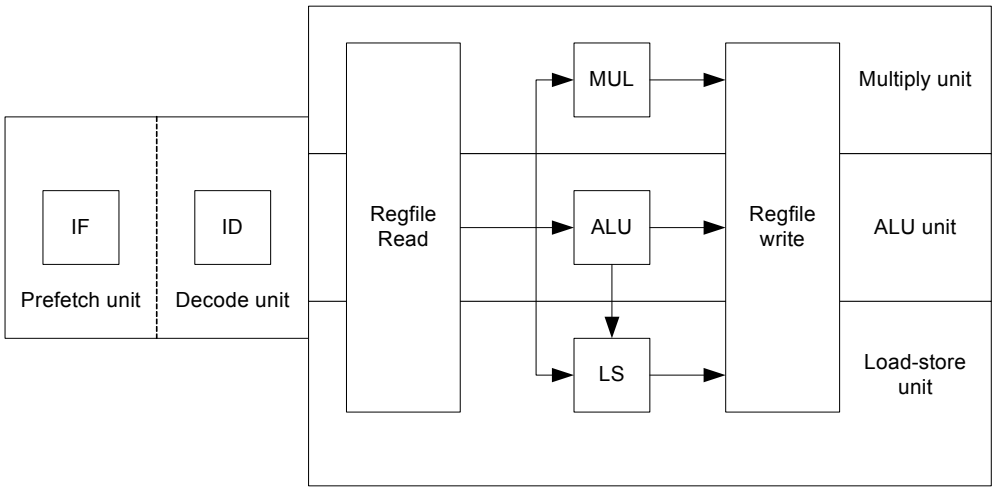
4.3.1 Pipeline Overview

AVR32UC has three pipeline stages, Instruction Fetch (IF), Instruction Decode (ID), and Instruction Execute (EX). The EX stage is split into three parallel subsections, one arithmetic/logic (ALU) section, one multiply (MUL) section, and one load/store (LS) section.

Instructions are issued and complete in order. Certain operations require several clock cycles to complete, and in this case, the instruction resides in the ID and EX stages for the required number of clock cycles. Since there is only three pipeline stages, no internal data forwarding is required, and no data dependencies can arise in the pipeline.

Figure 4-2 on page 28 shows an overview of the AVR32UC pipeline stages.

Figure 4-2. The AVR32UC Pipeline



4.3.2 AVR32A Microarchitecture Compliance

AVR32UC implements an AVR32A microarchitecture. The AVR32A microarchitecture is targeted at cost-sensitive, lower-end applications like smaller microcontrollers. This microarchitecture does not provide dedicated hardware registers for shadowing of register file registers in interrupt contexts. Additionally, it does not provide hardware registers for the return address registers and return status registers. Instead, all this information is stored on the system stack. This saves chip area at the expense of slower interrupt handling.

4.3.2.1 Interrupt Handling

Upon interrupt initiation, registers R8-R12 are automatically pushed to the system stack. These registers are pushed regardless of the priority level of the pending interrupt. The return address and status register are also automatically pushed to stack. The interrupt handler can therefore use R8-R12 freely. Upon interrupt completion, the old R8-R12 registers and status register are restored, and execution continues at the return address stored popped from stack.

The stack is also used to store the status register and return address for exceptions and *scall*. Executing the *rete* or *rets* instruction at the completion of an exception or system call will pop this status register and continue execution at the popped return address.

4.3.2.2 Java Support

AVR32UC does not provide Java hardware acceleration.

4.3.2.3 Floating Point Support

A fused multiply-accumulate Floating Point Unit (FPU), performaing a multiply and accumulate as a single operation with no intermediate rounding, therby increasing precision is provided. The floating point hardware conforms to the requirements of the C standard, which is based on the IEEE 754 floating point standard.

4.3.2.4 Memory Protection

The MPU allows the user to check all memory accesses for privilege violations. If an access is attempted to an illegal memory address, the access is aborted and an exception is taken. The MPU in AVR32UC is specified in the AVR32UC Technical Reference manual.

4.3.2.5 *Unaligned Reference Handling*

AVR32UC does not support unaligned accesses, except for doubleword accesses. AVR32UC is able to perform word-aligned *st.d* and *ld.d*. Any other unaligned memory access will cause an address exception. Doubleword-sized accesses with word-aligned pointers will automatically be performed as two word-sized accesses.

The following table shows the instructions with support for unaligned addresses. All other instructions require aligned addresses.

Table 4-1. Instructions with Unaligned Reference Support

Instruction	Supported Alignment
ld.d	Word
st.d	Word

4.3.2.6 *Unimplemented Instructions*

The following instructions are unimplemented in AVR32UC, and will cause an Unimplemented Instruction Exception if executed:

- All SIMD instructions
- All coprocessor instructions if no coprocessors are present
- retj, incjosp, popjc, pushjc
- tlbr, tlbs, tlbw
- cache

4.3.2.7 *CPU and Architecture Revision*

Three major revisions of the AVR32UC CPU currently exist. The device described in this datasheet uses CPU revision 3.

The Architecture Revision field in the CONFIG0 system register identifies which architecture revision is implemented in a specific device.

AVR32UC CPU revision 3 is fully backward-compatible with revisions 1 and 2, ie. code compiled for revision 1 or 2 is binary-compatible with revision 3 CPUs.

4.4 Programming Model

4.4.1 Register File Configuration

The AVR32UC register file is shown below.

Figure 4-3. The AVR32UC Register File

Application	Supervisor	INT0	INT1	INT2	INT3	Exception	NMI	Secure
Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0
PC	PC	PC	PC	PC	PC	PC	PC	PC
LR	LR	LR	LR	LR	LR	LR	LR	LR
SP_APP	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SEC
R12	R12	R12	R12	R12	R12	R12	R12	R12
R11	R11	R11	R11	R11	R11	R11	R11	R11
R10	R10	R10	R10	R10	R10	R10	R10	R10
R9	R9	R9	R9	R9	R9	R9	R9	R9
R8	R8	R8	R8	R8	R8	R8	R8	R8
R7	R7	R7	R7	R7	R7	R7	R7	R7
R6	R6	R6	R6	R6	R6	R6	R6	R6
R5	R5	R5	R5	R5	R5	R5	R5	R5
R4	R4	R4	R4	R4	R4	R4	R4	R4
R3	R3	R3	R3	R3	R3	R3	R3	R3
R2	R2	R2	R2	R2	R2	R2	R2	R2
R1	R1	R1	R1	R1	R1	R1	R1	R1
R0	R0	R0	R0	R0	R0	R0	R0	R0
SR	SR	SR	SR	SR	SR	SR	SR	SR

SS_STATUS
SS_ADRF
SS_ADRR
SS_ADR0
SS_ADR1
SS_SP_SYS
SS_SP_APP
SS_RAR
SS_RSR

4.4.2 Status Register Configuration

The Status Register (SR) is split into two halfwords, one upper and one lower, see Figure 4-4 and Figure 4-5. The lower word contains the C, Z, N, V, and Q condition code flags and the R, T, and L bits, while the upper halfword contains information about the mode and state the processor executes in. Refer to the *AVR32 Architecture Manual* for details.

Figure 4-4. The Status Register High Halfword

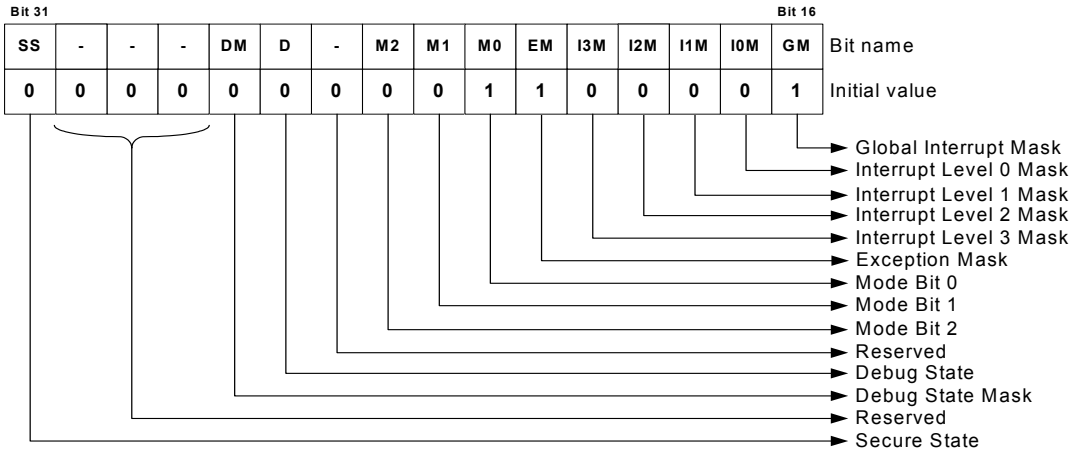
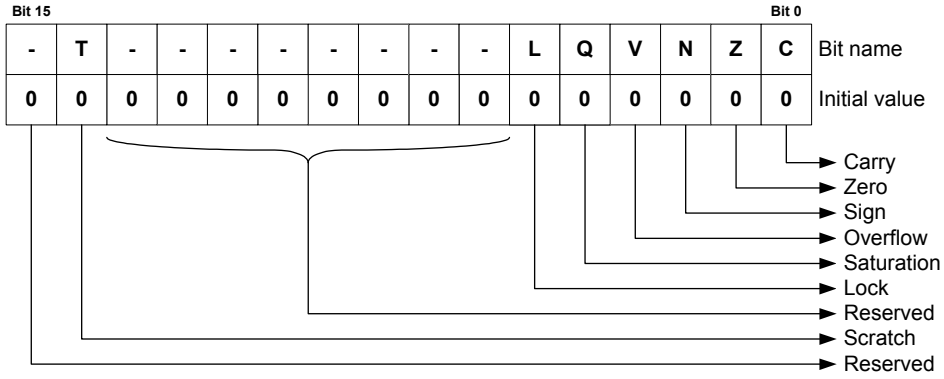


Figure 4-5. The Status Register Low Halfword



4.4.3 Processor States

4.4.3.1 Normal RISC State

The AVR32 processor supports several different execution contexts as shown in Table 4-2.

Table 4-2. Overview of Execution Modes, their Priorities and Privilege Levels.

Priority	Mode	Security	Description
1	Non Maskable Interrupt	Privileged	Non Maskable high priority interrupt mode
2	Exception	Privileged	Execute exceptions
3	Interrupt 3	Privileged	General purpose interrupt mode
4	Interrupt 2	Privileged	General purpose interrupt mode
5	Interrupt 1	Privileged	General purpose interrupt mode
6	Interrupt 0	Privileged	General purpose interrupt mode
N/A	Supervisor	Privileged	Runs supervisor calls
N/A	Application	Unprivileged	Normal program execution mode

Mode changes can be made under software control, or can be caused by external interrupts or exception processing. A mode can be interrupted by a higher priority mode, but never by one with lower priority. Nested exceptions can be supported with a minimal software overhead.

When running an operating system on the AVR32, user processes will typically execute in the application mode. The programs executed in this mode are restricted from executing certain instructions. Furthermore, most system registers together with the upper halfword of the status register cannot be accessed. Protected memory areas are also not available. All other operating modes are privileged and are collectively called System Modes. They have full access to all privileged and unprivileged resources. After a reset, the processor will be in supervisor mode.

4.4.3.2 Debug State

The AVR32 can be set in a debug state, which allows implementation of software monitor routines that can read out and alter system information for use during application development. This implies that all system and application registers, including the status registers and program counters, are accessible in debug state. The privileged instructions are also available.

All interrupt levels are by default disabled when debug state is entered, but they can individually be switched on by the monitor routine by clearing the respective mask bit in the status register.

Debug state can be entered as described in the *AVR32UC Technical Reference Manual*.

Debug state is exited by the *retd* instruction.

4.4.3.3 Secure State

The AVR32 can be set in a secure state, that allows a part of the code to execute in a state with higher security levels. The rest of the code can not access resources reserved for this secure code. Secure State is used to implement FlashVault technology. Refer to the *AVR32UC Technical Reference Manual* for details.

4.4.4 System Registers

The system registers are placed outside of the virtual memory space, and are only accessible using the privileged *mfsr* and *mtsr* instructions. The table below lists the system registers specified in the AVR32 architecture, some of which are unused in AVR32UC. The programmer is responsible for maintaining correct sequencing of any instructions following a *mtsr* instruction. For detail on the system registers, refer to the *AVR32UC Technical Reference Manual*.

Table 4-3. System Registers

Reg #	Address	Name	Function
0	0	SR	Status Register
1	4	EVBA	Exception Vector Base Address
2	8	ACBA	Application Call Base Address
3	12	CPUCR	CPU Control Register
4	16	ECR	Exception Cause Register
5	20	RSR_SUP	Unused in AVR32UC
6	24	RSR_INT0	Unused in AVR32UC
7	28	RSR_INT1	Unused in AVR32UC
8	32	RSR_INT2	Unused in AVR32UC
9	36	RSR_INT3	Unused in AVR32UC
10	40	RSR_EX	Unused in AVR32UC
11	44	RSR_NMI	Unused in AVR32UC
12	48	RSR_DBG	Return Status Register for Debug mode
13	52	RAR_SUP	Unused in AVR32UC
14	56	RAR_INT0	Unused in AVR32UC
15	60	RAR_INT1	Unused in AVR32UC
16	64	RAR_INT2	Unused in AVR32UC
17	68	RAR_INT3	Unused in AVR32UC
18	72	RAR_EX	Unused in AVR32UC
19	76	RAR_NMI	Unused in AVR32UC
20	80	RAR_DBG	Return Address Register for Debug mode
21	84	JECR	Unused in AVR32UC
22	88	JOSP	Unused in AVR32UC
23	92	JAVA_LV0	Unused in AVR32UC

Table 4-3. System Registers (Continued)

Reg #	Address	Name	Function
24	96	JAVA_LV1	Unused in AVR32UC
25	100	JAVA_LV2	Unused in AVR32UC
26	104	JAVA_LV3	Unused in AVR32UC
27	108	JAVA_LV4	Unused in AVR32UC
28	112	JAVA_LV5	Unused in AVR32UC
29	116	JAVA_LV6	Unused in AVR32UC
30	120	JAVA_LV7	Unused in AVR32UC
31	124	JTBA	Unused in AVR32UC
32	128	JBCR	Unused in AVR32UC
33-63	132-252	Reserved	Reserved for future use
64	256	CONFIG0	Configuration register 0
65	260	CONFIG1	Configuration register 1
66	264	COUNT	Cycle Counter register
67	268	COMPARE	Compare register
68	272	TLBEHI	Unused in AVR32UC
69	276	TLBELO	Unused in AVR32UC
70	280	PTBR	Unused in AVR32UC
71	284	TLBEAR	Unused in AVR32UC
72	288	MMUCR	Unused in AVR32UC
73	292	TLBARLO	Unused in AVR32UC
74	296	TLBARHI	Unused in AVR32UC
75	300	PCCNT	Unused in AVR32UC
76	304	PCNT0	Unused in AVR32UC
77	308	PCNT1	Unused in AVR32UC
78	312	PCCR	Unused in AVR32UC
79	316	BEAR	Bus Error Address Register
80	320	MPUAR0	MPU Address Register region 0
81	324	MPUAR1	MPU Address Register region 1
82	328	MPUAR2	MPU Address Register region 2
83	332	MPUAR3	MPU Address Register region 3
84	336	MPUAR4	MPU Address Register region 4
85	340	MPUAR5	MPU Address Register region 5
86	344	MPUAR6	MPU Address Register region 6
87	348	MPUAR7	MPU Address Register region 7
88	352	MPUPSR0	MPU Privilege Select Register region 0
89	356	MPUPSR1	MPU Privilege Select Register region 1

Table 4-3. System Registers (Continued)

Reg #	Address	Name	Function
90	360	MPUPSR2	MPU Privilege Select Register region 2
91	364	MPUPSR3	MPU Privilege Select Register region 3
92	368	MPUPSR4	MPU Privilege Select Register region 4
93	372	MPUPSR5	MPU Privilege Select Register region 5
94	376	MPUPSR6	MPU Privilege Select Register region 6
95	380	MPUPSR7	MPU Privilege Select Register region 7
96	384	MPUCRA	Unused in this version of AVR32UC
97	388	MPUCRB	Unused in this version of AVR32UC
98	392	MPUBRA	Unused in this version of AVR32UC
99	396	MPUBRB	Unused in this version of AVR32UC
100	400	MPUAPRA	MPU Access Permission Register A
101	404	MPUAPRB	MPU Access Permission Register B
102	408	MPUCR	MPU Control Register
103	412	SS_STATUS	Secure State Status Register
104	416	SS_ADRF	Secure State Address Flash Register
105	420	SS_ADRR	Secure State Address RAM Register
106	424	SS_ADR0	Secure State Address 0 Register
107	428	SS_ADR1	Secure State Address 1 Register
108	432	SS_SP_SYS	Secure State Stack Pointer System Register
109	436	SS_SP_APP	Secure State Stack Pointer Application Register
110	440	SS_RAR	Secure State Return Address Register
111	444	SS_RSR	Secure State Return Status Register
112-191	448-764	Reserved	Reserved for future use
192-255	768-1020	IMPL	IMPLEMENTATION DEFINED

4.5 Exceptions and Interrupts

In the AVR32 architecture, events are used as a common term for exceptions and interrupts. AVR32UC incorporates a powerful event handling scheme. The different event sources, like Illegal Op-code and interrupt requests, have different priority levels, ensuring a well-defined behavior when multiple events are received simultaneously. Additionally, pending events of a higher priority class may preempt handling of ongoing events of a lower priority class.

When an event occurs, the execution of the instruction stream is halted, and execution is passed to an event handler at an address specified in Table 4-4 on page 38. Most of the handlers are placed sequentially in the code space starting at the address specified by EVBA, with four bytes between each handler. This gives ample space for a jump instruction to be placed there, jumping to the event routine itself. A few critical handlers have larger spacing between them, allowing the entire event routine to be placed directly at the address specified by the EVBA-relative offset generated by hardware. All interrupt sources have autovectored interrupt service routine (ISR) addresses. This allows the interrupt controller to directly specify the ISR address as an address

relative to EVBA. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes. The target address of the event handler is calculated as (EVBA | event_handler_offset), not (EVBA + event_handler_offset), so EVBA and exception code segments must be set up appropriately. The same mechanisms are used to service all different types of events, including interrupt requests, yielding a uniform event handling scheme.

An interrupt controller does the priority handling of the interrupts and provides the autovector offset to the CPU.

4.5.1 System Stack Issues

Event handling in AVR32UC uses the system stack pointed to by the system stack pointer, SP_SYS, for pushing and popping R8-R12, LR, status register, and return address. Since event code may be timing-critical, SP_SYS should point to memory addresses in the IRAM section, since the timing of accesses to this memory section is both fast and deterministic.

The user must also make sure that the system stack is large enough so that any event is able to push the required registers to stack. If the system stack is full, and an event occurs, the system will enter an UNDEFINED state.

4.5.2 Exceptions and Interrupt Requests

When an event other than *scall* or debug request is received by the core, the following actions are performed atomically:

1. The pending event will not be accepted if it is masked. The I3M, I2M, I1M, I0M, EM, and GM bits in the Status Register are used to mask different events. Not all events can be masked. A few critical events (NMI, Unrecoverable Exception, TLB Multiple Hit, and Bus Error) can not be masked. When an event is accepted, hardware automatically sets the mask bits corresponding to all sources with equal or lower priority. This inhibits acceptance of other events of the same or lower priority, except for the critical events listed above. Software may choose to clear some or all of these bits after saving the necessary state if other priority schemes are desired. It is the event source's responsability to ensure that their events are left pending until accepted by the CPU.
2. When a request is accepted, the Status Register and Program Counter of the current context is stored to the system stack. If the event is an INT0, INT1, INT2, or INT3, registers R8-R12 and LR are also automatically stored to stack. Storing the Status Register ensures that the core is returned to the previous execution mode when the current event handling is completed. When exceptions occur, both the EM and GM bits are set, and the application may manually enable nested exceptions if desired by clearing the appropriate bit. Each exception handler has a dedicated handler address, and this address uniquely identifies the exception source.
3. The Mode bits are set to reflect the priority of the accepted event, and the correct register file bank is selected. The address of the event handler, as shown in Table 4-4 on page 38, is loaded into the Program Counter.

The execution of the event handler routine then continues from the effective address calculated.

The *rete* instruction signals the end of the event. When encountered, the Return Status Register and Return Address Register are popped from the system stack and restored to the Status Register and Program Counter. If the *rete* instruction returns from INT0, INT1, INT2, or INT3, registers R8-R12 and LR are also popped from the system stack. The restored Status Register contains information allowing the core to resume operation in the previous execution mode. This concludes the event handling.

4.5.3 Supervisor Calls

The AVR32 instruction set provides a supervisor mode call instruction. The *scall* instruction is designed so that privileged routines can be called from any context. This facilitates sharing of code between different execution modes. The *scall* mechanism is designed so that a minimal execution cycle overhead is experienced when performing supervisor routine calls from time-critical event handlers.

The *scall* instruction behaves differently depending on which mode it is called from. The behaviour is detailed in the instruction set reference. In order to allow the *scall* routine to return to the correct context, a return from supervisor call instruction, *rets*, is implemented. In the AVR32UC CPU, *scall* and *rets* uses the system stack to store the return address and the status register.

4.5.4 Debug Requests

The AVR32 architecture defines a dedicated Debug mode. When a debug request is received by the core, Debug mode is entered. Entry into Debug mode can be masked by the DM bit in the status register. Upon entry into Debug mode, hardware sets the SR.D bit and jumps to the Debug Exception handler. By default, Debug mode executes in the exception context, but with dedicated Return Address Register and Return Status Register. These dedicated registers remove the need for storing this data to the system stack, thereby improving debuggability. The Mode bits in the Status Register can freely be manipulated in Debug mode, to observe registers in all contexts, while retaining full privileges.

Debug mode is exited by executing the *retd* instruction. This returns to the previous context.

4.5.5 Entry Points for Events

Several different event handler entry points exist. In AVR32UC, the reset address is 0x80000000. This places the reset address in the boot flash memory area.

TLB miss exceptions and *scall* have a dedicated space relative to EVBA where their event handler can be placed. This speeds up execution by removing the need for a jump instruction placed at the program address jumped to by the event hardware. All other exceptions have a dedicated event routine entry point located relative to EVBA. The handler routine address identifies the exception source directly.

AVR32UC uses the ITLB and DTLB protection exceptions to signal a MPU protection violation. ITLB and DTLB miss exceptions are used to signal that an access address did not map to any of the entries in the MPU. TLB multiple hit exception indicates that an access address did map to multiple TLB entries, signalling an error.

All interrupt requests have entry points located at an offset relative to EVBA. This autovector offset is specified by an interrupt controller. The programmer must make sure that none of the autovector offsets interfere with the placement of other code. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes.

Special considerations should be made when loading EVBA with a pointer. Due to security considerations, the event handlers should be located in non-writeable flash memory, or optionally in a privileged memory protection region if an MPU is present.

If several events occur on the same instruction, they are handled in a prioritized way. The priority ordering is presented in Table 4-4 on page 38. If events occur on several instructions at different locations in the pipeline, the events on the oldest instruction are always handled before any events on any younger instruction, even if the younger instruction has events of higher priority

than the oldest instruction. An instruction B is younger than an instruction A if it was sent down the pipeline later than A.

The addresses and priority of simultaneous events are shown in Table 4-4 on page 38. Some of the exceptions are unused in AVR32UC since it has no MMU, coprocessor interface, or floating-point unit.

Table 4-4. Priority and Handler Addresses for Events

Priority	Handler Address	Name	Event source	Stored Return Address
1	0x80000000	Reset	External input	Undefined
2	Provided by OCD system	OCD Stop CPU	OCD system	First non-completed instruction
3	EVBA+0x00	Unrecoverable exception	Internal	PC of offending instruction
4	EVBA+0x04	TLB multiple hit	MPU	PC of offending instruction
5	EVBA+0x08	Bus error data fetch	Data bus	First non-completed instruction
6	EVBA+0x0C	Bus error instruction fetch	Data bus	First non-completed instruction
7	EVBA+0x10	NMI	External input	First non-completed instruction
8	Autovectored	Interrupt 3 request	External input	First non-completed instruction
9	Autovectored	Interrupt 2 request	External input	First non-completed instruction
10	Autovectored	Interrupt 1 request	External input	First non-completed instruction
11	Autovectored	Interrupt 0 request	External input	First non-completed instruction
12	EVBA+0x14	Instruction Address	CPU	PC of offending instruction
13	EVBA+0x50	ITLB Miss	MPU	PC of offending instruction
14	EVBA+0x18	ITLB Protection	MPU	PC of offending instruction
15	EVBA+0x1C	Breakpoint	OCD system	First non-completed instruction
16	EVBA+0x20	Illegal Opcode	Instruction	PC of offending instruction
17	EVBA+0x24	Unimplemented instruction	Instruction	PC of offending instruction
18	EVBA+0x28	Privilege violation	Instruction	PC of offending instruction
19	EVBA+0x2C	Floating-point	UNUSED	
20	EVBA+0x30	Coprocessor absent	Instruction	PC of offending instruction
21	EVBA+0x100	Supervisor call	Instruction	PC(Supervisor Call) +2
22	EVBA+0x34	Data Address (Read)	CPU	PC of offending instruction
23	EVBA+0x38	Data Address (Write)	CPU	PC of offending instruction
24	EVBA+0x60	DTLB Miss (Read)	MPU	PC of offending instruction
25	EVBA+0x70	DTLB Miss (Write)	MPU	PC of offending instruction
26	EVBA+0x3C	DTLB Protection (Read)	MPU	PC of offending instruction
27	EVBA+0x40	DTLB Protection (Write)	MPU	PC of offending instruction
28	EVBA+0x44	DTLB Modified	UNUSED	

5. Memories

5.1 Embedded Memories

- Internal High-Speed Flash (See Table 5-1 on page 40)
  - 512 Kbytes
  - 256 Kbytes
  - 128 Kbytes
  - 64 Kbytes
    - 0 Wait State Access at up to 33 MHz in Worst Case Conditions
    - 1 Wait State Access at up to 66 MHz in Worst Case Conditions
    - Pipelined Flash Architecture, allowing burst reads from sequential Flash locations, hiding penalty of 1 wait state access
    - Pipelined Flash Architecture typically reduces the cycle penalty of 1 wait state operation to only 15% compared to 0 wait state operation
    - 100 000 Write Cycles, 15-year Data Retention Capability
    - Sector Lock Capabilities, Bootloader Protection, Security Bit
    - 32 Fuses, Erased During Chip Erase
    - User Page For Data To Be Preserved During Chip Erase
- Internal High-Speed SRAM, Single-cycle access at full speed (See Table 5-1 on page 40)
  - 64 Kbytes
  - 32 Kbytes
  - 16 Kbytes
- Supplementary Internal High-Speed System SRAM (HSB RAM), Single-cycle access at full speed
  - Memory space available on System Bus for peripherals data.
  - 4 Kbytes

5.2 Physical Memory Map

The system bus is implemented as a bus matrix. All system bus addresses are fixed, and they are never remapped in any way, not even in boot. Note that AVR32UC CPU uses unsegmented translation, as described in the AVR32 Architecture Manual. The 32-bit physical address space is mapped as follows:

Table 5-1. AT32UC3C Physical Memory Map

Device	Start Address	AT32UC3 Derivatives							
		C0512C	C1512C C2512C	C0256C	C1256C C2256C	C0128C	C1128C C2128C	C064C	C164C C264C
Embedded SRAM	0x0000_0000	64 KB	64 KB	64 KB	64 KB	32 KB	32 KB	16 KB	16 KB
Embedded Flash	0x8000_0000	512 KB	512 KB	256 KB	256 KB	128 KB	128 KB	64 KB	64 KB
SAU	0x9000_0000	1 KB	1 KB	1 KB	1 KB	1 KB	1 KB	1 KB	1 KB
HSB SRAM	0xA000_0000	4 KB	4 KB	4 KB	4 KB	4 KB	4 KB	4 KB	4 KB
EBI SRAM CS0	0xC000_0000	16 MB	-	16 MB	-	16 MB	-	16 MB	-
EBI SRAM CS2	0xC800_0000	16 MB	-	16 MB	-	16 MB	-	16 MB	-
EBI SRAM CS3	0xCC00_0000	16 MB	-	16 MB	-	16 MB	-	16 MB	-
EBI SRAM /SDRAM CS1	0xD000_0000	128 MB	-	128 MB	-	128 MB	-	128 MB	-
HSB-PB Bridge C	0xFFFD_0000	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB
HSB-PB Bridge B	0xFFFE_0000	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB
HSB-PB Bridge A	0xFFFF_0000	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB

Table 5-2. Flash Memory Parameters

Part Number	Flash Size (FLASH_PW)	Number of pages (FLASH_P)	Page size (FLASH_W)
AT32UC3C0512C AT32UC3C1512C AT32UC3C2512C	512 Kbytes	1024	128 words
AT32UC3C0256C AT32UC3C1256C AT32UC3C2256C	256 Kbytes	512	128 words
AT32UC3C0128C AT32UC3C1128C AT32UC3C2128C	128 Kbytes	256	128 words
AT32UC3C064C AT32UC3C164C AT32UC3C264C	64 Kbytes	128	128 words

5.3 Peripheral Address Map

Table 5-3. Peripheral Address Mapping

Address		Peripheral Name
0xFFFD0000	PDCA	Peripheral DMA Controller - PDCA
0xFFFD1000	MDMA	Memory DMA - MDMA
0xFFFD1400	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter - USART1
0xFFFD1800	SPI0	Serial Peripheral Interface - SPI0
0xFFFD1C00	CANIF	Control Area Network interface - CANIF
0xFFFD2000	TC0	Timer/Counter - TC0
0xFFFD2400	ADCIFA	ADC controller interface with Touch Screen functionality - ADCIFA
0xFFFD2800	USART4	Universal Synchronous/Asynchronous Receiver/Transmitter - USART4
0xFFFD2C00	TWIM2	Two-wire Master Interface - TWIM2
0xFFFD3000	TWIS2	Two-wire Slave Interface - TWIS2

Table 5-3. Peripheral Address Mapping

0xFFFE0000	HFLASHC	Flash Controller - HFLASHC
0xFFFE1000	USBC	USB 2.0 OTG Interface - USBC
0xFFFE2000	HMATRIX	HSB Matrix - HMATRIX
0xFFFE2400	SAU	Secure Access Unit - SAU
0xFFFE2800	SMC	Static Memory Controller - SMC
0xFFFE2C00	SDRAMC	SDRAM Controller - SDRAMC
0xFFFE3000	MACB	Ethernet MAC - MACB
0xFFFF0000	INTC	Interrupt controller - INTC
0xFFFF0400	PM	Power Manager - PM
0xFFFF0800	SCIF	System Control Interface - SCIF
0xFFFF0C00	AST	Asynchronous Timer - AST
0xFFFF1000	WDT	Watchdog Timer - WDT
0xFFFF1400	EIC	External Interrupt Controller - EIC
0xFFFF1800	FREQM	Frequency Meter - FREQM
0xFFFF2000	GPIO	General Purpose Input/Output Controller - GPIO
0xFFFF2800	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter - USART0
0xFFFF2C00	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter - USART2
0xFFFF3000	USART3	Universal Synchronous/Asynchronous Receiver/Transmitter - USART3
0xFFFF3400	SPI1	Serial Peripheral Interface - SPI1

Table 5-3. Peripheral Address Mapping

0xFFFF3800	TWIM0	Two-wire Master Interface - TWIM0
0xFFFF3C00	TWIM1	Two-wire Master Interface - TWIM1
0xFFFF4000	TWIS0	Two-wire Slave Interface - TWIS0
0xFFFF4400	TWIS1	Two-wire Slave Interface - TWIS1
0xFFFF4800	IISC	Inter-IC Sound (I2S) Controller - IISC
0xFFFF4C00	PWM	Pulse Width Modulation Controller - PWM
0xFFFF5000	QDEC0	Quadrature Decoder - QDEC0
0xFFFF5400	QDEC1	Quadrature Decoder - QDEC1
0xFFFF5800	TC1	Timer/Counter - TC1
0xFFFF5C00	PEVC	Peripheral Event Controller - PEVC
0xFFFF6000	ACIFA0	Analog Comparators Interface - ACIFA0
0xFFFF6400	ACIFA1	Analog Comparators Interface - ACIFA1
0xFFFF6800	DACIFB0	DAC interface - DACIFB0
0xFFFF6C00	DACIFB1	DAC interface - DACIFB1
0xFFFF7000	AW	aWire - AW

5.4 CPU Local Bus Mapping

Some of the registers in the GPIO module are mapped onto the CPU local bus, in addition to being mapped on the Peripheral Bus. These registers can therefore be reached both by accesses on the Peripheral Bus, and by accesses on the local bus.

Mapping these registers on the local bus allows cycle-deterministic toggling of GPIO pins since the CPU and GPIO are the only modules connected to this bus. Also, since the local bus runs at CPU speed, one write or read operation can be performed per clock cycle to the local bus-mapped GPIO registers.

The following GPIO registers are mapped on the local bus:

Table 5-4. Local bus mapped GPIO registers

Port	Register	Mode	Local Bus Address	Access
A	Output Driver Enable Register (ODER)	WRITE	0x40000040	Write-only
		SET	0x40000044	Write-only
		CLEAR	0x40000048	Write-only
		TOGGLE	0x4000004C	Write-only
	Output Value Register (OVR)	WRITE	0x40000050	Write-only
		SET	0x40000054	Write-only
		CLEAR	0x40000058	Write-only
		TOGGLE	0x4000005C	Write-only
	Pin Value Register (PVR)	-	0x40000060	Read-only
B	Output Driver Enable Register (ODER)	WRITE	0x40000140	Write-only
		SET	0x40000144	Write-only
		CLEAR	0x40000148	Write-only
		TOGGLE	0x4000014C	Write-only
	Output Value Register (OVR)	WRITE	0x40000150	Write-only
		SET	0x40000154	Write-only
		CLEAR	0x40000158	Write-only
		TOGGLE	0x4000015C	Write-only
	Pin Value Register (PVR)	-	0x40000160	Read-only
C	Output Driver Enable Register (ODER)	WRITE	0x40000240	Write-only
		SET	0x40000244	Write-only
		CLEAR	0x40000248	Write-only
		TOGGLE	0x4000024C	Write-only
	Output Value Register (OVR)	WRITE	0x40000250	Write-only
		SET	0x40000254	Write-only
		CLEAR	0x40000258	Write-only
		TOGGLE	0x4000025C	Write-only
	Pin Value Register (PVR)	-	0x40000260	Read-only

Table 5-4. Local bus mapped GPIO registers

Port	Register	Mode	Local Bus Address	Access
D	Output Driver Enable Register (ODER)	WRITE	0x40000340	Write-only
		SET	0x40000344	Write-only
		CLEAR	0x40000348	Write-only
		TOGGLE	0x4000034C	Write-only
	Output Value Register (OVR)	WRITE	0x40000350	Write-only
		SET	0x40000354	Write-only
		CLEAR	0x40000358	Write-only
		TOGGLE	0x4000035C	Write-only
	Pin Value Register (PVR)	-	0x40000360	Read-only

6. Supply and Startup Considerations

6.1 Supply Considerations

6.1.1 Power Supplies

The AT32UC3C has several types of power supply pins:

- **VDDIO pins (VDDIO1, VDDIO2, VDDIO3): Power I/O lines. Two voltage ranges are available: 5V or 3.3V nominal. The VDDIO pins should be connected together.**
- **VDDANA: Powers the Analog part of the device (Analog I/Os, ADC, ACs, DACs). 2 voltage ranges available: 5V or 3.3V nominal.**
- **VDDIN_5: Input voltage for the 1.8V and 3.3V regulators. Two Voltage ranges are available: 5V or 3.3V nominal.**
- **VDDIN_33:**
  - **USB I/O power supply**
  - **if the device is 3.3V powered: Input voltage, voltage is 3.3V nominal.**
  - **if the device is 5V powered: stabilization for the 3.3V voltage regulator, requires external capacitors**
- **VDDCORE: Stabilization for the 1.8V voltage regulator, requires external capacitors.**
- **GNDCORE: Ground pins for the voltage regulators and the core.**
- **GNDANA: Ground pin for Analog part of the design**
- **GNDPLL: Ground pin for the PLLs**
- **GNDIO pins (GNDIO1, GNDIO2, GNDIO3): Ground pins for the I/O lines. The GNDIO pins should be connected together.**

See ”Electrical Characteristics” on page 1249 for power consumption on the various supply pins.

For decoupling recommendations for the different power supplies, please refer to the schematic checklist.

6.1.2 Voltage Regulators

The AT32UC3C embeds two voltage regulators:

- One 1.8V internal regulator that converts from VDDIN_5 to 1.8V. The regulator supplies the output voltage on VDDCORE.
- One 3.3V internal regulator that converts from VDDIN_5 to 3.3V. The regulator supplies the USB pads on VDDIN_33. If the USB is not used, the 3.3V regulator can be disabled through the VREG33CTL field of the VREGCTRL SCIF register.

6.1.3 Regulators Connection

The AT32UC3C supports two power supply configurations.

- 5V single supply mode
- 3.3V single supply mode

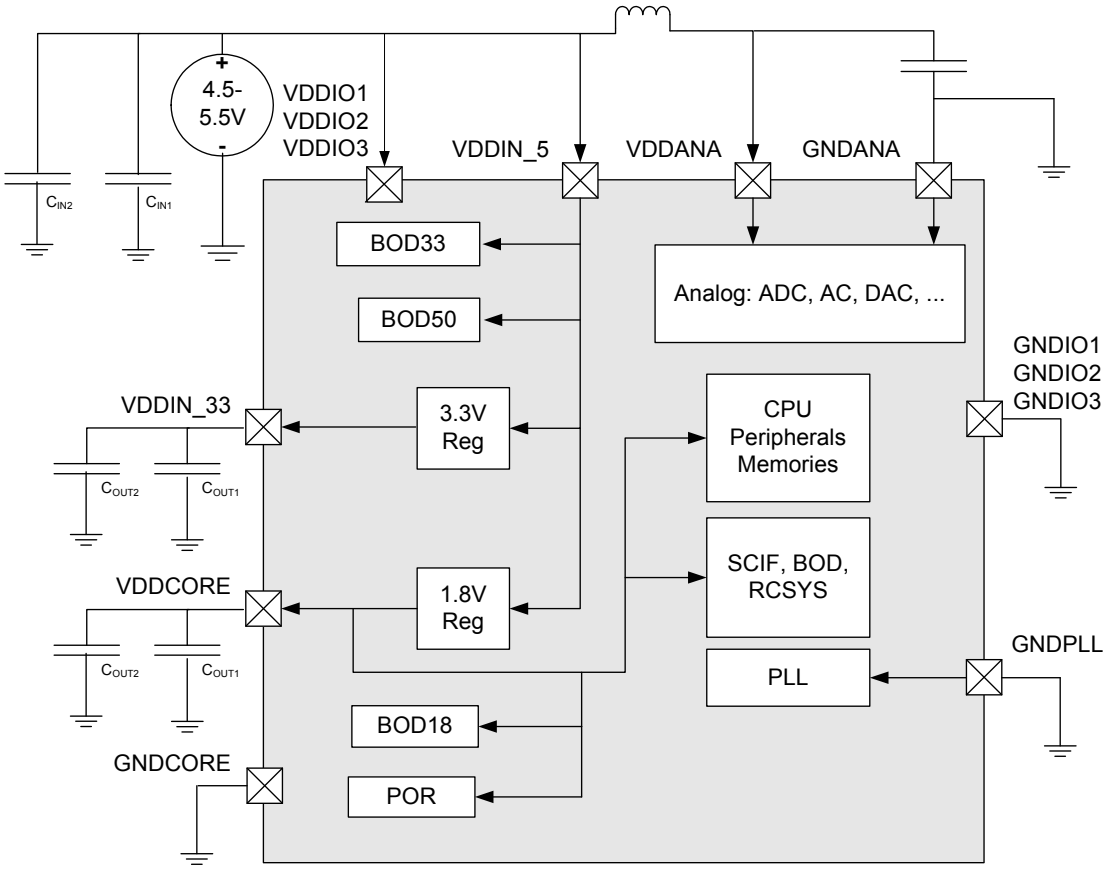
6.1.3.1 5V Single Supply Mode

In 5V single supply mode, the 1.8V internal regulator is connected to the 5V source (VDDIN_5 pin) and its output feeds VDDCORE.

The 3.3V regulator is connected to the 5V source (VDDIN_5 pin) and its output feeds the USB pads. If the USB is not used, the 3.3V regulator can be disabled through the VREG33CTL field of the VREGCTRL SCIF register.

Figure 6-1 on page 47 shows the power schematics to be used for 5V single supply mode. All I/O lines and analog blocks will be powered by the same power (VDDIN_5 = VDDIO1 = VDDIO2 = VDDIO3 = VDDANA).

Figure 6-1. 5V Single Power Supply mode



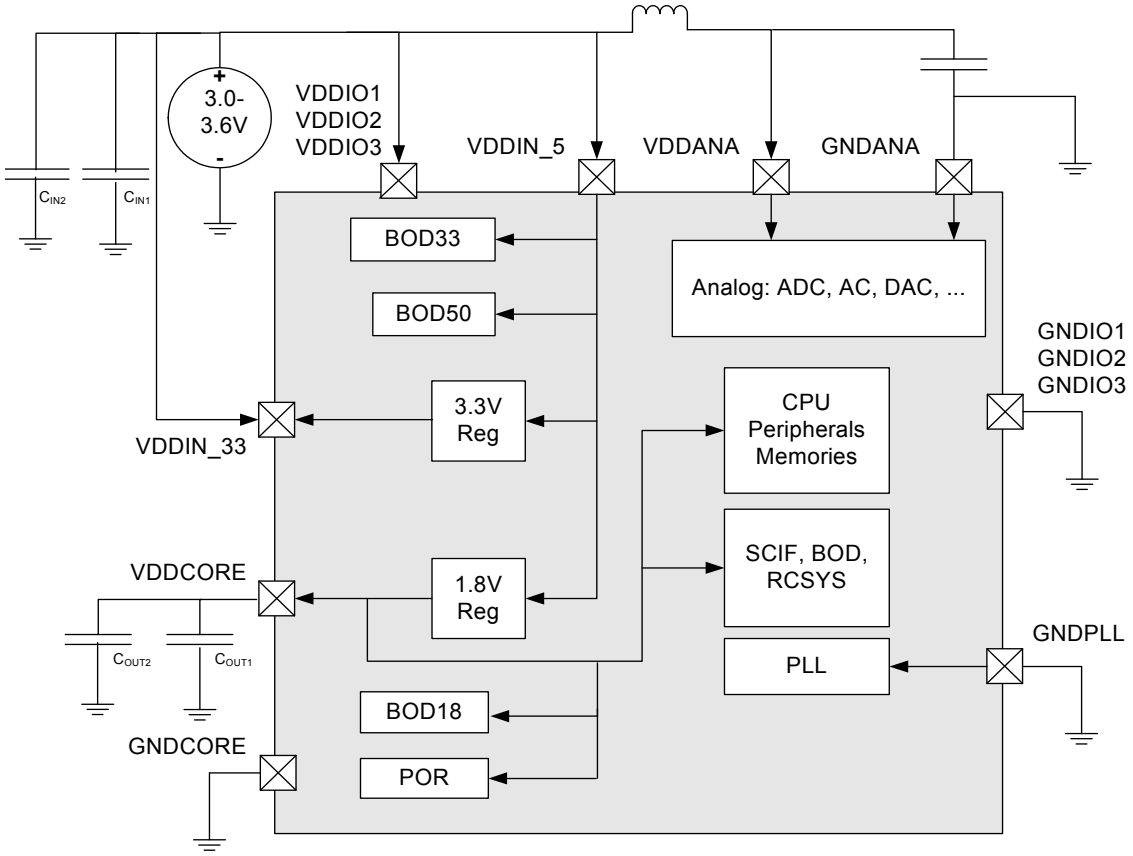
6.1.3.2 *3.3V Single Supply Mode*

In 3.3V single supply mode, the VDDIN_5 and VDDIN_33 pins should be connected together externally. The 1.8V internal regulator is connected to the 3.3V source (VDDIN_5 pin) and its output feeds VDDCORE.

The 3.3V regulator should be disabled once the circuit is running through the VREG33CTL field of the VREGCTRL SCIF register.

Figure 6-2 on page 48 shows the power schematics to be used for 3.3V single supply mode. All I/O lines and analog blocks will be powered by the same power (VDDIN_5 = VDDIN_33 = VDDIO1 = VDDIO2 = VDDIO3 = VDDANA).

Figure 6-2. 3 Single Power Supply Mode



6.1.4 Power-up Sequence

6.1.4.1 Maximum Rise Rate

To avoid risk of latch-up, the rise rate of the power supplies must not exceed the values described in Table 40-2 on page 1250 .

Recommended order for power supplies is also described in this table.

6.1.4.2 Minimum Rise Rate

The integrated Power-Reset circuitry monitoring the powering supply requires a minimum rise rate for the VDDIN_5 power supply.

See Table 40-2 on page 1250 for the minimum rise rate value.

If the application can not ensure that the minimum rise rate condition for the VDDIN power supply is met, the following configuration can be used:

- A logic “0” value is applied during power-up on pin RESET_N until:
  - VDDIN_5 rises above 4.5V in 5V single supply mode.
  - VDDIN_33 rises above 3V in 3.3V single supply mode.

6.2 Startup Considerations

This chapter summarizes the boot sequence of the AT32UC3C. The behavior after power-up is controlled by the Power Manager. For specific details, refer to the Power Manager chapter.

6.2.1 Starting of clocks

At power-up, the BOD33 and the BOD18 are enabled. The device will be held in a reset state by the power-up circuitry, until the VDDIN_33 (resp. VDDCORE) has reached the reset threshold of the BOD33 (resp BOD18). Refer to the Electrical Characteristics for the BOD thresholds. Once the power has stabilized, the device will use the System RC Oscillator (RCSYS, 115KHz typical frequency) as clock source. The BOD18 and BOD33 are kept enabled or are disabled according to the fuse settings (See the Fuse Setting section in the Flash Controller chapter).

On system start-up, the PLLs are disabled. All clocks to all modules are running. No clocks have a divided frequency, all parts of the system receive a clock with the same frequency as the internal RC Oscillator.

6.2.2 Fetching of initial instructions

After reset has been released, the AVR32UC CPU starts fetching instructions from the reset address, which is 0x8000_0000. This address points to the first address in the internal Flash.

The internal Flash uses VDDIO voltage during read and write operations. It is recommended to use the BOD33 to monitor this voltage and make sure the VDDIO is above the minimum level (3.0V).

The code read from the internal Flash is free to configure the system to use for example the PLLs, to divide the frequency of the clock routed to some of the peripherals, and to gate the clocks to unused peripherals.

7. Power Manager (PM)

Rev: 4.1.2.3

7.1 Features

- **Generates clocks and resets for digital logic**
- **On-the-fly frequency change of CPU, HSB and PBx clocks**
- **Sleep modes allow simple disabling of logic clocks and clock sources**
- **Module-level clock gating through maskable peripheral clocks**
- **Wake-up from internal or external interrupts**
- **Automatic identification of reset sources**

7.2 Overview

The Power Manager (PM) provides synchronous clocks used to clock the main digital logic in the device, namely the CPU, and the modules and peripherals connected to the High Speed Bus (HSB) and the Peripheral Buses (PBx).

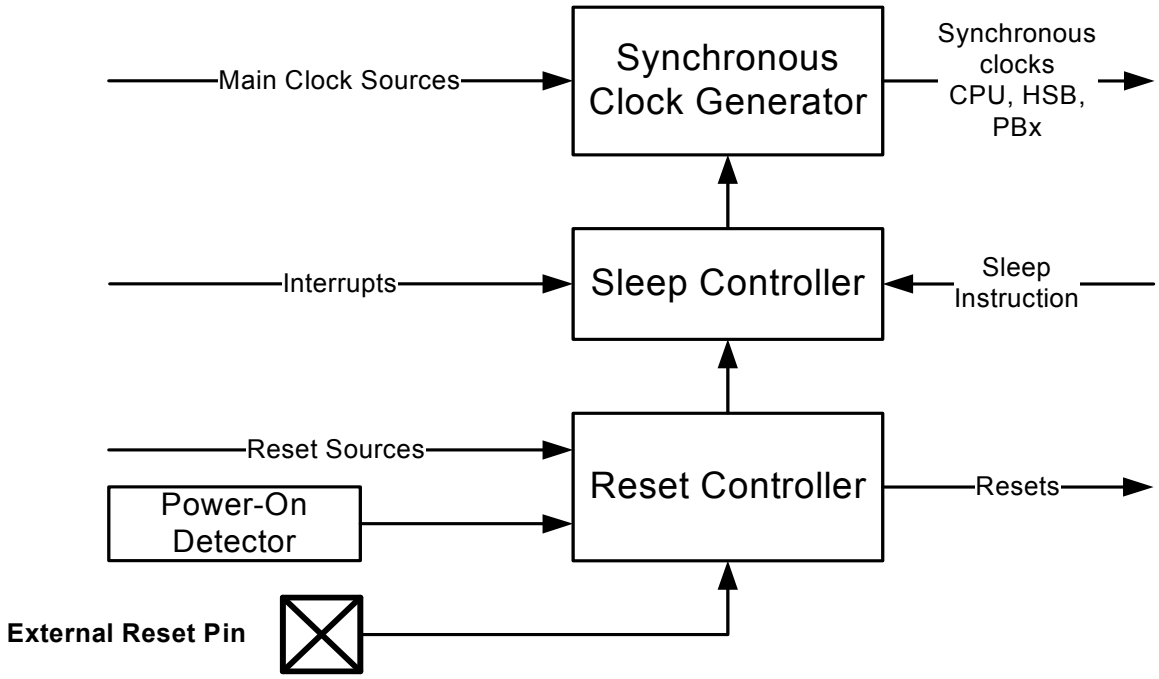
The PM also contains advanced power-saving features, allowing the user to optimize the power consumption for an application. The synchronous clocks are divided into a number of clock domains, one for the CPU and HSB and one for each PBx. The clocks can run at different speeds, so the user can save power by running peripherals at a relatively low clock, while maintaining a high CPU performance. Additionally, the clocks can be independently changed on-the-fly, without halting any peripherals. This enables the user to adjust the speed of the CPU and memories to the dynamic load of the application, without disturbing or re-configuring active peripherals.

Each module also has a separate clock, enabling the user to switch off the clock for inactive modules, to save further power. Additionally, clocks and oscillators can be automatically switched off during idle periods by using the sleep instruction on the CPU. The system will return to normal operation on occurrence of interrupts.

The Power Manager also contains a Reset Controller, which collects all possible reset sources, generates hard and soft resets, and allows the reset source to be identified by software.

7.3 Block Diagram

Figure 7-1. PM Block Diagram



7.4 I/O Lines Description

Table 7-1. I/O Lines Description

Name	Description	Type	Active Level
RESET_N	Reset	Input	Low

7.5 Product Dependencies

7.5.1 Interrupt

The PM interrupt line is connected to one of the internal sources of the interrupt controller. Using the PM interrupt requires the interrupt controller to be programmed first.

7.5.2 Clock Implementation

In AT32UC3C, the HSB shares the source clock with the CPU. This means that writing to the HSBSEL register has no effect. This register will always read the same value as CPUSEL.

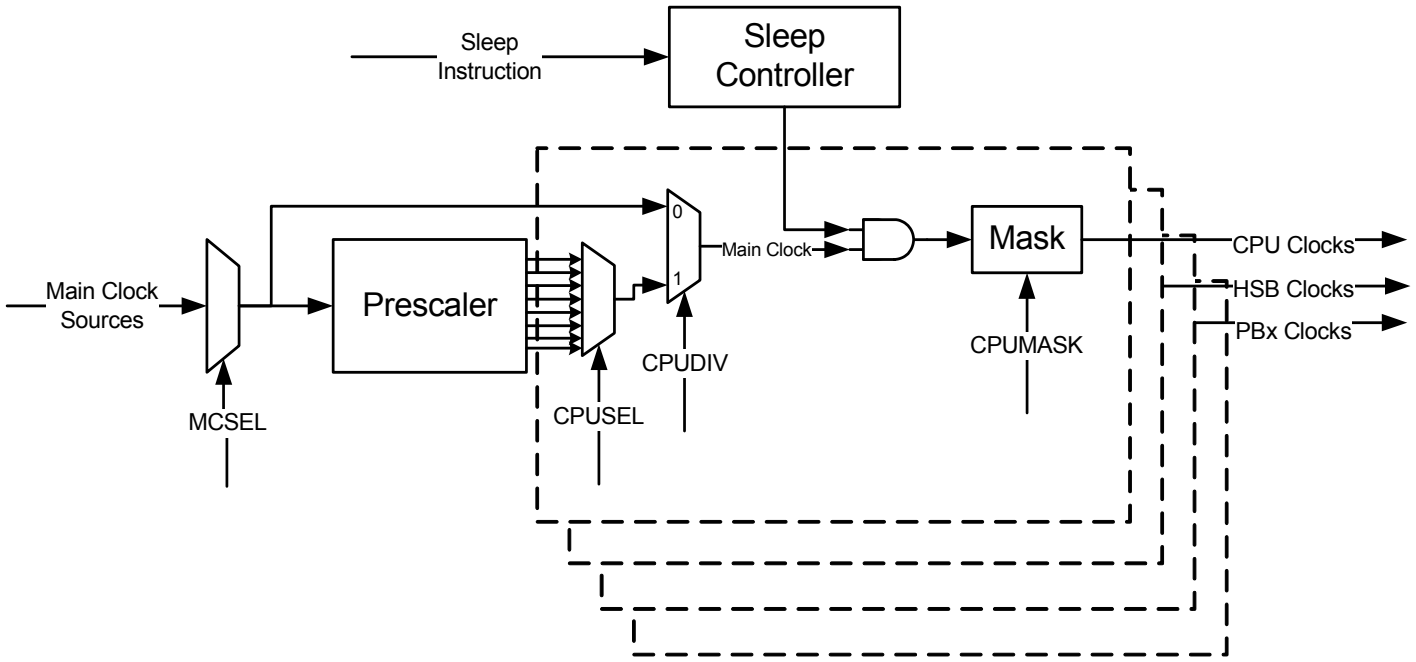
The clock for the PM bus interface (CLK_PM) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager, whoever if disabled it can only be re-enabled by a reset.

7.6 Functional Description

7.6.1 Synchronous Clocks

The System RC Oscillator (RCSYS) or a set of other clock sources provide the source for the main clock, which is the common root for the synchronous clocks for the CPU/HSB and PBx modules. For details about the other main clock sources, please refer to the register description of the Main Clock Control Register (MCCTRL). The main clock is divided by an 8-bit prescaler, and each of these synchronous clocks can run from any tapping of this prescaler, or the undivided main clock, as long as $f_{CPU} \geq f_{PBx,}$. The synchronous clock source can be changed on-the fly, responding to varying load in the application. The clock domains can be shut down in sleep mode, as described in Section 7.6.3. Additionally, the clocks for each module in each synchronous clock domain can be individually masked, to avoid power consumption in inactive modules.

Figure 7-2. Synchronous Clock Generation



7.6.1.1 *Selecting the main clock source*

The common main clock can be connected to RCSYS or a set of other clock sources. For details about the other main clock sources, please refer to the register description of the Main Clock Control Register (MCCTRL). By default, the main clock will be connected to RCSYS. The user can connect the main clock to an other source by writing the MCSEL field in the MCCTRL register. This must only be done after that unit has been enabled and is ready, otherwise a deadlock will occur. Care should also be taken that the new frequency of the synchronous clocks does not exceed the maximum frequency for each clock domain.

7.6.1.2 *Selecting synchronous clock division ratio*

The main clock feeds an 8-bit prescaler, which can be used to generate the synchronous clocks. By default, the synchronous clocks run on the undivided main clock. The user can select a prescaler division for the CPU clock by writing CPUDIV in CPUSEL register to one and CPUSEL in CPUSEL register to the value, resulting in a CPU clock frequency:

$$f_{CPU} = f_{main} / 2^{(CPUSEL+1)}$$

Similarly, the clock for the PBx can be divided by writing their respective registers. To ensure correct operation, frequencies must be selected so that $f_{CPU} \geq f_{PBx}$. Also, frequencies must never exceed the specified maximum frequency for each clock domain.

CPUSEL and PBxSEL can be written without halting or disabling peripheral modules. Writing CPUSEL and PBxSEL allows a new clock setting to be written to all synchronous clocks at the same time. It is possible to keep one or more clocks unchanged by writing a one to the registers. This way, it is possible to, e.g., scale CPU and HSB speed according to the required performance, while keeping the PBx frequency constant.

For modules connected to the HSB bus, the PB clock frequency must be set to the same frequency as the CPU clock.

7.6.1.3 Clock Ready flag

There is a slight delay from CPUSEL and PBxSEL is written and the new clock setting becomes effective. During this interval, the Clock Ready (CKRDY) flag in ISR will read as zero. If CKRDY in the IER register is written to one, the Power Manager interrupt can be triggered when the new clock setting is effective. CKSEL must not be re-written while CKRDY is zero, or the system may become unstable or hang.

7.6.2 Peripheral Clock Masking

By default, the clock for all modules are enabled, regardless of which modules are actually being used. It is possible to disable the clock for a module in the CPU, HSB or PBx clock domain by writing the corresponding bit in the Clock Mask register (CPU/HSB/PBx) to zero. When a module is not clocked, it will cease operation, and its registers cannot be read or written. The module can be re-enabled later by writing the corresponding mask bit to one.

A module may be connected to several clock domains, in which case it will have several mask bits.

The Maskable Module Clocks table contains a list of implemented maskable clocks.

7.6.2.1 Cautionary note

Note that clocks should only be switched off if it is certain that the module will not be used. Switching off the clock for the flash controller will cause a problem if the CPU needs to read from the flash. Switching off the clock to the Power Manager (PM), which contains the mask registers, or the corresponding PBx bridge, will make it impossible to write the mask registers again. In this case, they can only be re-enabled by a system reset.

7.6.3 Sleep Modes

In normal operation, all clock domains are active, allowing software execution and peripheral operation. When the CPU is idle, it is possible to switch off the CPU clock and optionally other clock domains to save power. This is activated by the sleep instruction, which takes the sleep mode index number from Table 7-2 on page 54 as argument.

7.6.3.1 Entering and exiting sleep modes

The sleep instruction will halt the CPU and all modules belonging to the stopped clock domains. The modules will be halted regardless of the bit settings of the mask registers.

Clock sources can also be switched off to save power. Some of these have a relatively long start-up time, and are only switched off when very low power consumption is required.

The CPU and affected modules are restarted when the sleep mode is exited. This occurs when an interrupt triggers. Note that even if an interrupt is enabled in sleep mode, it may not trigger if the source module is not clocked.

7.6.3.2 *Supported sleep modes*

The following sleep modes are supported. These are detailed in Table 7-2 on page 54.

- Idle: The CPU is stopped, the rest of the chip is operating.
- Frozen: The CPU and HSB modules are stopped, peripherals are operating.
- Standby: All synchronous clocks are stopped, but the clock sources are running, allowing quick wake-up to normal mode.
- Stop: As Standby, but oscillators, and other clock sources are stopped. 32KHz (if enabled), RC oscillators, AST and WDT will still operate.
- DeepStop: All synchronous clocks and clock sources are stopped. 32KHz oscillator can run if enabled. RC oscillator still operates. Bandgap voltage reference and BOD is turned off.
- Static: All clock sources, including RC oscillator are stopped. 32KHz oscillator can run if enabled. Bandgap voltage reference and BOD detector are turned off.

Table 7-2. Sleep Modes

Index[1]	Sleep Mode	CPU	HSB	PBx GCLK	Clock sources	Osc32	RCSYS	BOD & Bandgap	Voltage Regulator
0	**Idle**	Stop	Run	Run	Run	Run	Run	On	Normal mode
1	**Frozen**	Stop	Stop	Run	Run	Run	Run	On	Normal mode
2	**Standby**	Stop	Stop	Stop	Run	Run	Run	On	Normal mode
3	**Stop**	Stop	Stop	Stop	Stop	Run	Run	On	Low power mode
4	**DeepStop**	Stop	Stop	Stop	Stop	Run	Run	Off	Low power mode
5	**Static**	Stop	Stop	Stop	Stop	Run	Stop	Off	Low power mode

Note: 1. The sleep mode index corresponds to the argument to the sleep instruction.

The power level of the internal voltage regulator is also adjusted according to the sleep mode to reduce the internal regulator power consumption.

7.6.3.3 *Waking from sleep modes*

There are two types of wake up sources from sleep, synchronous and asynchronous. Synchronous wake up sources are any non-masked interrupt. Asynchronous wake up sources are AST, WDT, external interrupts from EIC, external resetor any asynchronous wake up enabled in the AWEN (Asynchronous Wake Up Enable) register. The valid wake up sources for each sleep mode is detailed in Section 7-3 on page 54.

Table 7-3. Wake up sources

Index[1]	Sleep Mode	Wake up Sources
0	**Idle**	Synchronous, Asynchronous
1	**Frozen**	Synchronous [1], Asynchronous
2	**Standby**	Asynchronous

Table 7-3. Wake up sources

Index[1]	Sleep Mode	Wake up Sources
3	**Stop**	Asynchronous
4	**DeepStop**	Asynchronous
5	**Static**	Asynchronous [2]

1. Only PB modules, as HSB modules have stopped clock.
2. WDT only if clocked from 32 KHz oscillator.

7.6.3.4 Precautions when entering sleep mode

Modules communicating with external circuits should normally be disabled before entering a sleep mode that will stop the module operation. This prevents erratic behavior when entering or exiting sleep mode. Please refer to the relevant module documentation for recommended actions.

Communication between the synchronous clock domains is disturbed when entering and exiting sleep modes. This means that bus transactions are not allowed between clock domains affected by the sleep mode. The system may hang if the bus clocks are stopped in the middle of a bus transaction.

The CPU is automatically stopped in a safe state to ensure that all CPU bus operations are complete when the sleep mode goes into effect. Thus, when entering Idle mode, no further action is necessary.

When entering a sleep mode (except Idle mode), all HSB masters must be stopped before entering the sleep mode. Also, if there is a chance that any PB write operations are incomplete, the CPU should perform a read operation from any register on the PB bus before executing the sleep instruction. This will stall the CPU while waiting for any pending PB operations to complete.

7.6.4 Divided PB Clocks

The clock generator in the Power Manager provides divided PBx clocks for use by peripherals that require a prescaled PBx clock. This is described in the documentation for the relevant modules.

The divided clocks are directly maskable, and are stopped in sleep modes where the PBx clocks are stopped.

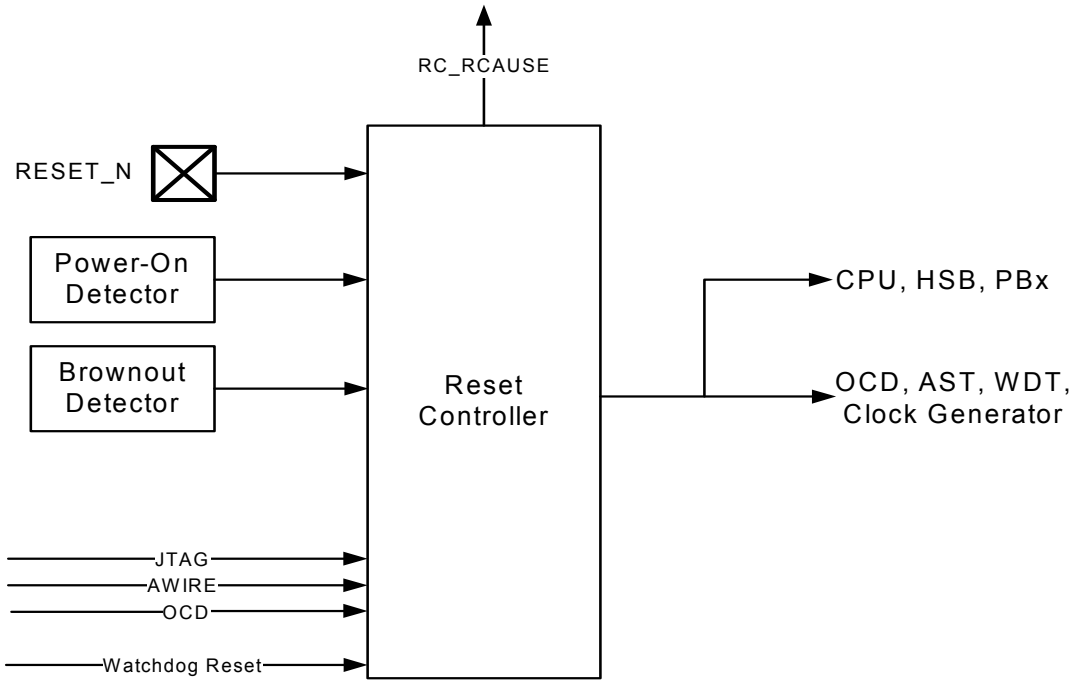
7.6.5 Reset Controller

The Reset Controller collects the various reset sources in the system and generates hard and soft resets for the digital logic.

The device contains a Power-On Detector, which keeps the system reset until power is stable. This eliminates the need for external reset circuitry to guarantee stable operation when powering up the device.

It is also possible to reset the device by asserting the RESET_N pin. This pin has an internal pull-up, and does not need to be driven externally when negated. Table 7-4 on page 56 lists these and other reset sources supported by the Reset Controller.

Figure 7-3. Reset Controller Block Diagram



In addition to the listed reset types, the JTAG & AWIRE can keep parts of the device statically reset. See JTAG and AWIRE documentation for details.

Table 7-4. Reset Description

Reset source	Description
Power-on Reset	Supply voltage below the power-on reset detector threshold voltage
External Reset	RESET_N pin asserted
Brownout Reset	Supply voltage on VDDCORE below the brownout reset detector threshold voltage
CPU Error	Caused by an illegal CPU access to external memory while in Supervisor mode
Watchdog Timer	See Watchdog Timer documentation.
OCD	See On-Chip Debug documentation

When a Reset occurs, some parts of the chip are not necessarily reset, depending on the reset source. Only the Power On Reset (POR) will force a reset of the whole chip. Refer to the module configuration chapter to know the effect of the different reset events.

The table located in the module configuration chapter lists parts of the device that are reset, depending on the reset source.The cause of the last reset can be read from the RCAUSE register. This register contains one bit for each reset source, and can be read during the boot sequence of an application to determine the proper action to be taken.

7.6.5.1 *Power-On Detector*

The Power-On Detector monitors the VDDCORE supply pin and generates a reset when the device is powered on. The reset is active until the supply voltage from the linear regulator is above the power-on threshold level. The reset will be re-activated if the voltage drops below the power-on threshold level. See Electrical Characteristics for parametric details.

7.6.5.2 *External Reset*

The external reset detector monitors the state of the RESET_N pin. By default, a low level on this pin will generate a reset.

7.6.6 Clock Failure Detector

This mechanism allows switching the main clock to the safe RCSYS clock, when the main clock source is considered off. This may happen when a external crystal is selected as the clock source of the main clock but the crystal is not mounted on the board. The mechanism is to detect, during a RCSYS period, at least one rising edge of the main clock. If no rising edge is seen the clock is considered failed.

Example:

* RCSYS = 115khz

=> Failure detected if the main clock is < 115 kHz

As soon as the detector is enabled, the clock failure detector will monitor the divided main clock. Note that the detector does not monitor if the RCSYS is the source of the main clock, or if the main clock is temporarily not available (startup-time after a wake-up, switching timing etc.), or in sleep mode where the main clock is driven by the RCSYS (Stop and DeepStop mode). When a clock failure is detected, the main clock automatically switches to the RCSYS clock and the CFD interrupt is generated if enabled.

The MCCTRL register that selects the source clock of the main clock is changed by hardware to indicate that the main clock comes from RCSYS.

7.6.7 Interrupts

The PM has a number of interrupts:

- AE: Access Error, set if a lock protected register is written without first being unlocked.
- CKRDY: Clock Ready, set when new CKSEL settings are effective.
- CFD: Clock Failure Detected, set if the system detects that the main clock is not running.

7.7 User Interface

Table 7-5. PM Register Memory Map

Offset	Register	Name	Access	Reset State
0x0000	Main Clock Control	MCCTRL	Read/Write	0x00000000
0x0004	CPU Clock Select	CPUSEL	Read/Write	0x00000000
0x0008	HSB Clock Select	HSBSEL	Read Only	0x00000000
0x000C	PBA Clock Select	PBASEL	Read/Write	0x00000000
0x00010	PBB Clock Select	PBBSEL	Read/Write	0x00000000
0x0014	PBC Clock Select	PBCSEL	Read/Write	0x00000000
0x0020	CPU Mask	CPUMASK	Read/Write	0x00000003
0x0024	HSB Mask	HSBMASK	Read/Write	0x00003FFF
0x0028	PBA Mask	PBAMASK	Read/Write	0x07FFFFFF
0x002C	PBB Mask	PBBMASK	Read/Write	0x0000007F
0x0030	PBC Mask	PBCMASK	Read/Write	0x000003FF
0x0040	PBA Divided Mask	PBADIVMASK	Read/Write	0x0000007F
0x0044	PBB Divided Mask	PBBDIVMASK	Read/Write	0x0000007F
0x0048	PBC Divided Mask	PBCDIVMASK	Read/Write	0x0000007F
0x0054	Clock Failure Detector Control	CFDCTRL	Read/Write	0x00000000
0x0058	Unlock Register	UNLOCK	Write Only	-
0x00C0	PM Interrupt Enable Register	IER	Write Only	0x00000000
0x00C4	PM Interrupt Disable Register	IDR	Write Only	0x00000000
0x00C8	PM Interrupt Mask Register	IMR	Read Only	0x00000000
0x00CC	PM Interrupt Status Register	ISR	Read Only	0x00000000
0x00D0	PM Interrupt Clear Register	ICR	Write Only	0x00000000
0x00D4	Status Register	SR	Read Only	0x00000000
0x0180	Reset Cause Register	RCAUSE	Read Only	Latest Reset Source
0x0184	Wake Cause Register	WCAUSE	Read Only	Latest Wake Source
0x0188	Asynchronous Wake Enable	AWEN	Read/Write	0x00000000
0x03F8	Configuration Register	CONFIG	Read Only	-(1)
0x03FC	Version Register	VERSION	Read Only	-(1)

Note: 1. The reset value is device specific. Please refer to the Module Configuration section at the end of this chapter.

7.7.1 Main Clock Control

Name: MCCTRL

Access Type: Read/Write

Offset: 0x0000

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	MCSEL		

- **MCSEL: Main Clock Select**

Table 7-6. Main clocks in AT32UC3C.

MCSEL[1:0]	Main clock source
0	System RC oscillator (RCSYS)
1	Oscillator 0
2	Oscillator 1
3	PLL0
4	PLL1
5	8 MHz RC oscillator (RC8M)
6	reserved
7	120 MHz RC oscillator (RC120M)
others	reserved

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

7.7.2 CPU Clock Select

Name: CPUSEL

Access Type: Read/Write

Offset: 0x0004

Reset Value: 0x00000000

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

- **CPUDIV, CPUSEL: CPU Division and Clock Select**
  - CPUDIV = 0: CPU clock equals main clock.
  - CPUDIV = 1: CPU clock equals main clock divided by $2^{(CPUSEL+1)}$.

Note that if CPUDIV is written to 0, CPUSEL should also be written to 0 to ensure correct operation.

Also note that writing this register clears POSCSR:CKRDY. The register must not be re-written until CKRDY goes high.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

7.7.3 HSB Clock Select

Name: HSBSEL

Access Type: Read Only

Offset: 0x0008

Reset Value: 0x00000000

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

This register is read-only and its content is always equal to CPU_SEL

7.7.4 PBx Clock Select

Name: PBxSEL

Access Type: Read/Write

Offset: 0x000C, 0x0010, 0x0014

Reset Value: 0x00000000

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

- **PBDIV, PBSEL: PBx Division and Clock Select**
  - PBDIV = 0: PBx clock equals main clock.
  - PBDIV = 1: PBx clock equals main clock divided by $2^{(PBSEL+1)}$.

Note that if PBDIV is written to 0, PBSEL should also be written to 0 to ensure correct operation.

Also note that writing this register clears POSCSR:CKRDY. The register must not be re-written until CKRDY goes high.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

7.7.5 Clock Mask

Name: CPU/HSB/PBA/PBBMASK

Access Type: Read/Write

Offset: 0x0020, 0x0024, 0x0028, 0x002C, 0x0030

Reset Value: -

31	30	29	28	27	26	25	24
MASK[31:24]							

23	22	21	20	19	18	17	16
MASK[23:16]							

15	14	13	12	11	10	9	8
MASK[15:8]							

7	6	5	4	3	2	1	0
MASK[7:0]							

- **MASK: Clock Mask**
  - If bit n is cleared, the clock for module n is stopped. If bit n is set, the clock for module n is enabled according to the current power mode. The number of implemented bits in each mask register, as well as which module clock is controlled by each bit, is shown in Table 7-7.

Table 7-7. Maskable module clocks in AT32UC3C.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK	PBCMASK
0	-	SAU	INTC	FLASHC	PDCA
1	OCD	PDCA	PM	USBC	MDMA
2	-	MDMA	SCIF	HMATRIX	USART1
3	-	USBC	AST	SAU	SPI0
4	-	CANIF	WDT	SMC	CANIF
5	-	HFLASHC	EIC	SDRAMC	TC0
6	-	PBA Bridge	FREQM	MACB	ADCIFA
7	-	PBB Bridge	GPIO	-	USART4
8	-	PBC Bridge	USART0	-	TWIM2
9	-	HSB RAM	USART2	-	TWIS2
10	-	EBI	USART3	-	-
11	-	MACB	SPI1	-	-
12	-	PEVC	TWIM0	-	-

Table 7-7. Maskable module clocks in AT32UC3C.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK	PBCMASK
13	-	-	TWIM1	-	-
14	-	-	TWIS0	-	-
15	-	-	TWIS1	-	-
16	-	-	IISC	-	-
17	-	-	PWM	-	-
18	-	-	QDEC0	-	-
19	-	-	QDEC1	-	-
20	-	-	TC1	-	-
21	-	-	PEVC	-	-
22	-	-	ACIFA0	-	-
23	-	-	ACIFA1	-	-
24	-	-	DACIFB0	-	-
25	-	-	DACIFB1	-	-
26	-	-	AW	-	-
31:27	-	-	-	-	-

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

7.7.6 Divided Clock Mask

Name: PBADIVMASK/PBBDIVMASK/PBCDIVMASK

Access Type: Read/Write

Offset: 0x0040, 0x0044, 0x0048

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-		-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	MASK[6:0]						

- **MASK: Clock Mask**

  If bit n is written to zero, the clock divided by $2^{(n+1)}$ is stopped. If bit n is written to one, the clock divided by $2^{(n+1)}$ is enabled according to the current power mode. Table 7-8 and Table 7-9 show what clocks are affected by the different MASK bits.

Table 7-8. PBA Divided Clock Mask

Bit	USART0	USART2	USART3	TC1
0	-			TIMER1_CLOCK2
1	-			-
2	CLK_PBA_USART_DIV			TIMER1_CLOCK3
3	-			-
4	-			TIMER1_CLOCK4
5	-			-
6	-			TIMER1_CLOCK5

Table 7-9. PBC Divided Clock Mask

Bit	USART1	USART4	TC0
0	-		TIMER0_CLOCK2
1	-		-
2	CLK_PBC_USART_DIV		TIMER0_CLOCK3
3	-		-

Table 7-9. PBC Divided Clock Mask

Bit	USART1	USART4	TC0
4	-		TIMER0_CLOCK4
5	-		-
6	-		TIMER0_CLOCK5

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

7.7.7 Clock Failure Detector Control Register

Name: CFDCTRL

Access Type: Read/Write

Offset: 0x0054

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SFV	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-		CFDEN

- **SFV: Store Final Value**
  - 0: The register is read/write
  - 1: The register is read-only, to protect against further accidental writes.
- **CFDEN: Clock Failure Detection Enable**
  - 0: Clock Failure Detector is disabled
  - 1: Clock Failure Detector is enabled

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

7.7.8 PM Unlock Register

Name: UNLOCK

Access Type: Write-Only

Offset: 0x0058

Reset Value: -

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	ADDR[9:8]	

7	6	5	4	3	2	1	0
ADDR[7:0]							

To unlock a write protected register, first write to the UNLOCK register with the address of the register to unlock in the ADDR field and 0xAA in the KEY field. Then, in the next PB access write to the register specified in the ADDR field.

- **KEY: Unlock Key**
  - Write this bit field to 0xAA to enable unlock.
- **ADDR: Unlock Address**
  - Write the address of the register to unlock to this bit field.

7.7.9 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x0C0

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

7.7.10 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x0C4

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

7.7.11 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x0C8

Reset Value: 0x00000000

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

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
This bit is cleared when the corresponding bit in IDR is written to one.
This bit is set when the corresponding bit in IER is written to one.

7.7.12 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x0CC

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	CKRDY	-	-	-	-	CFD

0: The corresponding interrupt is cleared.
1: The corresponding interrupt is pending.
This bit is cleared when the corresponding bit in ICR is written to one.
This bit is set when the corresponding interrupt occurs.

7.7.13 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x0D0

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	CKRDY	-	-	-	-	CFD

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in ISR.

7.7.14 Status Register

Name: SR

Access Type: Read-only

Offset: 0x0D4

Reset Value: 0x00000000

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

- **AE: Access Error**
  0: No access error has occured.
  1: A write to lock protected register without unlocking it has occured.
- **CKRDY: Clock Ready**
  0: The CKSEL register has been written, and the new clock setting is not yet effective.
  1: The synchronous clocks have frequencies as indicated in the CKSEL register.
- **CFD: Clock Failure Detected**
  0: Main clock is running correctly.
  1: Failure on main clock detected. Main clock is now running on RC osc.

7.7.15 Reset Cause

Name: RCAUSE

Access Type: Read-only

Offset: 0x0180

Reset Value: Latest Reset Source

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	BOD33	-	AWIRE	-	-	OCDRST

7	6	5	4	3	2	1	0
CPUERR		-	JTAG	WDT	EXT	BOD	POR

- **BOD33: Brown-out 3.3V reset**
  The CPU was reset due to the supply voltage being lower than the 3.3V Supply Monitor (BOD33) threshold level.
- **AWIRE: AWIRE Reset**
  The CPU was reset by the AWIRE
- **OCDRST: OCD Reset**
  The CPU was reset because the RES strobe in the OCD Development Control register has been written to one.
- **CPUERR: CPU Error**
  The CPU was reset because had detected an illegal access.
- **JTAG: JTAG Reset**
  The chip was reset by the JTAG system reset.
- **WDT: Watchdog Reset**
  The CPU was reset because of a watchdog time-out.
- **EXT: External Reset Pin**
  The CPU was reset due to the RESET pin being asserted.
- **BOD: Brown-out Reset**
  The CPU was reset due to the core supply voltage being lower than the brown-out threshold level.
- **POR: Power-on Reset**
  The CPU was reset due to the core supply voltage being lower than the power-on threshold level, or due to the input voltage being lower than the minimum required input voltage for the voltage regulator.

7.7.16 Wake Cause Register

Register name WCAUSE

Register access Read-only

Offset: 0x0184

Reset Value: Latest Reset Source

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	WCAUSE[17:16]	

15	14	13	12	11	10	9	8
WCAUSE[15:8]							

7	6	5	4	3	2	1	0
WCAUSE[7:0]							

- A bit in this register is set on wake up caused by the peripheral referred to in Table 7-10.

Table 7-10. Wake Cause

Bit	Wake Cause
1	USBC
1	CANIF-RXLINE[0]
2	CANIF-RXLINE[1]
3	-
4	TWI Slave 0
5	TWI Slave 1
6	TWI Slave 2
15:7	-
16	EIC
17	AST
31:18	-

7.7.17 Asynchronous Wake Up Enable Register

Register name AWEN

Register access Read/Write

Offset: 0x0188

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AWEN[31:24]							

23	22	21	20	19	18	17	16
AWEN[23:16]							

15	14	13	12	11	10	9	8
AWEN[15:8]							

7	6	5	4	3	2	1	0
AWEN[7:0]							

Each bit in this register corresponds to an asynchronous wake up, according to Table 7-11.

0: The corresponding wake up is disabled.

1: The corresponding wake up is enabled

Table 7-11. Asynchronous Wake Up

Bit	Asynchronous Wake Up
0	USBWEN
1	CANIF0WEN
2	CANIF1WEN
3	-
4	TWIS0WEN
5	TWIS1WEN
6	TWIS2WEN
31:7	-

7.7.18 Configuration Register

Name: CONFIG

Access Type: Read-Only

Offset: 0x03F8

Reset Value: 0x000000C3

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
HSBPEVC		-	-	PBD	PBC	PBB	PBA

This register shows the configuration of the PM.

- **HSBPEVC:HSB PEVC Clock Implemented**
  0: HSBPEVC not implemented.
  1: HSBPEVC implemented.
- **PBD: PBD Implemented**
  0: PBD not implemented.
  1: PBD implemented.
- **PBC: PBC Implemented**
  0: PBC not implemented.
  1: PBC implemented.
- **PBB: PBB Implemented**
  0: PBB not implemented.
  1: PBB implemented.
- **PBA: PBA Implemented**
  0: PBA not implemented.
  1: PBA implemented.

7.7.19 Version Register

Name: VERSION

Access Type: Read-Only

Offset: 0x03FC

Reset Value: 0x00000410

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

7.8 Module Configuration

The specific configuration for each PM instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 7-12. Module clock name

Module name	Clock name	Description
PM	CLK_PM	Peripheral Bus clock from the PBA clock domain

Table 7-13. Register Reset Values

Register	Reset Value
CONFIG	0x00000087
VERSION	0x00000412

Table 7-14. Effect of the different Reset Events

	Power-On Reset	External Reset	Watchdog Reset	1.8V BOD Reset	3.3V BOD Reset	CPU Error Reset	OCD Reset	JTAG Reset	Awire Reset
CPU/HSB/PBx (excluding Power Manager)	Y	Y	Y	Y	Y	Y	Y	Y	Y
32 KHz oscillator	Y	N	N	N	N	N	N	N	N
AST registers, except interrupt registers	Y	N	N	N	N	N	N	N	N
GPLP registers	Y	N	N	N	N	N	N	N	N
Watchdog control register	Y	Y	N	Y	Y	Y	Y	Y	Y
Voltage Regulator Calibration register	Y	N	N	N	N	N	N	N	N
RCSYS Calibration register	Y	N	N	N	N	N	N	N	N
1.8V BOD control register	Y	Y	Y	N	Y	Y	Y	Y	Y
3.3V BOD control register	Y	Y	Y	Y	N	Y	Y	Y	Y
Bandgap control register	Y	Y	Y	N	Y	Y	Y	Y	Y
Clock control registers	Y	Y	Y	Y	Y	Y	Y	Y	Y
OSC control registers	Y	Y	Y	Y	Y	Y	Y	Y	Y
OCD system and OCD registers	Y	Y	N	Y	Y	Y	N	Y	Y

8. System Control Interface (SCIF)

Rev: 1.0.2.0

8.1 Features

- **Controls integrated oscillators and PLLs**
- **Supports 2x General Purpose crystal oscillators, 0.4MHz-20MHz**
- **Supports 2x Phase-Locked-Loop, 80-240 MHz**
- **Supports 32 KHz low power oscillator (OSC32K)**
- **Integrated 115KHz RC Oscillator (RCSYS)**
- **Controls 8 MHz / 1 MHz integrated RC oscillator (RC8M)**
- **Controls 120 MHz integrated RC oscillator (RC120M)**
- **Generic clocks with wide frequency range provided**
- **Controls bandgap voltage reference through control and calibration registers**
- **Controls Brown-out detectors and supply monitors**
- **Controls Voltage Regulator behavior and calibration**
- **Two 32-bit general purpose low power registers**

8.2 Description

The System Control Interface (SCIF) controls the Oscillators, PLL, Generic Clocks, BODs, the voltage regulators and general purpose low power registers.

8.3 I/O Lines Description

Table 8-1. I/O Lines Description

Pin Name	Pin Description	Type
XIN0	Crystal 0 Input	Analog/Digital
XIN1	Crystal 1 Input	Analog/Digital
XIN32	Crystal 32 Input	Analog/Digital
XOUT0	Crystal 0 Output	Analog
XOUT1	Crystal 1 Output	Analog
XOUT32	Crystal 32 Output	Analog
GCLK[1:0]	Generic Clock Output	Digital

8.4 Product Dependencies

8.4.1 I/O Lines

The SCIF provides a number of generic clock outputs, which can be connected to output pins, multiplexed with GPIO lines. The programmer must first program the GPIO controller to assign these pins to their peripheral function. If the I/O pins of the SCIF are not used by the application, they can be used for other purposes by the GPIO controller. Oscillators pins are also multiplexed with GPIO. When oscillators are used, the related pins are controlled directly by the SCIF, overriding GPIO settings.

8.4.2 Interrupt

The SCIF interrupt line is connected to one of the internal sources of the interrupt controller. Using the SCIF interrupt requires the interrupt controller to be programmed first.

8.4.3 Debug Operation

The SCIF module does not interact with debug operations.

8.4.4 Clocks

The SCIF controls all oscillators on the part. Those oscillators can then be used as sources for for generic clocks (handled by the SCIF) and for the CPU and peripherals (in this case, selection of source is done by the Power Manager).

8.5 Functional Description

8.5.1 Oscillator Operation

The main oscillator is designed to be used with an external 0.4 to 20 MHz crystal and two biasing capacitors, as shown in Figure 8-1. The oscillator can be used for the main clock in the device, as described in the Power Manager chapter. The oscillator can be used as source for the generic clocks, as described in "Generic Clocks" on page 84.

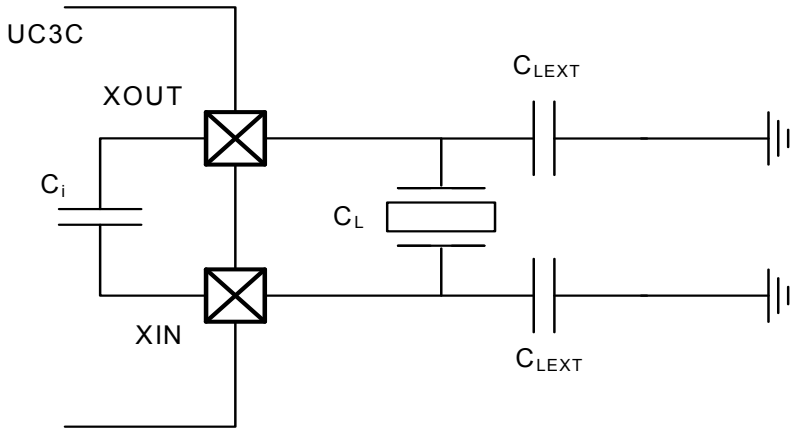
The oscillator is disabled by default after reset. When the oscillator is disabled, the XIN and XOUT pins can be used as general purpose I/Os. When the oscillator is configured to use an external clock, the clock must be applied to the XIN pin while the XOUT pin can be used as a general purpose I/O.

The oscillator can be enabled by writing a one to the OSCEN bit in OSCCTRLn. Operation mode (external clock or crystal) is chosen by writing to the MODE field in OSCCTRLn. The oscillator is automatically switched off in certain sleep modes to reduce power consumption, as described in the Power Manager chapter.

After a hard reset, or when waking up from a sleep mode that disabled the oscillator, the oscillator may need a certain amount of time to stabilize on the correct frequency. This start-up time can be set in the OSCCTRLn register.

The SCIF masks the oscillator outputs during the start-up time, to ensure that no unstable clocks propagate to the digital logic. The OSCnRDY bits in PCLKSR are automatically set and cleared according to the status of the oscillators. A zero to one transition on these bits can also be configured to generate an interrupt, as described in "Interrupts" on page 88.

Figure 8-1. Oscillator connections



8.5.2 32KHz Oscillator (OSC32K) Operation

The 32KHz oscillator (OSC32K) operates as described for the Oscillator above. The 32KHz oscillator is used as source clock for the Asynchronous Timer and the Watchdog Timer. The 32KHz oscillator can be used as source for the generic clocks, as described in "Generic Clocks" on page 84.

The oscillator is disabled by default, but can be enabled by writing a one to the OSC32EN bit in OSCCTRL32. The oscillator is an ultra-low power design and remains enabled in all sleep modes.

While the 32KHz oscillator is disabled, the XIN32 and XOUT32 pins are available as general purpose I/Os. When the oscillator is configured to work with an external clock (MODE field in OSCCTRL32 register), the external clock must be connected to XIN32 while the XOUT32 pin can be used as a general purpose I/O.

The startup time of the 32KHz oscillator can be set in the OSCCTRL32, after which OSC32RDY in PCLKSR is set. An interrupt can be generated on a zero to one transition of OSC32RDY.

As a crystal oscillator usually requires a very long startup time (up to 1 second), the 32 KHz oscillator will keep running across resets, except Power-On-Reset.

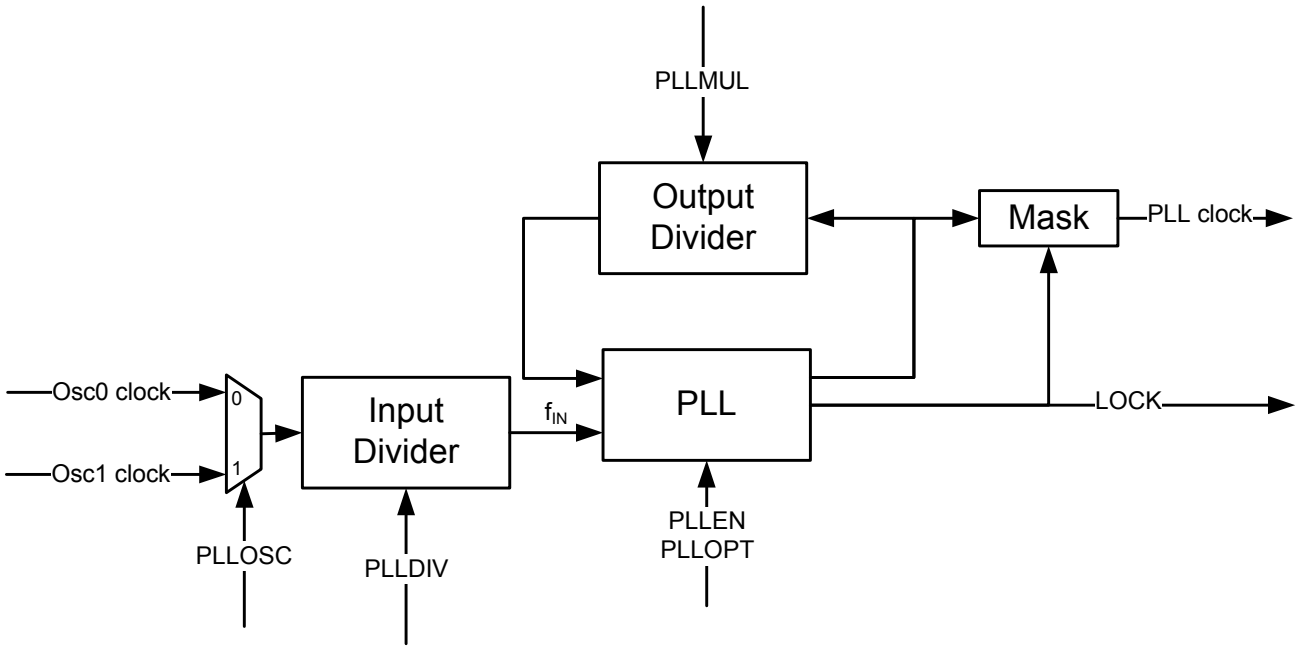
The 32KHz oscillator is not controlled by the sleep controller, and will run in all sleep modes if enabled.

8.5.3 PLL Operation

The device contains two PLLs, PLL0 and PLL1. These are disabled by default, but can be enabled to provide high frequency source clocks for synchronous or generic clocks. The PLLs can take either Oscillator 0, Oscillator 1 or 8MHz RC Oscillator (RC8M) as reference clock. The PLL output is divided by a multiplication factor, and the PLL compares the resulting clock to the reference clock. The PLL will adjust its output frequency until the two compared clocks are equal, thus locking the output frequency to a multiple of the reference clock frequency.

When the PLL is switched on, or when changing the clock source or multiplication factor for the PLL, the PLL is unlocked and the output frequency is undefined. The PLL clock for the digital logic is automatically masked when the PLL is unlocked, to prevent connected digital logic from receiving a too high frequency and thus becoming unstable.

Figure 8-2. PLL with control logic and filters



8.5.3.1 *Enabling the PLL*

PLLn is enabled by writing a one to the PLLEN bit in the PLLn register. PLLOSC selects Oscillator 0 or 1 as clock source. The PLLMUL and PLLDIV bit fields must be written with the multiplication and division factors.

The PLLn.PLLOPT field should be set to proper values according to the PLL operating frequency. The PLLOPT field can also be set to divide the output frequency of the PLLs by 2.

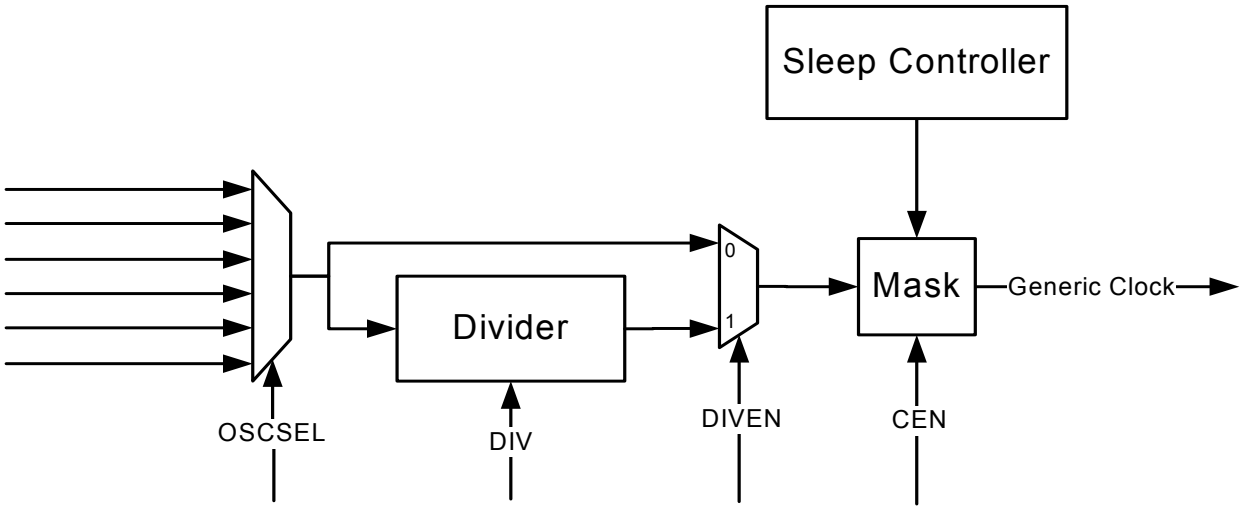
The lock signal for each PLL is available as a LOCKn flag in POSCSR. An interrupt can be generated on a 0 to 1 transition of these bits.

8.5.4 Generic Clocks

Timers, communication modules, and other modules connected to external circuitry may require specific clock frequencies to operate correctly. The SCIF contains an implementation defined number of generic clocks that can provide a wide range of accurate clock frequencies.

Each generic clock module runs from either clock source listed in "Generic Clock Source" on page 118. The selected source can optionally be divided by any even integer up to 512. Each clock can be independently enabled and disabled, and is also automatically disabled along with peripheral clocks by the Sleep Controller in the Power Manager.

Figure 8-3. Generic clock generation



8.5.4.1 *Enabling a generic clock*

A generic clock is enabled by writing a one to the CEN bit in GCCTRL to one. Each generic clock can individually select a clock source by setting the OSCSEL bits. The source clock can optionally be divided by writing a one to DIVEN and the division factor to DIV, resulting in the output frequency:

$$f_{GCLK} = f_{SRC} / (2*(DIV+1))$$

8.5.4.2 *Disabling a generic clock*

The generic clock can be disabled by writing a zero to CEN or entering a sleep mode that disables the PB clocks. In either case, the generic clock will be switched off on the first falling edge after the disabling event, to ensure that no glitches occur. If CEN is written to zero, the bit will still read as one until the next falling edge occurs, and the clock is actually switched off. When writing a zero to CEN, the other bits in GCCTRL should not be changed until CEN reads as zero, to avoid glitches on the generic clock.

When the clock is disabled, both the prescaler and output are reset.

8.5.4.3 *Changing clock frequency*

When changing generic clock frequency by writing GCCTRL, the clock should be switched off by the procedure above, before being re-enabled with the new clock source or division setting. This prevents glitches during the transition.

8.5.4.4 *Generic clock implementation*

In AT32UC3C, the generic clocks are allocated to different functions as shown in Table 8-2.

Table 8-2. Generic clock allocation

Clock number	Function	Name
0	USB clock (48 MHz)	GCLK_USBC
1	CANIF	GCLK_CANIF
2	AST	GCLK_AST

Table 8-2. Generic clock allocation

Clock number	Function	Name
3	-	
4	PWM	GCLK_PWM
5	QDEC0	GCLK_QDEC0
6	QDEC1	GCLK_QDEC1
7	GCLK event, mapped to event number 16. See the Module Configuration of PEVC for more details.	
8	GCLK event, mapped to event number 17. See the Module Configuration of PEVC for more details.	
9	GCLK[0] output pin	
10	GCLK[1] output pin	
11	IISC	GCLK_IISC

8.5.5 1.8V Brown Out Detection (BOD18)

The 1.8V Brown-Out Detector (BOD18) monitors the VDDCORE supply pin and compares the supply voltage to the brown-out detection level, as set in BOD.LEVEL. The BOD18 is disabled by default, but can be enabled either by software or by flash fuses. The 1.8V Brown-Out Detector can either generate an interrupt or a reset when the supply voltage is below the brown-out detection level. In any case, the BOD18 output value is given by the PCLKSR.BODDET bit.

Note that any change to the BOD.LEVEL field of the BOD register should be done with the BOD18 deactivated to avoid spurious reset or interrupt. When turned-on, the BOD18 output will be masked during one half of a RCSYS clock cycle and two main clocks cycles to avoid false results.

If the JTAG or the AWIRE is enabled, the BOD18 reset and interrupt will be masked.

See Electrical Characteristics for parametric details.

Although it is not recommended, it is still possible to override the default factory settings by writing to those registers. To prevent unexpected writes due to software bugs, write access to this register is protected by a locking mechanism, for details please refer to the UNLOCK register description.

8.5.6 3.3V Brown Out Detection (BOD33)

The 3.3V Brown-Out Detector (BOD33) monitors the VDDIN_5 supply pin and compares the supply voltage to the brown-out detection level, as set in BOD33.LEVEL. The BOD33 is disabled by default, but can be enabled by software or by flash fuses. The 3.3V Brown-Out Detector can generate an interrupt or a reset when the supply voltage is below the brown-out detection level. In any case, the BOD33 value is given by the PCLKSR.BOD33DET bit.

Note that any change to the BOD33.LEVEL field of the BOD33 register should be done with the BOD33 deactivated to avoid spurious interrupt. When turned-on, the BOD33 output will be masked during one half of a RCSYS clock cycle and two main clocks cycles to avoid false results.

If the JTAG or the AWIRE is enabled, the BOD33 reset and interrupt will be masked.

See Electrical Characteristics for parametric details.

To prevent unexpected writes due to software bugs, write access to this register is protected by a locking mechanism, for details please refer to the UNLOCK register description.

8.5.7 5V Brown Out Detection (BOD50)

The 5V Brown-Out Detector (BOD50) monitors the VDDIN_5 supply pin and compares the supply voltage to the brown-out detection level, as set in BOD50.LEVEL. The BOD50 is disabled by default, but can be enabled by software. The 5V Brown-Out Detector can generate an interrupt when the supply voltage is below the brown-out detection level. In any case, the BOD50 output value is given by the PCLKSR.BOD50DET bit.

Note that any change to the BOD50.LEVEL field of the BOD50 register should be done with the BOD50 deactivated to avoid spurious interrupt. When turned-on, the BOD50 output will be masked during one half of a RCSYS clock cycle and two main clocks cycles to avoid false results.

If the JTAG or the AWIRE is enabled, the BOD50 interrupt will be masked.

See Electrical Characteristics for parametric details.

To prevent unexpected writes due to software bugs, write access to this register is protected by a locking mechanism, for details please refer to the UNLOCK register description.

8.5.8 Bandgap

The Flash memory, the Brown-Out Detectors need a stable voltage reference to operate. This reference voltage is provided by an internal Bandgap voltage reference. This reference is automatically turned on at startup and turned off during DEEPSTOP and STATIC sleep modes to save power.

The Bandgap voltage reference is calibrated through the BGCR.CALIB field. This field is loaded after a Power On Reset with default values stored in factory-programmed flash fuses.

It is not recommended to override default factory settings as it may prevent correct operation of the Flash and BODs. To prevent unexpected writes due to software bugs, write access to this register is protected by a locking mechanism, for details please refer to the UNLOCK register description.

8.5.9 Voltage Regulators

The embedded 1.8V regulator provides the core supply. The embedded 3.3V voltage regulator is used to supply the USB pads. Both regulators are turned on at startup.

If the application is supplied with a voltage range around 3.3V or the application does not use the USB interface, the 3.3V voltage regulator has to be turned off by writing 11 binary to VREGCTRL.VREG33CTL.

The 1.8V voltage regulator has its own voltage reference that is calibrated through the VREGCR.CALIB field. This field is loaded after a Power On Reset with default values stored in factory-programmed flash fuses.

Although it is not recommended, it is still possible to override the default factory settings by writing to those registers. To prevent unexpected writes due to software bugs, write access to this register is protected by a locking mechanism, for details please refer to the UNLOCK register description.

8.5.10 System RC Oscillator (RCSYS)

The system RC oscillator (RCSYS) has a 3 cycles startup time, and is always available except in the STATIC sleep mode. The system RC oscillator operates at a nominal frequency of 115 kHz, and is calibrated using the RCCR.CALIB Calibration field. After a Power On Reset, the RCCR.CALIB field is loaded with a factory defined value stored in the Flash fuses.

Although it is not recommended, it is still possible to override the default factory settings by writing to the RCCR.CALIB field. To prevent unexpected writes due to software bugs, write access to this register is protected by a locking mechanism, for details please refer to the UNLOCK register description.

8.5.11 8MHz / 1MHz RC Oscillator (RC8M)

The 8MHz / 1MHz RC oscillator (RC8M) operates at a nominal frequency of 8MHz or 1 MHz according to RCCR8.FREQMODE bit. It is calibrated using the RCCR8.CALIB Calibration field. After a Power On Reset, the RCCR8.CALIB field is automatically loaded with the RC8M_CALIB field of the Oscillator Calibration register, a factory defined value stored in the factory page of the Flash.

If the user wants to run the oscillator at 1MHz or if the device operates at VDDIN_5 within the 5V range, it has to write the RCCR8.CALIB field with the corresponding field from the Oscillator Calibration register.

Although it is not recommended, it is still possible to override the default factory settings by writing to the RCCR8.CALIB field. To prevent unexpected writes due to software bugs, write access to this register is protected by a locking mechanism, for details please refer to the UNLOCK register description.

8.5.12 RC120M

The 120MHz RC Oscillator can be used for the main clock in the device, as described in the Power Manager chapter. To enable the clock, the user must write a one to the EN bit in the RC120MCR register, and read back the RC120MCR register until the EN bit reads one. The clock is disabled by writing a zero to the EN bit.

The oscillator is automatically switched off in certain sleep modes to reduce power consumption, as described in the Power Manager chapter.

8.5.13 General Purpose Low Power Registers (GPLP)

The GPLP registers are 32-bit registers that are reset only by power-on-reset. User software can use these registers to save context variables in a very low power mode.

8.5.14 Interrupts

The SCIF has separate interrupt requests:

- AE - Access Error:
  - Set when a protected SCIF register was accessed without first being correctly unlocked.
- PLL1LOCKLOST - PLL1 lock lost:
  - Set when a 0 to 1 transition on the PCLKSR.PLL1LOCKLOST bit is detected.
- PLL0LOCKLOST - PLL0 lock lost:
  - Set when a 0 to 1 transition on the PCLKSR.PLL0LOCKLOST bit is detected.

- BOD50DET - 5V Brown out detection:
  - Set when a 0 to 1 transition on the PCLKSR.BOD50DET bit is detected.
- BOD33DET - 3.3V Brown out detection:
  - Set when a 0 to 1 transition on the PCLKSR.BOD33DET bit is detected.
- BODDET - 1.8V Brown out detection:
  - Set when a 0 to 1 transition on the PCLKSR.BODDET bit is detected.
- PLL1LOCK - PLL1 locked:
  - Set when a 0 to 1 transition on the PCLKSR.PLL1LOCK bit is detected.
- PLL0LOCK - PLL0 locked:
  - Set when an 0 to 1 transition on the PCLKSR.PLL0LOCK bit is detected.
- RCOSC8MRDY - 8MHz / 1MHz RCOSC Ready:
  - Set when a 0 to 1 transition on the PCLKSR.RCOSC8MRDY bit is detected.
- OSC32RDY - 32KHz Oscillator Ready:
  - Set when a 0 to 1 transition on the PCLKSR.OSC32RDY bit is detected.
- OSCR1DY - OSC1 Ready:
  - Set when a 0 to 1 transition on the PCLKSR.OSC1RDY bit is detected.
- OSCR0DY - OSC0 Ready:
  - Set when a 0 to 1 transition on the PCLKSR.OSC0RDY bit is detected.

This allows the user to allocate separate handlers and priorities to the different interrupt types.

The interrupt request will be generated if the corresponding bit in the Interrupt Mask Register (IMR) is set. Bits in IMR are set by writing a one to the corresponding bit in the Interrupt Enable Register (IER), and cleared by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The interrupt request remains active until the corresponding bit in the Interrupt Status Register (ISR) is cleared by writing a one to the corresponding bit in the Interrupt Clear Register (ICR).

8.6 User Interface

Table 8-3. SCIF Register Memory Map

Offset	Register	Register Name	Access	Reset
0x0000	Interrupt Enable Register	IER	Write-Only	0x00000000
0x0004	Interrupt Disable Register	IDR	Write-Only	0x00000000
0x0008	Interrupt Mask Register	IMR	Read-Only	0x00000000
0x000C	Interrupt Status Register	ISR	Read-Only	0x00000000
0x0010	Interrupt Clear Register	ICR	Write-Only	0x00000000
0x0014	Power and Clocks Status Register	PCLKSR	Read-Only	0x00000000
0x0018	Unlock Register	UNLOCK	Write-Only	0x00000000
0x001C	PLL0 Control Register	PLL0	Read/Write	0x00000000
0x0020	PLL1 Control Register	PLL1	Read/Write	0x00000000
0x0024	Oscillator0 Control Register	OSCCTRL0	Read/Write	0x00000000
0x0028	Oscillator1 Control Register	OSCCTRL1	Read/Write	0x00000000
0x002C	1.8V BOD Control Register	BOD	Read/Write	0x00000000
0x0030	Bandgap Calibration Register	BGCR	Read/Write	0x00000000
0x0034	3.3V BOD Control Register	BOD33	Read/Write	0x00000000
0x0038	5V BOD Control Register	BOD50	Read/Write	0x00000000
0x003C	Voltage Regulator Calibration Register	VREGCR	Read/Write	0x00000000
0x0040	Voltage Regulator Control Register	VREGCTRL	Read/Write	0x00000000
0x0044	RCSYS Calibration Register	RCCR	Read/Write	0x00000000
0x0048	8MHz / 1 MHz RC Oscillator Control Register	RCCR8	Read/Write	0x00000000
0x004C	Oscillator 32 Control Register	OSCCTRL32	Read/Write	0x00000000
0x0058	120MHz RC Oscillator Control Register	RC120MCR	Read/Write	0x00000000
0x005C	General Purpose Low Power Register 0	GPLP0	Read/Write	0x00000000
0x0060	General Purpose Low Power Register 1	GPLP1	Read/Write	0x00000000
0x0064-0x008C	Generic Clock Control	GCCTRL	Read/Write	0x00000000
0x03C8	PLL interface Version Register	PLLVERSION	Read-Only	-(1)
0x03CC	Oscillator 0/1 Interface Version Register	OSCVERSION	Read-Only	-(1)
0x03D0	1.8V BOD Interface Version Register	BODVERSION	Read-Only	-(1)
0x03D4	3.3/5.0V BOD Interface Version Register	BODBVERSION	Read-Only	-(1)
0x03D8	Voltage Regulator interface Version Register	VREGVERSION	Read-Only	-(1)
0x03DC	RCSYS Interface Version Register	RCCRVERSION	Read-Only	-(1)
0x03E0	8MHz/1MHz RCOSC Interface Version Register	RCCR8VERSION	Read-Only	-(1)
0x03E4	32 KHz Oscillator Interface Version Register	OSC32VERSION	Read-Only	-(1)
0x03F0	120MHz RC Oscillator Interface Version Register	RC120MVERSION	Read-Only	-(1)

Table 8-3. SCIF Register Memory Map

Offset	Register	Register Name	Access	Reset
0x03F4	GPLP Version Register	GPLPVERSION	Read-Only	-(1)
0x03F8	Generic Clock Version Register	GCLKVERSION	Read-Only	-(1)
0x03FC	SCIF Version Register	VERSION	Read-Only	-(1)

Note: 1. The reset value is device specific. Please refer to the Module Configuration section at the end of this chapter.

8.6.1 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x0000

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	PLL1_LOCK LOST	PLL0_LOCK LOST	BOD50DET
7	6	5	4	3	2	1	0
BOD33DET	BODDET	PLL1_LOCK	PLL0_LOCK	RCOSC8MRDY	OSC32RDY	OSC1RDY	OSC0RDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

8.6.2 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x0004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	PLL1_LOCK LOST	PLL0_LOCK LOST	BOD50DET

7	6	5	4	3	2	1	0
BOD33DET	BODDET	PLL1_LOCK	PLL0_LOCK	RCOSC8MRDY	OSC32RDY	OSC1RDY	OSC0RDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

8.6.3 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x0008

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	PLL1_LOCK LOST	PLL0_LOCK LOST	BOD50DET

7	6	5	4	3	2	1	0
BOD33DET	BODDET	PLL1_LOCK	PLL0_LOCK	RCOSC8MRDY	OSC32RDY	OSC1RDY	OSC0RDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
This bit is cleared when the corresponding bit in IDR is written to one.
This bit is set when the corresponding bit in IER is written to one.

8.6.4 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x000C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	PLL1_LOCK LOST	PLL0_LOCK LOST	BOD50DET

7	6	5	4	3	2	1	0
BOD33DET	BODDET	PLL1_LOCK	PLL0_LOCK	RCOSC8MRDY	OSC32RDY	OSC1RDY	OSC0RDY

0: The corresponding interrupt is cleared.

1: The corresponding interrupt is pending.

This bit is cleared when the corresponding bit in ICR is written to one.

This bit is set when the corresponding interrupt occurs.

8.6.5 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x0010

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	PLL1_LOCK LOST	PLL0_LOCK LOST	BOD50DET

7	6	5	4	3	2	1	0
BOD33DET	BODDET	PLL1_LOCK	PLL0_LOCK	RCOSC8MRDY	OSC32RDY	OSC1RDY	OSC0RDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in ISR.

8.6.6 Power and Clocks Status Register

Name: PCLKSR

Access Type: Read-Only

Offset: 0x0014

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	PLL1_LOCK LOST	PLL0_LOCK LOST	BOD50DET

7	6	5	4	3	2	1	0
BOD33DET	BODDET	PLL1_LOCK	PLL0_LOCK	RCOSC8MRDY	OSC32RDY	OSC1RDY	OSC0RDY

- **AE: SCIF Access Error**
  - 0: No access error has occurred on the SCIF.
  - 1: An access error has occurred on the SCIF.
- **PLLL1_LOCKLOST: PLL1 lock lost value**
  - 0: PLL1 has not lost its lock or has never been enabled.
  - 1: PLL1 has lost its lock, either by disabling the PLL1 or due to faulty operation.
- **PLLL0_LOCKLOST: PLL0 lock lost value**
  - 0: PLL0 has not lost its lock or has never been enabled.
  - 1: PLL0 has lost its lock, either by disabling the PLL0 or due to faulty operation.
- **BOD50DET: 5.0V Brown out detection**
  - 0: BOD50 not enabled or the 5.0V power supply is above the BOD50 threshold.
  - 1: BOD50 enabled and the 5.0V power supply is going below BOD50 threshold.
- **BOD33DET: 3.3V Brown out detection**
  - 0: BOD33 not enabled or the 3.3V power supply is above the BOD33 threshold.
  - 1: BOD33 enabled and the 3.3V power supply is going below BOD33 threshold.
- **BODDET: 1.8V Brown out detection**
  - 0: BOD18 not enabled or the 1.8V power supply is above the BOD18 threshold.
  - 1: BOD18 enabled and the 1.8V power supply is going below BOD18 threshold.
- **PLL1_LOCK: PLL1 Locked on Accurate value**
  - 0: PLL1 is unlocked on accurate value.
  - 1: PLL1 is locked on accurate value, and is ready to be selected as clock source with an accurate output clock.
- **PLL0_LOCK: PLL0 Locked on Accurate value**
  - 0: PLL0 is unlocked on accurate value.
  - 1: PLL0 is locked on accurate value, and is ready to be selected as clock source with an accurate output clock.

- **RCOSC8MRDY: 8MHz / 1MHz RCOSC Ready**
  0: 8MHz / 1MHz RC Oscillator not enabled or not ready.
  1: 8MHz / 1MHz RC Oscillator is stable and ready to be used as clock source.
- **OSC32RDY: 32 KHz oscillator Ready**
  0: Oscillator 32 not enabled or not ready.
  1: Oscillator 32 is stable and ready to be used as clock source.
- **OSC1RDY: OSC1Ready**
  0: Oscillator not enabled or not ready.
  1: Oscillator is stable and ready to be used as clock source.
- **OSC0RDY: OSC0Ready**
  0: Oscillator not enabled or not ready.
  1: Oscillator is stable and ready to be used as clock source.

8.6.7 Unlock Register

Name: UNLOCK

Access Type: Write-Only

Offset: 0x0018

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	ADDR[9:8]	

7	6	5	4	3	2	1	0
ADDR[7:0]							

To unlock a write protected register, first write to the UNLOCK register with the address of the register to unlock in the ADDR field and 0xAA in the KEY field. Then, in the next PB access write to the register specified in the ADDR field.

- **KEY: Unlock Key**

  Write this bit field to 0xAA to enable unlock.

- **ADDR: Unlock Address**

  Write the address of the register to unlock to this field.

8.6.8 PLL Control Register

Name: PLL0,1

Access Type: Read/Write

Offset: 0x001C,0x0020

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	PLLCOUNT					

23	22	21	20	19	18	17	16
-	-	-	-	PLLMUL			

15	14	13	12	11	10	9	8
-	-	-	-	PLLDIV			

7	6	5	4	3	2	1	0
-	-	PLLOPT			PLLOSC		PLLEN

- **PLLCOUNT: PLL Count**

  Specifies the number of slow clock cycles before ISR.LOCKn is set after PLLn has been written, or after PLLn has been automatically re-enabled after exiting a sleep mode.
- **PLLMUL: PLL Multiply Factor**
- **PLLDIV: PLL Division Factor**

  These fields determine the ratio of the output frequency of the internal VCO of the PLL (f_{vco}) to the source oscillator frequency:

  f_{vco} = (PLLMUL+1) / (PLLDIV) * f_{osc} if PLLDIV > 0

  f_{vco} = 2 * (PLLMUL+1) * f_{osc} if PLLDIV = 0

  According to PLLOPT[1] bit, it gives the following PLL frequency value f_{PLL}:

  if the PLLOPT[1] bit is set to 0: $f_{PLL} = f_{VCO}$

  if the PLLOPT[1] bit is set to 1: $f_{PLL} = f_{VCO} / 2$

  Note that the PLLMUL field cannot be equal to 0 or 1, or the behavior of the PLL will be undefined.

  PLLDIV gives also the input frequency of the PLL (f_{IN}):

  if the PLLDIV field is set to 0: $f_{IN} = f_{OSC}$

  if the PLLDIV field is greater than 0: $f_{IN} = f_{OSC} / (2 * PLLDIV)$
- **PLLOPT: PLL Option**

  Select the operating range for the PLL.

  PLLOPT[0]: Select the VCO frequency range.

  PLLOPT[1]: Enable the extra output divider.

  PLLOPT[2]: Disable the Wide-Bandwidth mode (Wide-Bandwidth mode allows a faster startup time and out-of-lock time).

Table 8-4. PLLOPT Fields Description

	Description
PLLOPT[0]: VCO frequency	
0	160MHz<f_{vco}<240MHz
1	80MHz<f_{vco}<180MHz
PLLOPT[1]: Output divider	
0	$f_{PLL} = f_{vco}$
1	$f_{PLL} = f_{vco}/2$
PLLOPT[2]	
0	Wide Bandwidth Mode enabled
1	Wide Bandwidth Mode disabled

- **PLLOSC: PLL Oscillator Select**
  - 0: Oscillator 0 is the source for the PLL.
  - 1: Oscillator 1 is the source for the PLL.
  - 2: 8MHz/1MHz RCOSC is the source for the PLL.
  - 3: Reserved.
- **PLLEN: PLL Enable**
  - 0: PLL is disabled.
  - 1: PLL is enabled.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.9 Oscillator Control Register

Name: OSCCTRL0,1

Access Type: Read/Write

Offset: 0x0024,0x0028

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	OSCEN
15	14	13	12	11	10	9	8
-	-	-	-	STARTUP			
7	6	5	4	3	2	1	0
-	-	-	-	AGC	GAIN		MODE

- **OSCEN**
  0: Disable the Oscillator.
  1: Enable the Oscillator.
- **STARTUP: Oscillator Startup Time**
  Select startup time for the oscillator.

Table 8-5. Startup time for oscillators 0 and 1

STARTUP	Number of RC oscillator clock cycle	Approximative Equivalent time (RCSYS = 115 kHz)
0	0	0
1	64	560 us
2	128	1.1 ms
3	2048	18 ms
4	4096	36 ms
5	8192	71 ms
6	16384	142 ms
7	32768	285 ms
8	4	35 us
9	8	70 us
10	16	140 us
11	32	280 us
12	256	2.2 ms

Table 8-5. Startup time for oscillators 0 and 1

STARTUP	Number of RC oscillator clock cycle	Approximative Equivalent time (RCSYS = 115 kHz)
13	512	4.5 ms
14	1024	9 ms
15	Reserved	Reserved

- **AGC: Automatic Gain Control**
  0: Disable the automatic gain control of the Oscillator.
  1: Enable the automatic gain control of the Oscillator.
- **GAIN: Oscillator Gain**
  Set the gain of the Oscillator.

Table 8-6. Gain value for oscillators 0 and 1

GAIN	crystal frequency
0	< 2 MHz
1	between 2 and 10 MHz
2	between 10 and 16 MHz
3	> 16 MHz

- **MODE: Oscillator Mode**
  0: External clock connected on XIN, XOUT can be used as an I/O (no crystal), Disable the Oscillator.
  1: Enable the Oscillator.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.10 1.8V BOD Control Register

Name: BOD

Access Type: Read/Write

Offset: 0x002C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SFV	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD
15	14	13	12	11	10	9	8
-	-	-	-	-	-	CTRL	
7	6	5	4	3	2	1	0
-	HYST	LEVEL					

- **SFV: Store Final Value**
  0: The register is read/write.
  1: The register is read-only, to protect against further accidental writes.
- **FCD: BOD18 Fuse Calibration Done**
  Set to 1 when the CTRL, HYST and LEVEL fields have been updated by the Flash fuses after a reset.
  0: The flash calibration will be redone after any reset.
  1: The flash calibration will not be redone after a BOD18 reset.
- **CTRL: BOD18 Control**

Table 8-7. Operation mode for BOD18

CTRL	Description
0x0	BOD18 is off
0x1	BOD18 is enabled and can reset the chip
0x2	BOD18 is enabled, but cannot reset the chip. Only interrupt will be sent to interrupt controller, if enabled in the IMR register.
0x3	Reserved

- **HYST: BOD18 Hysteresis**
  0: No hysteresis.
  1: Hysteresis on.

- **LEVEL: BOD18 Level**

  This field sets the triggering threshold of the BOD18. See Electrical Characteristics for actual voltage levels.

  Note that any change to the LEVEL field of the BOD register should be done with the BOD18 deactivated to avoid spurious reset or interrupt.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.11 Bandgap Calibration Register

Name: BGCR

Access Type: Read/Write

Offset: 0x0030

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SFV	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	CALIB		

- **SFV: Store Final Value**
  0: The register is read/write.
  1: The register is read-only, to protect against further accidental writes.
- **FCD: Flash Calibration Done**
  Set to 1 when the CALIB field has been updated by the Flash fuses after a reset.
  0: The flash calibration will be redone after any reset.
  1: The flash calibration will not be redone after a BOD18 reset.
- **CALIB: Calibration value**
  Calibration value for Bandgap. See Electrical Characteristics for voltage values.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.12 3.3V BOD Control Register

Name: BOD33

Access Type: Read/Write

Offset: 0x0034

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SFV	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CTRL	

7	6	5	4	3	2	1	0
-	HYST	LEVEL					

- **SFV: Store Final Value**
  0: The register is read/write.
  1: The register is read-only, to protect against further accidental writes.
- **FCD: Flash Calibration Done**
  Set to 1 when the CTRL field has been updated by the Flash fuses after a reset.
  0: The flash calibration will be redone after any reset.
  1: The flash calibration will not be redone after a BOD33 reset.
- **CTRL: BOD33 Control**

Table 8-8. Operation mode for BOD33

CTRL	Description
0x0	BOD33 is off
0x1	BOD33 is enabled and can reset the chip
0x2	BOD33 is enabled, but cannot reset the chip. Only interrupt will be sent to interrupt controller, if enabled in the IMR register.
0x3	Reserved

- **HYST: BOD33 Hysteresis**
  0: No hysteresis.
  1: Hysteresis on.

- **LEVEL: BOD33 Level**
  This field sets the triggering threshold of the BOD33. See Electrical Characteristics for actual voltage levels.
  Note that any change to the LEVEL field of the BOD33 register should be done with the BOD33 deactivated to avoid spurious reset or interrupt.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.13 5V BOD Control Register

Name: BOD50

Access Type: Read/Write

Offset: 0x0038

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SFV	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	CTRL

7	6	5	4	3	2	1	0
-	HYST	LEVEL					

- **SFV: Store Final Value**
  0: The register is read/write.
  1: The register is read-only, to protect against further accidental writes.
- **CTRL: BOD50 Control**
  0: BOD50 is off.
  1: BOD50 is enabled.
- **HYST: BOD50 Hysteresis**
  0: No hysteresis.
  1: Hysteresis on.
- **LEVEL: BOD50 Level**
  This field sets the triggering threshold of the BOD50. See Electrical Characteristics for actual voltage levels.
  Note that any change to the LEVEL field of the BOD50 register should be done with the BOD50 deactivated to avoid spurious reset or interrupt.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.14 Voltage Regulator Calibration Register

Name: VREGCR

Access Type: Read/Write

Offset: 0x003C

Reset Value: 0x00000000

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

- **SFV: Store Final Value**
  0: The register is read/write.
  1: The register is read-only, to protect against further accidental writes.
- **FCD: Flash Calibration Done**
  Set to 1 when the CALIB field has been updated by the Flash fuses after a reset.
  0: The flash calibration will be redone after any reset.
  1: The flash calibration will only be redone after a power-on reset.
- **CALIB: Calibration value**
  Calibration value for voltage reference of the 1.8V voltage regulator.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.15 Voltage Regulator Control Register

Name: VREGCTRL

Access Type: Read/Write

Offset: 0x0040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SFV	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	VREG33CTL		-	

- **SFV: Store Final Value**
  0: The register is read/write.
  1: The register is read-only, to protect against further accidental writes.
- **VREG33CTL: 3.3 Voltage Regulator Control**

Table 8-9. Operation mode of 3.3V Voltage Regulator

VREG33CTL	Description
0x0	3.3V Regulator is ON
0x1	Reserved
0x2	Reserved
0x3	3.3V Regulator is OFF

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.16 RCSYS Calibration Register

Name: RCCR

Access Type: Read/Write

Offset: 0x0044

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CALIB[9:8]	

7	6	5	4	3	2	1	0
CALIB[7:0]							

- **FCD: Flash Calibration Done**
  Set to 1 when CALIB field has been updated by the Flash fuses after a reset.
  0: The flash calibration will be redone after any reset.
  1: The flash calibration will only be redone after a power-on reset.
- **CALIB: Calibration Value**
  Calibration Value for the RC oscillator.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.17 8MHz / 1MHz RC Oscillator Control Register

Name: RCCR8

Access Type: Read/Write

Offset: 0x0048

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	FREQMODE	RCOSC8_EN

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
CALIB[7:0]							

- **FREQMODE: Frequency Mode**
  0: the RC8M RC oscillator will run at 8 MHz.
  1: the RC8M RC oscillator will run at 1 MHz.
- **RCOSC8_EN: RCOSC Enable**
  0: the RC8M RC oscillator is disabled.
  1: the RC8M RC oscillator is enabled.
- **FCD: Flash Calibration Done**
  Set to 1 when CALIB field has been updated by the Flash fuses after a reset.
  0: The flash calibration will be redone after any reset.
  1: The flash calibration will only be redone after a power-on reset.
- **CALIB: Calibration Value**
  Calibration Value for the RC8M RC oscillator.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.18 32KHz Oscillator Control Register

Name: OSCCTRL32

Access Type: Read/Write

Offset: 0x004C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	STARTUP		
15	14	13	12	11	10	9	8
-	-	-	-	-	-	MODE	
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OSC32EN

Note: This register is only reset by Power-On Reset.

- **STARTUP: Oscillator Startup Time**
  Select startup time for 32 KHz oscillator.

Table 8-10. Startup time for 32 KHz oscillator

STARTUP	Number of RC oscillator clock cycle	Approximative Equivalent time (RCSYS = 115 kHz)
0	0	0
1	128	1.1 ms
2	8192	72.3 ms
3	16384	143 ms
4	65536	570 ms
5	131072	1.1 s
6	262144	2.3 s
7	524288	4.6 s

- **MODE: Oscillator Mode**

Table 8-11. Operation mode for 32 KHz oscillator

MODE	Description
0	External clock connected to XIN32, XOUT32 can be used as I/O (no crystal)
1	2-pin crystal mode. Crystal is connected to XIN32/XOUT32
2	2-pin crystal and I-Current mode. Crystal is connected to XIN32/XOUT32
3	Reserved

- **OSC32EN: Enable the 32 KHz oscillator**
  0: 32 KHz Oscillator is disabled.
  1: 32 KHz Oscillator is enabled.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.19 120MHz RC Oscillator Control Register

Name: RC120MCR

Access Type: Read/Write

Offset: 0x0058

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	EN

- **EN: RC120M Enable**
  0: Clock is stopped.
  1: Clock is running.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.20 General Purpose Low-power Register 0/1

Name: GPLP0,1

Access Type: Read/Write

Offset: 0x005C,0x0060

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

These registers are general purpose 32-bit registers that are reset only by power-on-reset. Any other reset will keep the bits of these registers untouched.

Note that this registers are protected by a lock. To write to these registers the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

8.6.21 Generic Clock Control

Name: GCCTRL

Access Type: Read/Write

Offset: 0x0064-0x008C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
DIV							

15	14	13	12	11	10	9	8
				OSCSEL			

7	6	5	4	3	2	1	0
-	-	-	-	-	-	DIVEN	CEN

There is one GCCTRL register per generic clock in the device.

- **DIV: Division Factor**
- **OSCSEL: Oscillator Select**

Table 8-12. Generic Clock Source

OSCSEL	Clock	Description
0	RCSYS	System RC oscillator clock
1	OSC32K	Output clock from OSC32K
2	8MHz / 1MHz RCOSC	Output clock from RC8M
3	OSC0 out	Output clock from Oscillator 0
4	OSC1 out	Output clock from Oscillator 1
5	PLL0 out	Output from PLL 0
6	PLL1 out	Output from PLL 1
7	CPU clock	The clock the CPU runs on
8	HSB clock	High Speed Bus clock
9	PBA clock	Peripheral Bus A clock
10	PBB clock	Peripheral Bus B clock
11	PBC clock	Peripheral Bus C clock
12-15	Reserved	

- **DIVEN: Divide Enable**
  0: The generic clock equals the undivided source clock.
  1: The generic clock equals the source clock divided by 2*(DIV+1).
- **CEN: Clock Enable**
  0: Clock is stopped.
  1: Clock is running.

8.6.22 PLL Interface Version Register

Name: PLLVERSION

Access Type: Read-Only

Offset: 0x03C8

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.23 Oscillator Interface Version Register

Name: OSCVERSION

Access Type: Read-Only

Offset: 0x03CC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.24 1.8V BOD Interface Version Register

Name: BODVERSION

Access Type: Read-Only

Offset: 0x03D0

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.25 3.3V / 5V BOD Interface Version Register

Name: BODBVERSION

Access Type: Read-Only

Offset: 0x03D4

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.26 Voltage Regulator Interface Version Register

Name: VREGVERSION

Access Type: Read-Only

Offset: 0x03D8

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.27 RCSYS Interface Version Register

Name: RCCRVERSION

Access Type: Read-Only

Offset: 0x03DC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.28 8MHz / 1MHz RCOSC Interface Version Register

Name: RCCR8VERSION

Access Type: Read-Only

Offset: 0x03E0

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.29 32KHz Oscillator Interface Version Register

Name: OSC32VERSION

Access Type: Read-Only

Offset: 0x03E4

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.30 120MHz RC Oscillator Version Register

Name: RC120MVERSION

Access Type: Read-Only

Offset: 0x03F0

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.31 GPLP Version Register

Name: GPLPVERSION

Access Type: Read-Only

Offset: 0x03F4

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.32 Generic Clock Version Register

Name: GCLKVERSION

Access Type: Read-Only

Offset: 0x03F8

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.6.33 SCIF Version Register

Name: VERSION

Access Type: Read-Only

Offset: 0x03FC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:0]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

8.7 Module Configuration

The specific configuration for each SCIF instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 8-13. SCIF Clock Name

Module name	Clock Name	Description
SCIF	CLK_SCIF	Peripheral Bus clock from the PBA clock domain

Table 8-14. Register Reset Values

Register	Reset Value
PLLVERSION	0x00000100
OSCVERSION	0x00000101
BODVERSION	0x00000101
BODBVERSION	0x00000100
VREGVERSION	0x00000100
RCCRVERSION	0x00000100
RCCR8VERSION	0x00000100
OSC32VERSION	0x00000100
TSENSVERSION	0x00000100
RC120MIFAVERSION	0x00000100
GPLPVERSION	0x00000110
GCLKVERSION	0x00000100
VERSION	0x00000101

9. Asynchronous Timer (AST)

Rev: 2.0.0.1

9.1 Features

- **32-bit counter with 32-bit prescaler**
- **Clocked Source**
  - **System RC oscillator (RCSYS)**
  - **32KHz crystal oscillator (OSC32K)**
  - **PB clock**
  - **Generic clock (GCLK)**
  - **1KHz clock from 32KHz oscillator**
- **Optional calendar mode supported**
- **Digital prescaler tuning for increased accuracy**
- **Periodic interrupt(s) and peripheral event(s) supported**
- **Alarm interrupt(s) and peripheral event(s) supported**
  - **Optional clear on alarm**

9.2 Overview

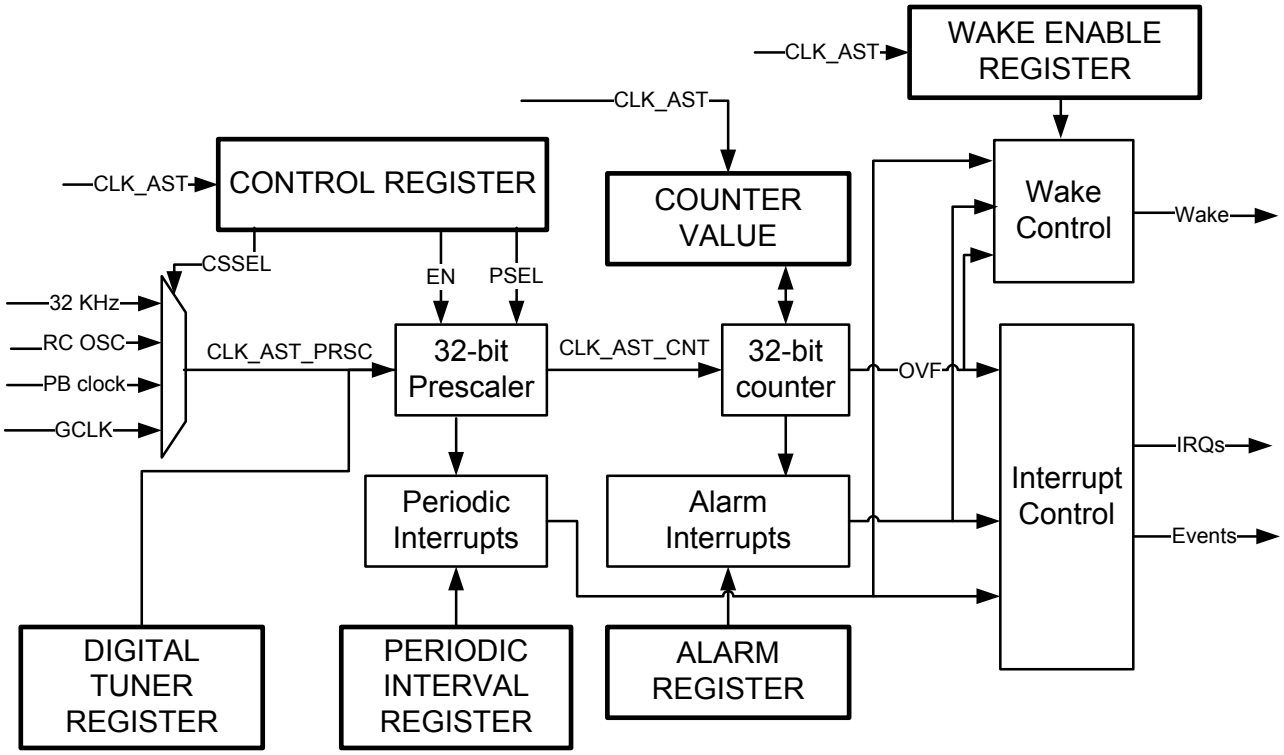
The Asynchronous Timer (AST) enables periodic interrupts and periodic peripheral events, as well as interrupts and peripheral events at a specified time in the future. The AST consists of a 32-bit prescaler which feeds a 32-bit up-counter. The prescaler can be clocked from four different clock sources, including the low-power 32KHz clock, which allows the AST to be used as a real-time timer with a maximum timeout of more than 100 years. Also, the PB clock or a generic clock can be used for high-speed operation, allowing the AST to be used as a general timer.

The AST can generate periodic interrupts and peripheral events from output from the prescaler, as well as alarm interrupts and peripheral events, which can trigger at any counter value. Additionally, the timer can trigger an overflow interrupt and peripheral event, and be reset on the occurrence of any alarm. This allows periodic interrupts and peripheral events at very long and accurate intervals.

The AST has been designed to meet the system tick and Real Time Clock requirements of most embedded operating systems.

9.3 Block Diagram

Figure 9-1. Asynchronous Timer block diagram



9.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

9.4.1 Power Management

When the AST is enabled, it will remain clocked as long as its selected clock source is running. It can also wake the CPU from the currently active sleep mode. Refer to the Power Manager chapter for details on the different sleep modes.

9.4.2 Clocks

The clock for the AST bus interface (CLK_AST) is generated by the Power Manager. This clock is turned on by default, and can be enabled and disabled in the Power Manager.

A number of clocks can be selected as source for the internal prescaler clock CLK_AST_PRSC. The prescaler, counter, and interrupt will function as long as this selected clock source is active. The selected clock must be enabled in the System Control Interface (SCIF).

The following clock sources are available:

- System RC oscillator (RCSYS). This oscillator is always enabled, except in some sleep modes. Please refer to the Electrical Characteristics chapter for the characteristic frequency of this oscillator.
- 32KHz crystal oscillator (OSC32K). This oscillator must be enabled before use.

- Peripheral Bus clock (PB clock). This is the clock of the peripheral bus the AST is connected to.
- Generic clock (GCLK). One of the generic clocks is connected to the AST. This clock must be enabled before use, and remains enabled in sleep modes when the PB clock is active.

9.4.3 Interrupt

The AST interrupt request lines are connected to the interrupt controller. Using the AST interrupts requires the interrupt controller to be programmed first.

9.4.4 Peripheral Events

The AST peripheral events are connected via the Peripheral Event System. Refer to the Peripheral Event System chapter for details.

9.4.5 Debug Operation

The AST prescaler and counter is frozen during debug operation, unless the Run In Debug bit in the Development Control Register is set and the bit corresponding to the AST is set in the Peripheral Debug Register (PDBG). Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual, and the OCD Module Configuration section, for details.

If the AST is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

9.5 Functional Description

9.5.1 Initialization

Before enabling the AST, the internal AST clock CLK_AST_PRSC must be enabled, following the procedure specified in Section 9.5.1.1. The Clock Source Select field in the Clock register (CLOCK.CSSEL) selects the source for this clock. The Clock Enable bit in the Clock register (CLOCK.CEN) enables the CLK_AST_PRSC.

When CLK_AST_PRSC is enabled, the AST can be enabled by writing a one to the Enable bit in the Control Register (CR.EN).

9.5.1.1 Enabling and disabling the AST clock

The Clock Source Selection field (CLOCK.CSSEL) and the Clock Enable bit (CLOCK.CEN) cannot be changed simultaneously. Special procedures must be followed for enabling and disabling the CLK_AST_PRSC and for changing the source for this clock.

To enable CLK_AST_PRSC:

- Write the selected value to CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero
- Write a one to CLOCK.CEN, without changing CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero

To disable the clock:

- Write a zero to CLOCK.CEN to disable the clock, without changing CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero

9.5.1.2 *Changing the source clock*

The CLK_AST_PRSC must be disabled before switching to another source clock. The Clock Busy bit in the Status Register (SR.CLKBUSY) indicates whether the clock is busy or not. This bit is set when the CEN bit in the CLOCK register is changed, and cleared when the CLOCK register can be changed.

To change the clock:

- Write a zero to CLOCK.CEN to disable the clock, without changing CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero
- Write the selected value to CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero
- Write a one to CLOCK.CEN to enable the clock, without changing CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero

9.5.2 Basic Operation

9.5.2.1 *Prescaler*

When the AST is enabled, the 32-bit prescaler will increment on the rising edge of CLK_AST_PRSC. The prescaler value cannot be read or written, but it can be reset by writing a one to the Prescaler Clear bit in the Control Register (CR.PCLR).

The Prescaler Select field in the Control Register (CR.PSEL) selects the prescaler bit PSEL as source clock for the counter (CLK_AST_CNT). This results in a counter frequency of:

$$f_{CNT} = \frac{f_{PRSC}}{2^{PSEL+1}}$$

where f_{PRSC} is the frequency of the internal prescaler clock CLK_AST_PRSC.

9.5.2.2 *Counter operation*

When enabled, the AST will increment on every 0-to-1 transition of the selected prescaler tapping. When the Calender bit in the Control Register (CR.CAL) is zero, the counter operates in counter mode. It will increment until it reaches the top value of 0xFFFFFFFF, and then wrap to 0x00000000. This sets the status bit Overflow in the Status Register (SR.OVF). Optionally, the counter can also be reset when a timer alarm occurs (see Section 9.5.3.2), which will also set the OVF bit.

The AST counter value can be read from or written to the Counter Value (CV) register. Note that due to synchronization, continuous reading of the CV register with the lowest prescaler setting will skip every third value. In addition, if CLK_AST_PRSC is as fast as, or faster than, the CLK_AST, the prescaler value must be 3 or higher to be able to read the CV without skipping values.

9.5.2.3 *Calendar operation*

When the CAL bit in the Control Register is one, the counter operates in calendar mode. Before this mode is enabled, the prescaler should be set up to give a pulse every second. The date and time can then be read from or written to the Calendar Value (CALV) register.

Time is reported as seconds, minutes, and hours according to the 24-hour clock format. Date is the numeral date of month (starting on 1). Month is the numeral month of the year (1 = January, 2 = February, etc). Year is a 6-bit field counting the offset from a software-defined leap year (e.g. 2000). The date is automatically compensated for leap years, assuming every year divisible by 4 is a leap year.

All peripheral events and interrupts work the same way in calendar mode as in counter mode. However, the Alarm Register (AR) must be written in time/date format for the alarm to trigger correctly.

9.5.3 Interrupts

The AST can generate five separate interrupt requests:

- OVF: OVF
- PER: PER0, PER1
- ALARM: ALARM0, ALARM1
- CLKREADY
- READY

This allows the user to allocate separate handlers and priorities to the different interrupt types.

The generation of the PER interrupt is described in Section 9.5.3.1., and the generation of the ALARM interrupt is described in Section 9.5.3.2. The OVF interrupt is generated when the counter overflows, or when the alarm value is reached, if the Clear on Alarm bit in the Control Register is one. The CLKREADY interrupt is generated when SR.CLKBUSY has a 1-to-0 transition, and indicates that the clock synchronization is completed. The READY interrupt is generated when SR.BUSY has a 1-to-0 transition, and indicates that the synchronization described in Section 9.5.7 is completed.

An interrupt request will be generated if the corresponding bit in the Interrupt Mask Register (IMR) is set. Bits in IMR are set by writing a one to the corresponding bit in the Interrupt Enable Register (IER), and cleared by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The interrupt request remains active until the corresponding bit in SR is cleared by writing a one to the corresponding bit in the Status Clear Register (SCR).

The AST interrupts can wake the CPU from any sleep mode where the source clock and the interrupt controller is active.

9.5.3.1 *Periodic interrupt*

The AST can generate periodic interrupts. If the PERn bit in the Interrupt Mask Register (IMR) is one, the AST will generate an interrupt request on the 0-to-1 transition of the selected bit in the

prescaler when the AST is enabled. The bit is selected by the Interval Select field in the corresponding Periodic Interval Register (PIRn.INSEL), resulting in a periodic interrupt frequency of

$$f_{PA} = \frac{f_{CS}}{2^{INSEL+1}}$$

where f_{CS} is the frequency of the selected clock source.

The corresponding PERn bit in the Status Register (SR) will be set when the selected bit in the prescaler has a 0-to-1 transition.

Because of synchronization, the transfer of the INSEL value will not happen immediately. When changing/setting the INSEL value, the user must make sure that the prescaler bit number INSEL will not have a 0-to-1 transition before the INSEL value is transferred to the register. In that case, the first periodic interrupt after the change will not be triggered.

9.5.3.2 *Alarm interrupt*

The AST can also generate alarm interrupts. If the ALARMn bit in IMR is one, the AST will generate an interrupt request when the counter value matches the selected alarm value, when the AST is enabled. The alarm value is selected by writing the value to the VALUE field in the corresponding Alarm Register (ARn.VALUE).

The corresponding ALARMn bit in SR will be set when the counter reaches the selected alarm value.

Because of synchronization, the transfer of the alarm value will not happen immediately. When changing/setting the alarm value, the user must make sure that the counter will not count the selected alarm value before the value is transferred to the register. In that case, the first alarm interrupt after the change will not be triggered.

If the Clear on Alarm bit in the Control Register (CR.CAn) is one, the corresponding alarm interrupt will clear the counter and set the OVF bit in the Status Register. This will generate an overflow interrupt if the OVF bit in IMR is set.

9.5.4 Peripheral events

The AST can generate a number of peripheral events:

- OVF
- PER0
- PER1
- ALARM0
- ALARM1

The PERn peripheral event(s) is generated the same way as the PER interrupt, as described in Section 9.5.3.1. The ALARMn peripheral event(s) is generated the same way as the ALARM interrupt, as described in Section 9.5.3.2. The OVF peripheral event is generated the same way as the OVF interrupt, as described in Section 9.5.3-

The peripheral event will be generated if the corresponding bit in the Event Mask (EVM) register is set. Bits in EVM register are set by writing a one to the corresponding bit in the Event Enable (EVE) register, and cleared by writing a one to the corresponding bit in the Event Disable (EVD) register.

9.5.5 AST wakeup

The AST can wake up the CPU directly, without the need to trigger an interrupt. A wakeup can be generated when the counter overflows, when the counter reaches the selected alarm value, or when the selected prescaler bit has a 0-to-1 transition. In this case, the CPU will continue executing from the instruction following the sleep instruction.

The AST wakeup is enabled by writing a one to the corresponding bit in the Wake Enable Register (WER). When the CPU wakes from sleep, the wake signal must be cleared by writing a one to the corresponding bit in SCR to clear the internal wake signal to the sleep controller. If the wake signal is not cleared after waking from sleep, the next sleep instruction will have not effect because the CPU will wake immediately after this sleep instruction.

The AST wakeup can wake the CPU from any sleep mode where the source clock is active. The AST wakeup can be configured independently of the interrupt masking.

9.5.6 Digital tuner

The digital tuner adds the possibility to compensate for a too slow or a too fast input clock. The ADD bit in the Digital Tuner Register (DTR.ADD) selects if the tuned frequency should be reduced or increased. If ADD is ‘0’, the prescaler frequency is reduced:

$$f_{TUNED} = f_0\left(1 - \frac{1}{roundup\left(\frac{256}{VALUE}\right) \cdot (2^{EXP}) + 1}\right)$$

Where f_{TUNED} is the tuned frequency, f_0 is the original prescaler frequency, and VALUE and EXP are the corresponding fields to be programmed in DTR. Note that DTR.EXP must be greater than zero. Frequency tuning is disabled by programming DTR.VALUE as zero.

If ADD is ‘1’, the prescaler frequency is increased:

$$f_{TUNED} = f_0\left(1 + \frac{1}{roundup\left(\frac{256}{VALUE}\right) \cdot (2^{EXP}) - 1}\right)$$

Note that these formulas to be within an error of 0.01%, it is recommended that the prescaler bit that is used as the clock for the counter (selected by CR.PSEL) or to trigger the periodic interrupt (selected by PIRn.INSEL) be bit 6 or higher.

9.5.7 Synchronization

As the prescaler and counter operate asynchronously from the user interface, the AST needs a few clock cycles to synchronize values written to the CR, CV, SCR, WER, EVE, EVD, PIRx, ARx and DTR registers. The Busy bit in the Status Register (SR.BUSY) indicates that the synchronization is ongoing. During this time, writes to these registers will be discarded.

Note that synchronization takes place also if the prescaler is clocked from CLK_AST.

9.6 User Interface

Table 9-1. AST Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Read/Write	0x00000000
0x04	Counter Value	CV	Read/Write	0x00000000
0x08	Status Register	SR	Read-only	0x00000000
0x0C	Status Clear Register	SCR	Write-only	0x00000000
0x10	Interrupt Enable Register	IER	Write-only	0x00000000
0x14	Interrupt Disable Register	IDR	Write-only	0x00000000
0x18	Interrupt Mask Register	IMR	Read-only	0x00000000
0x1C	Wake Enable Register	WER	Read/write	0x00000000
0x20	Alarm Register 0(2)	AR0	Read/Write	0x00000000
0x24	Alarm Register 1(2)	AR1	Read/Write	0x00000000
0x30	Periodic Interval Register 0(2)	PIR0	Read/Write	0x00000000
0x34	Periodic Interval Register 1(2)	PIR1	Read/Write	0x00000000
0x40	Clock Control Register	CLOCK	Read/Write	0x00000000
0x44	Digital Tuner Register	DTR	Read/Write	0x00000000
0x48	Event Enable	EVE	Write-only	0x00000000
0x4C	Event Disable	EVD	Write-only	0x00000000
0x50	Event Mask	EVM	Read-only	0x00000000
0x54	Calendar Value	CALV	Read/Write	0x00000000
0xF0	Parameter Register	PARAMETER	Read-only	-(1)
0xFC	Version Register	VERSION	Read-only	-(1)

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

2. The number of Alarm and Periodic Interval registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

9.6.1 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	PSEL				

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CA1	CA0

7	6	5	4	3	2	1	0
-	-	-	-		CAL	PCLR	EN

When the SR.BUSY bit is set, writes to this register will be discarded and this register will read as zero.

- **PSEL: Prescaler Select**
  Selects prescaler bit PSEL as source clock for the counter.
- **CAn: Clear on Alarm n**
  0: The corresponding alarm will not clear the counter.
  1: The corresponding alarm will clear the counter.
- **CAL: Calendar Mode**
  0: The AST operates in counter mode.
  1: The AST operates in calendar mode.
- **PCLR: Prescaler Clear**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the prescaler.
  This bit always reads as zero.
- **EN: Enable**
  0: The AST is disabled.
  1: The AST is enabled.

9.6.2 Counter Value

Name: CV

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
VALUE[31:24]							

23	22	21	20	19	18	17	16
VALUE[23:16]							

15	14	13	12	11	10	9	8
VALUE[15:8]							

7	6	5	4	3	2	1	0
VALUE[7:0]							

When the SR.BUSY bit is set, writes to this register will be discarded and this register will read as zero.

- **VALUE: AST Value**
  - The current value of the AST counter.

9.6.3 Status Register

Name: SR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	CLKREADY	CLKBUSY	-	-	READY	BUSY
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

- **CLKREADY: Clock ready**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the SR.CLKBUSY bit has a 1-to-0 transition.
- **CLKBUSY: Clock busy**
  - 0: The clock is ready and can be changed.
  - 1: CLOCK.CEN has been written and the clock is busy.
- **READY: AST ready**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the SR.BUSY bit has a 1-to-0 transition.
- **BUSY: AST busy**
  0: The AST accepts writes to CR, CV, SCR, WER, EVE, EVD, ARn, and PIRn, and DTR.
  1: The AST is busy and will discard writes to CR, CV, SCR, WER, EVE, EVD, ARn, and PIRn, and DTR.
- **PERn: Periodic n**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the selected bit in the prescaler has a 0-to-1 transition.
- **ALARMn: Alarm n**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the counter reaches the selected alarm value.
- **OVF: Overflow**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when an overflow has occurred.

9.6.4 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	CLKREADY	-	-	-	READY	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

When the SR.BUSY bit is set, writes to this register will be discarded.
Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

9.6.5 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	CLKREADY	-	-	-	READY	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

9.6.6 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	CLKREADY	-	-	-	READY	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

9.6.7 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	CLKREADY	-	-	-	READY	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

9.6.8 Wake Enable Register

Name: WER

Access Type: Read/Write

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

When the SR.BUSY bit is set writes to this register will be discarded and this register will read as zero.

This register enables the wakeup signal from the AST.

- **PERn: Periodic n**
  0: The CPU will not wake up from sleep mode when the selected bit in the prescaler has a 0-to-1 transition.
  1: The CPU will wake up from sleep mode when the selected bit in the prescaler has a 0-to-1 transition.
- **ALARMn: Alarm n**
  0: The CPU will not wake up from sleep mode when the counter reaches the selected alarm value.
  1: The CPU will wake up from sleep mode when the counter reaches the selected alarm value.
- **OVF: Overflow**
  0: A counter overflow will not wake up the CPU from sleep mode.
  1: A counter overflow will wake up the CPU from sleep mode.

9.6.9 Alarm Register

Name: AR0/1

Access Type: Read/Write

Offset: 0x20/0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
VALUE[31:24]							

23	22	21	20	19	18	17	16
VALUE[23:16]							

15	14	13	12	11	10	9	8
VALUE[15:8]							

7	6	5	4	3	2	1	0
VALUE[7:0]							

When the SR.BUSY bit is set writes to this register will be discarded and this register will read as zero.

- **VALUE: Alarm Value**
  - When the counter reaches this value, an alarm is generated.

9.6.10 Periodic Interval Register

Name: PIR0/1

Access Type: Read/Write

Offset: 0x30/0x34

Reset Value: 0x00000000

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

When the SR.BUSY bit is set writes to this register will be discarded and this register will read as zero.

- **INSEL: Interval Select**
  - The PERn bit in SR will be set when the INSEL bit in the prescaler has a 0-to-1 transition.

9.6.11 Clock Control Register

Name: CLOCK

Access Type: Read/Write

Offset: 0x40

Reset Value: 0x00000000

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

When writing to this register, follow the sequence in Section 9.5.1.

- **CSSEL: Clock Source Selection**
  This field defines the clock source CLK_AST_PRSC for the prescaler:

Table 9-2. Clock Source Selection

CSSEL	Clock Source
0	System RC oscillator (RCSYS)
1	32KHz oscillator (OSC32K)
2	PB clock
3	Generic clock (GCLK)

- **CEN: Clock Enable**
  0: CLK_AST_PRSC is disabled.
  1: CLK_AST_PRSC is enabled.

9.6.12 Digital Tuner Register

Name: DTR

Access Type: Read/Write

Offset: 0x44

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
VALUE							

7	6	5	4	3	2	1	0
-	-	ADD	EXP				

When the SR.BUSY bit is set writes to this register will be discarded and this register will read as zero.

- **VALUE:**
  0: The frequency is unchanged.
  1-255: The frequency will be adjusted according to the formula below.

- **ADD:**
  0: The resulting frequency is $f0\left(1 - \frac{1}{roundup\left(\frac{256}{VALUE}\right) \cdot 2^{(EXP)} + 1}\right)$ for $VALUE > 0$.

  1: The resulting frequency is $f0\left(1 + \frac{1}{roundup\left(\frac{256}{VALUE}\right) \cdot 2^{(EXP)} - 1}\right)$ for $VALUE > 0$.

- **EXP:**
  The frequency will be adjusted according to the formula above.

9.6.13 Event Enable Register

Name: EVE

Access Type: Write-only

Offset: 0x48

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

When the SR.BUSY bit is set writes to this register will be discarded and this register will read as zero.
Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in EVM.

9.6.14 Event Disable Register

Name: EVD

Access Type: Write-only

Offset: 0x4C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

When the SR.BUSY bit is set writes to this register will be discarded and this register will read as zero.
Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in EVM.

9.6.15 Event Mask Register

Name: EVM

Access Type: Read-only

Offset: 0x50

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OVF

0: The corresponding peripheral event is disabled.
1: The corresponding peripheral event is enabled.
This bit is cleared when the corresponding bit in EVD is written to one.
This bit is set when the corresponding bit in EVE is written to one.

9.6.16 Calendar Value

Name: CALV

Access Type: Read/Write

Offset: 0x54

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
YEAR						MONTH[3:2]	

23	22	21	20	19	18	17	16
MONTH[1:0]		DAY					HOUR[4]

15	14	13	12	11	10	9	8
HOUR[3:0]				MIN[5:2]			

7	6	5	4	3	2	1	0
MIN[1:0]		SEC					

When the SR.BUSY bit is set writes to this register will be discarded and this register will read as zero.

- **YEAR: Year**
  - Current year. The year is considered a leap year if YEAR[1:0] = 0.
- **MONTH: Month**
  - 1 = January
  - 2 = February
  - ...
  - 12 = December
- **DAY: Day**
  - Day of month, starting with 1.
- **HOUR: Hour**
  - Hour of day, in 24-hour clock format.
  - Legal values are 0 through 23.
- **MIN: Minute**
  - Minutes, 0 through 59.
- **SEC: Second**
  - Seconds, 0 through 59.

9.6.17 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0xF0

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	PER1VALUE				

23	22	21	20	19	18	17	16
-	-	-	PER0VALUE				

15	14	13	12	11	10	9	8
PIR1WA	PIR0WA	-	NUMPIR			NUMAR	

7	6	5	4	3	2	1	0
-	DTEXPVALUE					DTEXPWA	DT

This register gives the configuration used in the specific device. Also refer to the Module Configuration section.

- **DT: Digital Tuner**
  - 0: Digital tuner off
  - 1: Digital tuner on
- **DTREXPWA: Digital Tuner Exponent Writeable**
  - 0: Digital tuner exponent is a constant value. Writes to EXP bitfield in DTR will be discarded.
  - 1: Digital tuner exponent is chosen by writing to EXP bitfield in DTR
- **DTREXPVALUE: Digital Tuner Exponent Value**
  - Digital tuner exponent value if DT_EXP_WA is zero
- **NUMAR: Number of Alarm Comparators**
  0: Zero alarm comparators.
  1: One alarm comparator.
  2: Two alarm comparators.
- **NUMPIR: Number of Periodic Comparators**
  0: One periodic comparator.
  1: Two periodic comparator.
- **PIRnWA: Periodic Interval n Writeable**
  0: Periodic interval n prescaler tapping is a constant value. Writes to INSEL field in PIRn register will be discarded.
  1: Periodic interval n prescaler tapping is chosen by writing to INSEL field in PIRn register.
- **PERnVALUE: Periodic Interval n Value**
  Periodic interval prescaler n tapping if PIRnWA is zero.

9.6.18 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: 0x00000300

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  - Reserved. No functionality associated.
- **VERSION: Version number**
  - Version of the module. No functionality associated.

9.7 Module configuration

The specific configuration for each AST instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks according to the table in the System Bus Clock Connections section.

Table 9-3. Module configuration

Feature	AST
Number of timer alarms	2
Number of periodic alarms	2
Digital tuner	On

Table 9-4. Module clock name

Module name	Clock name	Description
AST	CLK_AST	Peripheral Bus clock from the PBA clock domain
	GCLK	The generic clock used for the AST is GCLK2
	PB clock	Peripheral Bus clock from the PBA clock domain

Table 9-5. Register Reset Values

Module name	Clock name
VERSION	0x00000200
PARAMETER	0x0000D203

10. Watchdog Timer (WDT)

Rev: 4.1.0.0

10.1 Features

- **Watchdog Timer counter with 32-bit counter**
- **Timing window watchdog**
- **Clocked from system RC oscillator or the 32 KHz crystal oscillator**
- **Configuration lock**
- **WDT may be enabled at reset by a fuse**

10.2 Overview

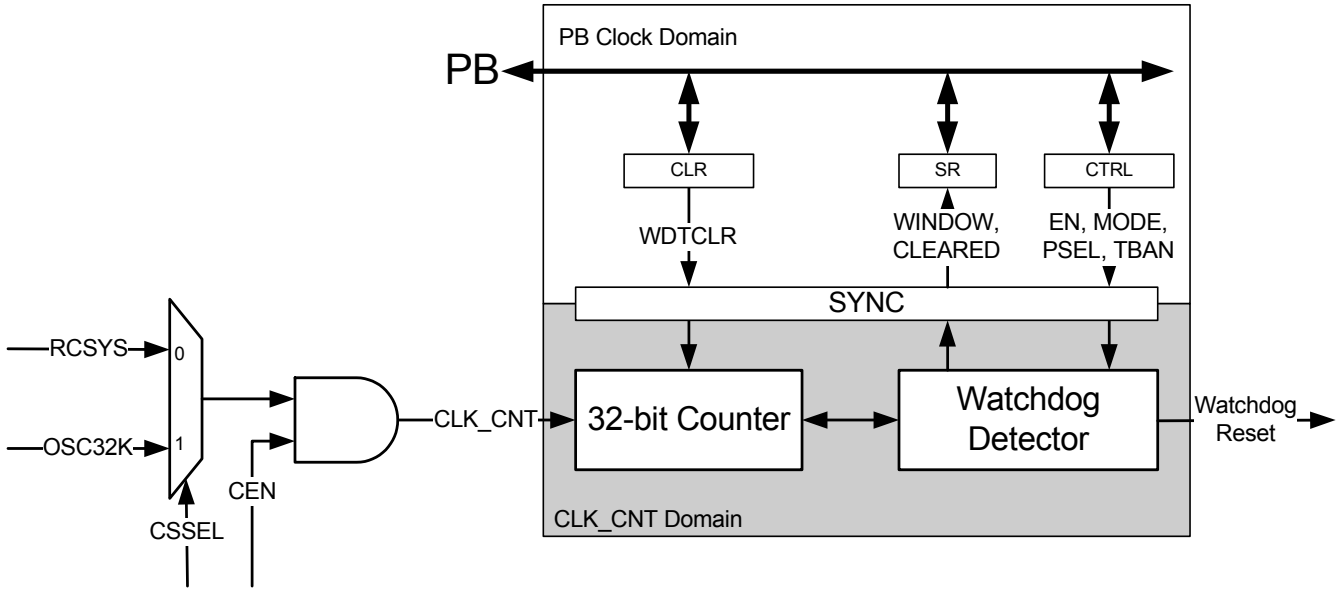
The Watchdog Timer (WDT) will reset the device unless it is periodically serviced by the software. This allows the device to recover from a condition that has caused the system to be unstable.

The WDT has an internal counter clocked from the system RC oscillator or the 32 KHz crystal oscillator.

The WDT counter must be periodically cleared by software to avoid a watchdog reset. If the WDT timer is not cleared correctly, the device will reset and start executing from the boot vector.

10.3 Block Diagram

Figure 10-1. WDT Block Diagram



10.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

10.4.1 Power Management

When the WDT is enabled, the WDT remains clocked in all sleep modes. It is not possible to enter sleep modes where the source clock of CLK_CNT is stopped. Attempting to do so will result in the chip entering the lowest sleep mode where the source clock is running, leaving the WDT operational. Please refer to the Power Manager chapter for details about sleep modes.

After a watchdog reset the WDT bit in the Reset Cause Register (RCAUSE) in the Power Manager will be set.

10.4.2 Clocks

The clock for the WDT bus interface (CLK_WDT) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the WDT before disabling the clock, to avoid freezing the WDT in an undefined state.

There are two possible clock sources for the Watchdog Timer (CLK_CNT):

- System RC oscillator (RCSYS): This oscillator is always enabled when selected as clock source for the WDT. Please refer to the Power Manager chapter for details about the RCSYS and sleep modes. Please refer to the Electrical Characteristics chapter for the characteristic frequency of this oscillator.
- 32 KHz crystal oscillator (OSC32K): This oscillator has to be enabled in the System Control Interface before using it as clock source for the WDT. The WDT will not be able to detect if this clock is stopped.

10.4.3 Debug Operation

The WDT counter is frozen during debug operation, unless the Run In Debug bit in the Development Control Register is set and the bit corresponding to the WDT is set in the Peripheral Debug Register (PDBG). Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual, and the OCD Module Configuration section, for details. If the WDT counter is not frozen during debug operation it will need periodically clearing to avoid a watchdog reset.

10.4.4 Fuses

The WDT can be enabled at reset. This is controlled by the WDTAUTO fuse, see Section 10.5.4 for details. Please refer to the Fuse Settings section in the Flash Controller chapter for details about WDTAUTO and how to program the fuses.

10.5 Functional Description

10.5.1 Basic Mode

10.5.1.1 WDT Control Register Access

To avoid accidental disabling of the watchdog, the Control Register (CTRL) must be written twice, first with the KEY field set to 0x55, then 0xAA without changing the other bits. Failure to do so will cause the write operation to be ignored, and the value in the CTRL Register will not be changed.

10.5.1.2 Changing CLK_CNT Clock Source

After any reset, except for watchdog reset, CLK_CNT will be enabled with the RCSYS as source.

To change the clock for the WDT the following steps need to be taken. Note that the WDT should always be disabled before changing the CLK_CNT source:

1. Write a zero to the Clock Enable (CEN) bit in the CTRL Register, leaving the other bits as they are in the CTRL Register. This will stop CLK_CNT.

2. Read back the CTRL Register until the CEN bit reads zero. The clock has now been stopped.

3. Modify the Clock Source Select (CSSEL) bit in the CTRL Register with your new clock selection and write it to the CTRL Register.

4. Write a one to the CEN bit, leaving the other bits as they are in the CTRL Register. This will enable the clock.

5. Read back the CTRL Register until the CEN bit reads one. The clock has now been enabled.

10.5.1.3 Configuring the WDT

If the MODE bit in the CTRL Register is zero, the WDT is in basic mode. The Time Out Prescale Select (PSEL) field in the CTRL Register selects the WDT timeout period:

$$T_{timeout} = T_{psel} = 2^{(PSEL+1)} / f_{clk\_cnt}$$

10.5.1.4 Enabling the WDT

To enable the WDT write a one to the Enable (EN) bit in the CTRL Register. Due to internal synchronization, it will take some time for the CTRL.EN bit to read back as one.

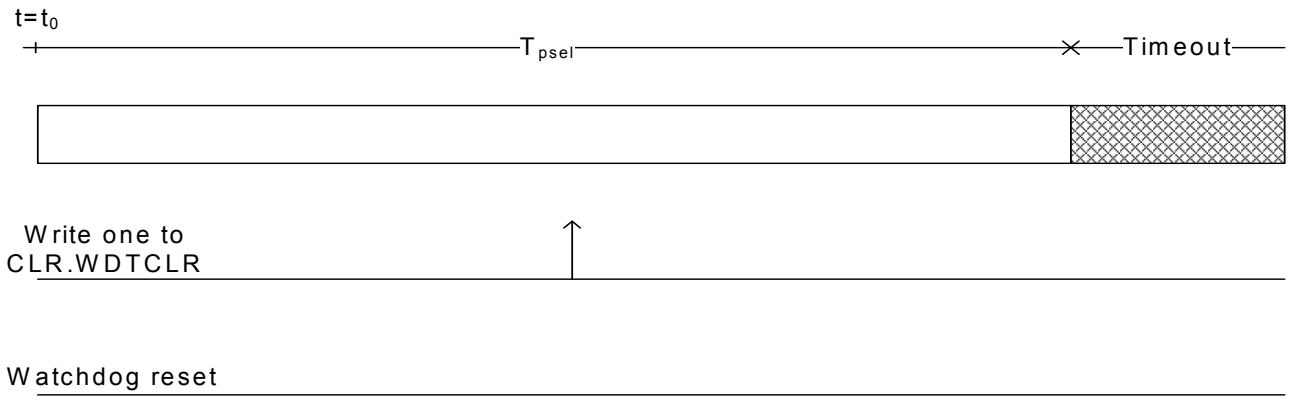
10.5.1.5 Clearing the WDT Counter

The WDT counter is cleared by writing a one to the Watchdog Clear (WDTCLR) bit in the Clear (CLR) Register, at any correct write to the CTRL Register, or when the counter reaches $T_{timeout}$ and the chip is reset. In basic mode the CLR.WDTCLR can be written at any time when the WDT Counter Cleared (CLEARED) bit in the Status Register (SR) is one. Due to internal synchronization, clearing the WDT counter takes some time. The SR.CLEARED bit is cleared when writing to CLR.WDTCLR bit and set when the clearing is done. Any write to the CLR.WDTCLR bit while SR.CLEARED is zero will not clear the counter.

Writing to the CLR.WDTCLR bit has to be done in a particular sequence to be valid. The CLR Register must be written twice, first with the KEY field set to 0x55 and WDTCLR set to one, then a second write with the KEY set to 0xAA without changing the WDTCLR bit. Writing to the CLR Register without the correct sequence has no effect.

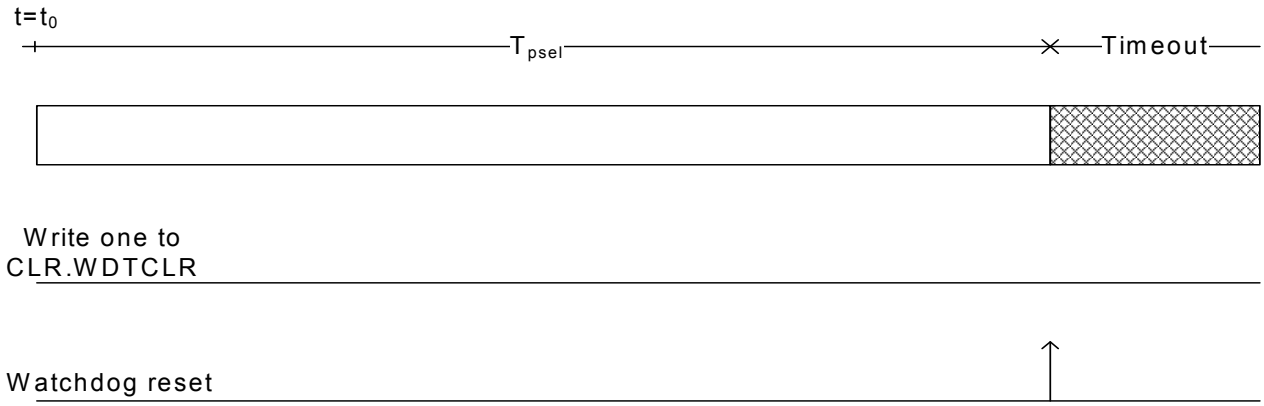
If the WDT counter is periodically cleared within T_{psel} no watchdog reset will be issued, see Figure 10-2 on page 164.

Figure 10-2. Basic Mode WDT Timing Diagram, normal operation.



If the WDT counter is not cleared within T_{psel} a watchdog reset will be issued at the end of T_{psel}, see Figure 10-3 on page 164.

Figure 10-3. Basic Mode WDT Timing Diagram, no clear within T_{psel}.



10.5.1.6 Watchdog Reset

A watchdog reset will result in a reset and the code will start executing from the boot vector, please refer to the Power Manager chapter for details. If the Disable After Reset (DAR) bit in the CTRL Register is zero, the WDT counter will restart counting from zero when the watchdog reset is released.

If the CTRL.DAR bit is one the WDT will be disabled after a watchdog reset. Only the CTRL.EN bit will be changed after the watchdog reset. However, if WDTAUTO fuse is configured to enable the WDT after a watchdog reset, and the CTRL.FCD bit is zero, writing a one to the CTRL.DAR bit will have no effect.

10.5.2 Window Mode

The window mode can protect against tight loops of runaway code. This is obtained by adding a ban period to timeout period. During the ban period clearing the WDT counter is not allowed.

If the WDT Mode (MODE) bit in the CTRL Register is one, the WDT is in window mode. Note that the CTRL.MODE bit can only be changed when the WDT is disabled (CTRL.EN=0).

The PSEL and Time Ban Prescale Select (TBAN) fields in the CTRL Register selects the WDT timeout period

$$T_{timeout} = T_{tban} + T_{psel} = (2^{(TBAN+1)} + 2^{(PSEL+1)}) / f_{clk\_cnt}$$

where T_{tban} sets the time period when clearing the WDT counter by writing to the CLR.WDTCLR bit is not allowed. Doing so will result in a watchdog reset, the device will receive a reset and the code will start executing form the boot vector, see Figure 10-5 on page 165. The WDT counter will be cleared.

Writing a one to the CLR.WDTCLR bit within the T_{psel} period will clear the WDT counter and the counter starts counting from zero ($t=t_0$), entering T_{tban}, see Figure 10-4 on page 165.

If the value in the CTRL Register is changed, the WDT counter will be cleared without a watchdog reset, regardless of if the value in the WDT counter and the TBAN value.

If the WDT counter reaches $T_{timeout}$, the counter will be cleared, the device will receive a reset and the code will start executing form the boot vector.

Figure 10-4. Window Mode WDT Timing Diagram

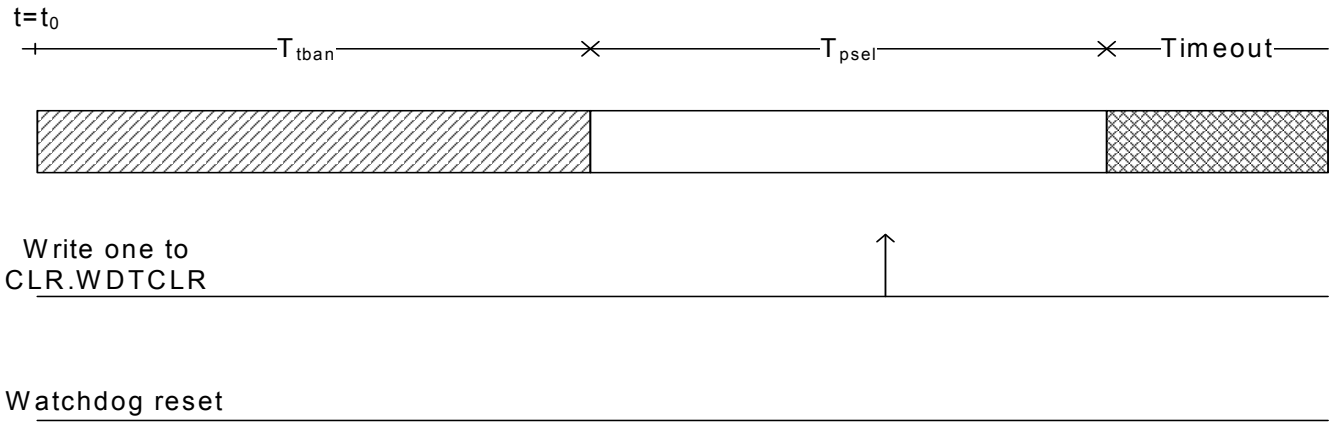
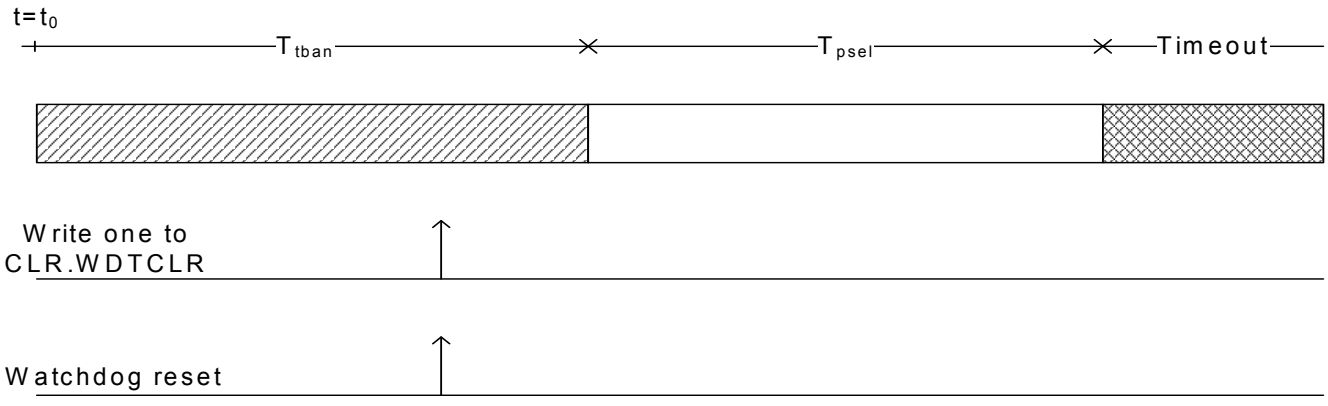


Figure 10-5. Window Mode WDT Timing Diagram, clearing within T_{tban}, resulting in watchdog reset.



10.5.3 Disabling the WDT

The WDT is disabled by writing a zero to the CTRL.EN bit. When disabling the WDT no other bits in the CTRL Register should be changed until the CTRL.EN bit reads back as zero. If the CTRL.CEN bit is written to zero, the CTRL.EN bit will never read back as zero if changing the value from one to zero.

10.5.4 Flash Calibration

The WDT can be enabled at reset. This is controlled by the WDTAUTO fuse. The WDT will be set in basic mode, RCSYS is set as source for CLK_CNT, and PSEL will be set to a value giving T_{psel} above 100 ms. Please refer to the Fuse Settings chapter for details about WDTAUTO and how to program the fuses.

If the Flash Calibration Done (FCD) bit in the CTRL Register is zero at a watchdog reset the flash calibration will be redone, and the CTRL.FCD bit will be set when the calibration is done. If CTRL.FCD is one at a watchdog reset, the configuration of the WDT will not be changed during flash calibration. After any other reset the flash calibration will always be done, and the CTRL.FCD bit will be set when the calibration is done.

10.5.5 Special Considerations

Care must be taken when selecting the PSEL/TBAN values so that the timeout period is greater than the startup time of the chip. Otherwise a watchdog reset will reset the chip before any code has been run. This can also be avoided by writing the CTRL.DAR bit to one when configuring the WDT.

If the Store Final Value (SFV) bit in the CTRL Register is one, the CTRL Register is locked for further write accesses. All writes to the CTRL Register will be ignored. Once the CTRL Register is locked, it can only be unlocked by a reset (e.g. POR, OCD, and WDT).

The CTRL.MODE bit can only be changed when the WDT is disabled (CTRL.EN=0).

10.6 User Interface

Table 10-1. WDT Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Control Register	CTRL	Read/Write	0x00010080
0x004	Clear Register	CLR	Write-only	0x00000000
0x008	Status Register	SR	Read-only	0x00000003
0x3FC	Version Register	VERSION	Read-only	-(1)

Note: 1. The reset value for this register is device specific. Please refer to the Module Configuration section at the end of this chapter.

10.6.1 Control Register

Name: CTRL

Access Type: Read/Write

Offset: 0x000

Reset Value: 0x00010080

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	TBAN					CSSEL	CEN

15	14	13	12	11	10	9	8
-	-	-	PSEL				

7	6	5	4	3	2	1	0
FCD	-	-	-	SFV	MODE	DAR	EN

- **KEY**
  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to be effective. This field always reads as zero.
- **TBAN: Time Ban Prescale Select**
  Counter bit TBAN is used as watchdog “banned” time frame. In this time frame clearing the WDT timer is forbidden, otherwise a watchdog reset is generated and the WDT timer is cleared.
- **CSSEL: Clock Source Select**
  0: Select the system RC oscillator (RCSYS) as clock source.
  1: Select the 32KHz crystal oscillator (OSC32K) as clock source.
- **CEN: Clock Enable**
  0: The WDT clock is disabled.
  1: The WDT clock is enabled.
- **PSEL: Time Out Prescale Select**
  Counter bit PSEL is used as watchdog timeout period.
- **FCD: Flash Calibration Done**
  This bit is set after any reset.
  0: The flash calibration will be redone after a watchdog reset.
  1: The flash calibration will not be redone after a watchdog reset.
- **SFV: WDT Control Register Store Final Value**
  0: WDT Control Register is not locked.
  1: WDT Control Register is locked.
  Once locked, the Control Register can not be re-written, only a reset unlocks the SFV bit.
- **MODE: WDT Mode**
  0: The WDT is in basic mode, only PSEL time is used.
  1: The WDT is in window mode. Total timeout period is now TBAN+PSEL.
  Writing to this bit when the WDT is enabled has no effect.

- **DAR: WDT Disable After Reset**
  0: After a watchdog reset, the WDT will still be enabled.
  1: After a watchdog reset, the WDT will be disabled.
- **EN: WDT Enable**
  0: WDT is disabled.
  1: WDT is enabled.
  After writing to this bit the read back value will not change until the WDT is enabled/disabled. This due to internal synchronization.

10.6.2 Clear Register

Name: CLR

Access Type: Write-only

Offset: 0x004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	WDTCLR

When the Watchdog Timer is enabled, this Register must be periodically written within the window time frame or within the watchdog timeout period, to prevent a watchdog reset.

- **KEY**
  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to be effective.
- **WDTCLR: Watchdog Clear**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the WDT counter.

10.6.3 Status Register

Name: SR

Access Type: Read-only

Offset: 0x008

Reset Value: 0x00000003

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	CLEARED	WINDOW

- **CLEARED: WDT Counter Cleared**
  This bit is cleared when writing a one to the CLR.WDTCLR bit.
  This bit is set when clearing the WDT counter is done.
- **WINDOW: Within Window**
  This bit is cleared when the WDT counter is inside the TBAN period.
  This bit is set when the WDT counter is inside the PSEL period.

10.6.4 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x3FC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

10.7 Module Configuration

The specific configuration for each WDT instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 10-2. Module clock name

Module name	Clock name	Description
WDT	CLK_WDT	Peripheral Bus clock from the PBA clock domain

Table 10-3. Register Reset Values

Register	Reset Value
VERSION	0x00000410

11. Interrupt Controller (INTC)

Rev: 1.0.2.5

11.1 Features

- **Autovectored low latency interrupt service with programmable priority**
  - **4 priority levels for regular, maskable interrupts**
  - **One Non-Maskable Interrupt**
- **Up to 64 groups of interrupts with up to 32 interrupt requests in each group**

11.2 Overview

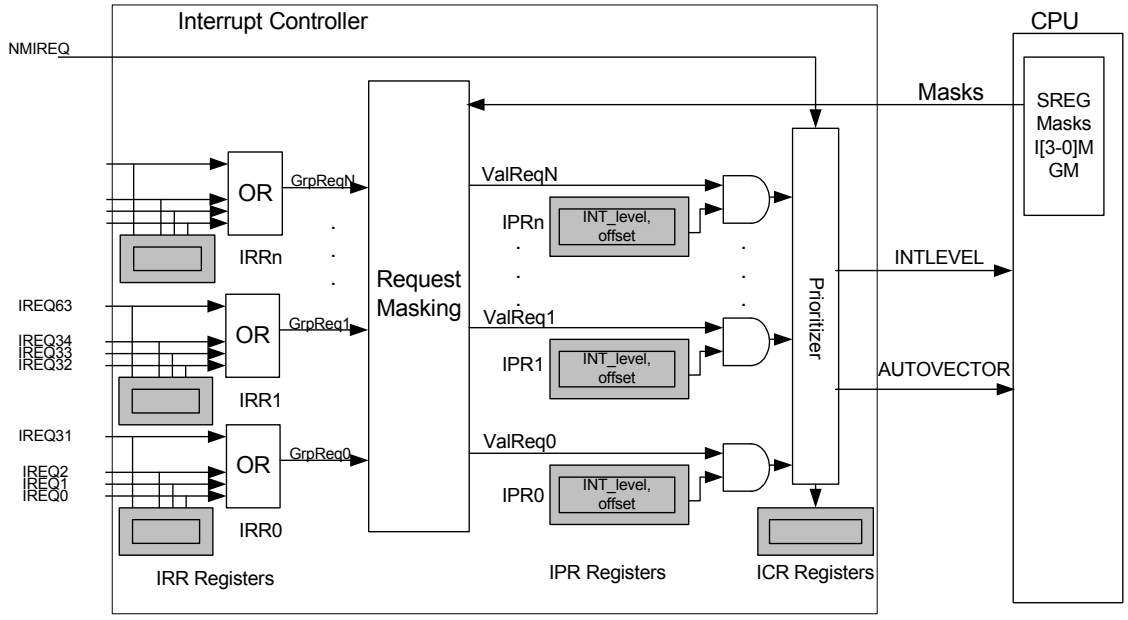
The INTC collects interrupt requests from the peripherals, prioritizes them, and delivers an interrupt request and an autovector to the CPU. The AVR32 architecture supports 4 priority levels for regular, maskable interrupts, and a Non-Maskable Interrupt (NMI).

The INTC supports up to 64 groups of interrupts. Each group can have up to 32 interrupt request lines, these lines are connected to the peripherals. Each group has an Interrupt Priority Register (IPR) and an Interrupt Request Register (IRR). The IPRs are used to assign a priority level and an autovector to each group, and the IRRs are used to identify the active interrupt request within each group. If a group has only one interrupt request line, an active interrupt group uniquely identifies the active interrupt request line, and the corresponding IRR is not needed. The INTC also provides one Interrupt Cause Register (ICR) per priority level. These registers identify the group that has a pending interrupt of the corresponding priority level. If several groups have a pending interrupt of the same level, the group with the lowest number takes priority.

11.3 Block Diagram

Figure 11-1 gives an overview of the INTC. The grey boxes represent registers that can be accessed via the user interface. The interrupt requests from the peripherals (IREQn) and the NMI are input on the left side of the figure. Signals to and from the CPU are on the right side of the figure.

Figure 11-1. INTC Block Diagram



11.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

11.4.1 Power Management

If the CPU enters a sleep mode that disables CLK_SYNC, the INTC will stop functioning and resume operation after the system wakes up from sleep mode.

11.4.2 Clocks

The clock for the INTC bus interface (CLK_INTC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager.

The INTC sampling logic runs on a clock which is stopped in any of the sleep modes where the system RC oscillator is not running. This clock is referred to as CLK_SYNC. This clock is enabled at reset, and only turned off in sleep modes where the system RC oscillator is stopped.

11.4.3 Debug Operation

When an external debugger forces the CPU into debug mode, the INTC continues normal operation.

11.5 Functional Description

All of the incoming interrupt requests (IREQs) are sampled into the corresponding Interrupt Request Register (IRR). The IRRs must be accessed to identify which IREQ within a group that is active. If several IREQs within the same group are active, the interrupt service routine must prioritize between them. All of the input lines in each group are logically ORed together to form the GrpReqN lines, indicating if there is a pending interrupt in the corresponding group.

The Request Masking hardware maps each of the GrpReq lines to a priority level from INT0 to INT3 by associating each group with the Interrupt Level (INTLEVEL) field in the corresponding

Interrupt Priority Register (IPR). The GrpReq inputs are then masked by the mask bits from the CPU status register. Any interrupt group that has a pending interrupt of a priority level that is not masked by the CPU status register, gets its corresponding ValReq line asserted.

Masking of the interrupt requests is done based on five interrupt mask bits of the CPU status register, namely Interrupt Level 3 Mask (I3M) to Interrupt Level 0 Mask (I0M), and Global Interrupt Mask (GM). An interrupt request is masked if either the GM or the corresponding interrupt level mask bit is set.

The Prioritizer hardware uses the ValReq lines and the INTLEVEL field in the IPRs to select the pending interrupt of the highest priority. If an NMI interrupt request is pending, it automatically gets the highest priority of any pending interrupt. If several interrupt groups of the highest pending interrupt level have pending interrupts, the interrupt group with the lowest number is selected.

The INTLEVEL and handler autovector offset (AUTOVECTOR) of the selected interrupt are transmitted to the CPU for interrupt handling and context switching. The CPU does not need to know which interrupt is requesting handling, but only the level and the offset of the handler address. The IRR registers contain the interrupt request lines of the groups and can be read via user interface registers for checking which interrupts of the group are actually active.

The delay through the INTC from the peripheral interrupt request is set until the interrupt request to the CPU is set is three cycles of CLK_SYNC.

11.5.1 Non-Maskable Interrupts

A NMI request has priority over all other interrupt requests. NMI has a dedicated exception vector address defined by the AVR32 architecture, so AUTOVECTOR is undefined when INTLEVEL indicates that an NMI is pending.

11.5.2 CPU Response

When the CPU receives an interrupt request it checks if any other exceptions are pending. If no exceptions of higher priority are pending, interrupt handling is initiated. When initiating interrupt handling, the corresponding interrupt mask bit is set automatically for this and lower levels in status register. E.g, if an interrupt of level 3 is approved for handling, the interrupt mask bits I3M, I2M, I1M, and I0M are set in status register. If an interrupt of level 1 is approved, the masking bits I1M and I0M are set in status register. The handler address is calculated by logical OR of the AUTOVECTOR to the CPU system register Exception Vector Base Address (EVBA). The CPU will then jump to the calculated address and start executing the interrupt handler.

Setting the interrupt mask bits prevents the interrupts from the same and lower levels to be passed through the interrupt controller. Setting of the same level mask bit prevents also multiple requests of the same interrupt to happen.

It is the responsibility of the handler software to clear the interrupt request that caused the interrupt before returning from the interrupt handler. If the conditions that caused the interrupt are not cleared, the interrupt request remains active.

11.5.3 Clearing an Interrupt Request

Clearing of the interrupt request is done by writing to registers in the corresponding peripheral module, which then clears the corresponding NMIREQ/IREQ signal.

The recommended way of clearing an interrupt request is a store operation to the controlling peripheral register, followed by a dummy load operation from the same register. This causes a

pipeline stall, which prevents the interrupt from accidentally re-triggering in case the handler is exited and the interrupt mask is cleared before the interrupt request is cleared.

11.6 User Interface

Table 11-1. INTC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Interrupt Priority Register 0	IPR0	Read/Write	0x00000000
0x004	Interrupt Priority Register 1	IPR1	Read/Write	0x00000000
...	...	...	...	...
0x0FC	Interrupt Priority Register 63	IPR63	Read/Write	0x00000000
0x100	Interrupt Request Register 0	IRR0	Read-only	N/A
0x104	Interrupt Request Register 1	IRR1	Read-only	N/A
...	...	...	...	...
0x1FC	Interrupt Request Register 63	IRR63	Read-only	N/A
0x200	Interrupt Cause Register 3	ICR3	Read-only	N/A
0x204	Interrupt Cause Register 2	ICR2	Read-only	N/A
0x208	Interrupt Cause Register 1	ICR1	Read-only	N/A
0x20C	Interrupt Cause Register 0	ICR0	Read-only	N/A

11.6.1 Interrupt Priority Registers

Name: IPR0...IPR63

Access Type: Read/Write

Offset: 0x000 - 0x0FC

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
INTLEVEL[1:0]		-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	AUTOVECTOR[13:8]					

7	6	5	4	3	2	1	0
AUTOVECTOR[7:0]							

- **INTLEVEL: Interrupt Level**
  Indicates the EVBA-relative offset of the interrupt handler of the corresponding group:
  00: INT0: Lowest priority
  01: INT1
  10: INT2
  11: INT3: Highest priority
- **AUTOVECTOR: Autovector Address**
  Handler offset is used to give the address of the interrupt handler. The least significant bit should be written to zero to give halfword alignment.

11.6.2 Interrupt Request Registers

Name: IRR0...IRR63

Access Type: Read-only

Offset: 0x0FF - 0x1FC

Reset Value: N/A

31	30	29	28	27	26	25	24
IRR[32*x+31]	IRR[32*x+30]	IRR[32*x+29]	IRR[32*x+28]	IRR[32*x+27]	IRR[32*x+26]	IRR[32*x+25]	IRR[32*x+24]
23	22	21	20	19	18	17	16
IRR[32*x+23]	IRR[32*x+22]	IRR[32*x+21]	IRR[32*x+20]	IRR[32*x+19]	IRR[32*x+18]	IRR[32*x+17]	IRR[32*x+16]
15	14	13	12	11	10	9	8
IRR[32*x+15]	IRR[32*x+14]	IRR[32*x+13]	IRR[32*x+12]	IRR[32*x+11]	IRR[32*x+10]	IRR[32*x+9]	IRR[32*x+8]
7	6	5	4	3	2	1	0
IRR[32*x+7]	IRR[32*x+6]	IRR[32*x+5]	IRR[32*x+4]	IRR[32*x+3]	IRR[32*x+2]	IRR[32*x+1]	IRR[32*x+0]

- **IRR: Interrupt Request line**

  This bit is cleared when no interrupt request is pending on this input request line.

  This bit is set when an interrupt request is pending on this input request line.

  The are 64 IRRs, one for each group. Each IRR has 32 bits, one for each possible interrupt request, for a total of 2048 possible input lines. The IRRs are read by the software interrupt handler in order to determine which interrupt request is pending. The IRRs are sampled continuously, and are read-only.

11.6.3 Interrupt Cause Registers

Name: ICR0...ICR3

Access Type: Read-only

Offset: 0x200 - 0x20C

Reset Value: N/A

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	CAUSE					

- **CAUSE: Interrupt Group Causing Interrupt of Priority n**
  ICRn identifies the group with the highest priority that has a pending interrupt of level n. This value is only defined when at least one interrupt of level n is pending.

11.7 Interrupt Request Signal Map

The various modules may output Interrupt request signals. These signals are routed to the Interrupt Controller (INTC), described in a later chapter. The Interrupt Controller supports up to 64 groups of interrupt requests. Each group can have up to 32 interrupt request signals. All interrupt signals in the same group share the same autovector address and priority level. Refer to the documentation for the individual submodules for a description of the semantics of the different interrupt requests.

The interrupt request signals are connected to the INTC as follows.

Table 11-2. Interrupt Request Signal Map

Group	Line	Module	Signal
0	0	AVR32UC3 CPU	SYSBLOCK COMPARE
1	0	AVR32UC3 CPU	OCD DCEMU DIRTY
	1	AVR32UC3 CPU	OCD DCCPU READ
2	0	Secure Access Unit	SAU
3	0	Peripheral DMA Controller	PDCA 0
	1	Peripheral DMA Controller	PDCA 1
	2	Peripheral DMA Controller	PDCA 2
	3	Peripheral DMA Controller	PDCA 3
4	0	Peripheral DMA Controller	PDCA 4
	1	Peripheral DMA Controller	PDCA 5
	2	Peripheral DMA Controller	PDCA 6
	3	Peripheral DMA Controller	PDCA 7
5	0	Peripheral DMA Controller	PDCA 8
	1	Peripheral DMA Controller	PDCA 9
	2	Peripheral DMA Controller	PDCA 10
	3	Peripheral DMA Controller	PDCA 11
6	0	Peripheral DMA Controller	PDCA 12
	1	Peripheral DMA Controller	PDCA 13
	2	Peripheral DMA Controller	PDCA 14
	3	Peripheral DMA Controller	PDCA 15
7	0	Memory DMA	MDMA
8	0	USB 2.0 OTG Interface	USBC

Table 11-2. Interrupt Request Signal Map

9	0	Control Area Network interface	CANIF BOFF 0
	1	Control Area Network interface	CANIF ERROR 0
	2	Control Area Network interface	CANIF RXOK 0
	3	Control Area Network interface	CANIF TXOK 0
	4	Control Area Network interface	CANIF WAKEUP 0
	5	Control Area Network interface	CANIF BOFF 1
	6	Control Area Network interface	CANIF ERROR 1
	7	Control Area Network interface	CANIF RXOK 1
	8	Control Area Network interface	CANIF TXOK 1
	9	Control Area Network interface	CANIF WAKEUP 1
10	0	Flash Controller	HFLASHC
11	0	SDRAM Controller	SDRAMC
12	0	Power Manager	PM
13	0	System Control Interface	SCIF
14	0	Asynchronous Timer	AST ALARM
	1	Asynchronous Timer	AST CLKREADY
	2	Asynchronous Timer	AST OVF
	3	Asynchronous Timer	AST PER
	4	Asynchronous Timer	AST READY
15	0	External Interrupt Controller	EIC 1
	1	External Interrupt Controller	EIC 2
	2	External Interrupt Controller	EIC 3
	3	External Interrupt Controller	EIC 4
16	0	External Interrupt Controller	EIC 5
	1	External Interrupt Controller	EIC 6
	2	External Interrupt Controller	EIC 7
	3	External Interrupt Controller	EIC 8
17	0	Frequency Meter	FREQM

Table 11-2. Interrupt Request Signal Map

18	0	General Purpose Input/Output Controller	GPIO 0
	1	General Purpose Input/Output Controller	GPIO 1
	2	General Purpose Input/Output Controller	GPIO 2
	3	General Purpose Input/Output Controller	GPIO 3
	4	General Purpose Input/Output Controller	GPIO 4
	5	General Purpose Input/Output Controller	GPIO 5
	6	General Purpose Input/Output Controller	GPIO 6
	7	General Purpose Input/Output Controller	GPIO 7
	8	General Purpose Input/Output Controller	GPIO 8
	9	General Purpose Input/Output Controller	GPIO 9
	10	General Purpose Input/Output Controller	GPIO 10
	11	General Purpose Input/Output Controller	GPIO 11
	12	General Purpose Input/Output Controller	GPIO 12
	13	General Purpose Input/Output Controller	GPIO 13
	14	General Purpose Input/Output Controller	GPIO 14
	15	General Purpose Input/Output Controller	GPIO 15
19	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART0
20	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART1
21	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART2
22	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART3
23	0	Serial Peripheral Interface	SPI0
24	0	Serial Peripheral Interface	SPI1
25	0	Two-wire Master Interface	TWIM0
26	0	Two-wire Master Interface	TWIM1
27	0	Two-wire Slave Interface	TWIS0
28	0	Two-wire Slave Interface	TWIS1
29	0	Inter-IC Sound (I2S) Controller	IISC
30	0	Pulse Width Modulation Controller	PWM
31	0	Quadrature Decoder	QDEC0
32	0	Quadrature Decoder	QDEC1
33	0	Timer/Counter	TC00
	1	Timer/Counter	TC01
	2	Timer/Counter	TC02

Table 11-2. Interrupt Request Signal Map

34	0	Timer/Counter	TC10
	1	Timer/Counter	TC11
	2	Timer/Counter	TC12
35	0	Peripheral Event Controller	PEVC TR
	1	Peripheral Event Controller	PEVC OV
36	0	ADC controller interface with Touch Screen functionality	ADCIFA SEQ0
	1	ADC controller interface with Touch Screen functionality	ADCIFA SEQ1
	2	ADC controller interface with Touch Screen functionality	ADCIFA SUTD
	3	ADC controller interface with Touch Screen functionality	ADCIFA WINDOW
	4	ADC controller interface with Touch Screen functionality	ADCIFA AWAKEUP
	5	ADC controller interface with Touch Screen functionality	ADCIFA PENDET
37	0	Analog Comparators Interface	ACIFA0
38	0	Analog Comparators Interface	ACIFA1
39	0	DAC interface	DACIFB0 CHB UNDERRUN
	1	DAC interface	DACIFB0 CHB OVERRUN
	2	DAC interface	DACIFB0 CHB DATA EMPTY
	3	DAC interface	DACIFB0 CHA UNDERRUN
	4	DAC interface	DACIFB0 CHA OVERRUN
	5	DAC interface	DACIFB0 CHA DATA EMPTY
40	0	DAC interface	DACIFB1 CHA DATA EMPTY
	1	DAC interface	DACIFB1 CHA OVERRUN
	2	DAC interface	DACIFB1 CHA UNDERRUN
	3	DAC interface	DACIFB1 CHB DATA EMPTY
	4	DAC interface	DACIFB1 CHB OVERRUN
	5	DAC interface	DACIFB1 CHB UNDERRUN

Table 11-2. Interrupt Request Signal Map

41	0	aWire	AW
42	0	Ethernet MAC	MACB
44	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART4
45	0	Two-wire Master Interface	TWIM2
46	0	Two-wire Slave Interface	TWIS2

12. External Interrupt Controller (EIC)

Rev: 3.0.2.0

12.1 Features

- **Dedicated interrupt request for each interrupt**
- **Individually maskable interrupts**
- **Interrupt on rising or falling edge**
- **Interrupt on high or low level**
- **Asynchronous interrupts for sleep modes without clock**
- **Filtering of interrupt lines**
- **Non-Maskable NMI interrupt**

12.2 Overview

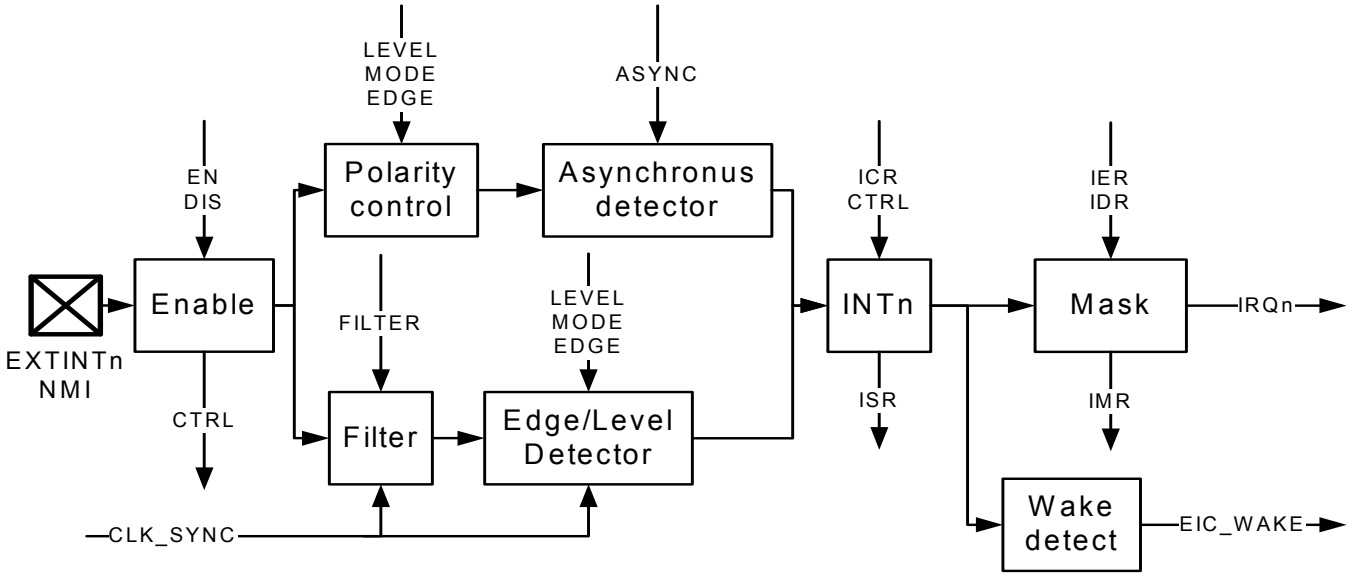
The External Interrupt Controller (EIC) allows pins to be configured as external interrupts. Each external interrupt has its own interrupt request and can be individually masked. Each external interrupt can generate an interrupt on rising or falling edge, or high or low level. Every interrupt input has a configurable filter to remove spikes from the interrupt source. Every interrupt pin can also be configured to be asynchronous in order to wake up the part from sleep modes where the CLK_SYNC clock has been disabled.

A Non-Maskable Interrupt (NMI) is also supported. This has the same properties as the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other interrupt mode.

The EIC can wake up the part from sleep modes without triggering an interrupt. In this mode, code execution starts from the instruction following the sleep instruction.

12.3 Block Diagram

Figure 12-1. EIC Block Diagram



12.4 I/O Lines Description

Table 12-1. I/O Lines Description

Pin Name	Pin Description	Type
NMI	Non-Maskable Interrupt	Input
EXTINTn	External Interrupt	Input

12.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

12.5.1 I/O Lines

The external interrupt pins (EXTINTn and NMI) may be multiplexed with I/O Controller lines. The programmer must first program the I/O Controller to assign the desired EIC pins to their peripheral function. If I/O lines of the EIC are not used by the application, they can be used for other purposes by the I/O Controller.

It is only required to enable the EIC inputs actually in use. If an application requires two external interrupts, then only two I/O lines will be assigned to EIC inputs.

12.5.2 Power Management

All interrupts are available in all sleep modes as long as the EIC module is powered. However, in sleep modes where CLK_SYNC is stopped, the interrupt must be configured to asynchronous mode.

12.5.3 Clocks

The clock for the EIC bus interface (CLK_EIC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager.

The filter and synchronous edge/level detector runs on a clock which is stopped in any of the sleep modes where the system RC oscillator (RCSYS) is not running. This clock is referred to as CLK_SYNC.

12.5.4 Interrupts

The external interrupt request lines are connected to the interrupt controller. Using the external interrupts requires the interrupt controller to be programmed first.

Using the Non-Maskable Interrupt does not require the interrupt controller to be programmed.

12.5.5 Debug Operation

When an external debugger forces the CPU into debug mode, the EIC continues normal operation. If the EIC is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

12.6 Functional Description

12.6.1 External Interrupts

The external interrupts are not enabled by default, allowing the proper interrupt vectors to be set up by the CPU before the interrupts are enabled.

Each external interrupt INTn can be configured to produce an interrupt on rising or falling edge, or high or low level. External interrupts are configured by the MODE, EDGE, and LEVEL registers. Each interrupt has a bit INTn in each of these registers. Writing a zero to the INTn bit in the MODE register enables edge triggered interrupts, while writing a one to the bit enables level triggered interrupts.

If INTn is configured as an edge triggered interrupt, writing a zero to the INTn bit in the EDGE register will cause the interrupt to be triggered on a falling edge on EXTINTn, while writing a one to the bit will cause the interrupt to be triggered on a rising edge on EXTINTn.

If INTn is configured as a level triggered interrupt, writing a zero to the INTn bit in the LEVEL register will cause the interrupt to be triggered on a low level on EXTINTn, while writing a one to the bit will cause the interrupt to be triggered on a high level on EXTINTn.

Each interrupt has a corresponding bit in each of the interrupt control and status registers. Writing a one to the INTn bit in the Interrupt Enable Register (IER) enables the external interrupt from pin EXTINTn to propagate from the EIC to the interrupt controller, while writing a one to INTn bit in the Interrupt Disable Register (IDR) disables this propagation. The Interrupt Mask Register (IMR) can be read to check which interrupts are enabled. When an interrupt triggers, the corresponding bit in the Interrupt Status Register (ISR) will be set. This bit remains set until a one is written to the corresponding bit in the Interrupt Clear Register (ICR) or the interrupt is disabled.

Writing a one to the INTn bit in the Enable Register (EN) enables the external interrupt on pin EXTINTn, while writing a one to INTn bit in the Disable Register (DIS) disables the external interrupt. The Control Register (CTRL) can be read to check which interrupts are enabled. If a bit in the CTRL register is set, but the corresponding bit in IMR is not set, an interrupt will not propagate to the interrupt controller. However, the corresponding bit in ISR will be set, and EIC_WAKE will be set. Note that an external interrupt should not be enabled before it has been configured correctly.

If the CTRL.INTn bit is zero, the corresponding bit in ISR will always be zero. Disabling an external interrupt by writing a one to the DIS.INTn bit will clear the corresponding bit in ISR.

Please refer to the Module Configuration section for the number of external interrupts.

12.6.2 Synchronization and Filtering of External Interrupts

In synchronous mode the pin value of the EXTINTn pin is synchronized to CLK_SYNC, so spikes shorter than one CLK_SYNC cycle are not guaranteed to produce an interrupt. The synchronization of the EXTINTn to CLK_SYNC will delay the propagation of the interrupt to the interrupt controller by two cycles of CLK_SYNC, see Figure 12-2 and Figure 12-3 for examples (FILTER off).

It is also possible to apply a filter on EXTINTn by writing a one to the INTn bit in the FILTER register. This filter is a majority voter, if the condition for an interrupt is true for more than one of the latest three cycles of CLK_SYNC the interrupt will be set. This will additionally delay the propagation of the interrupt to the interrupt controller by one or two cycles of CLK_SYNC, see Figure 12-2 and Figure 12-3 for examples (FILTER on).

Figure 12-2. Timing Diagram, Synchronous Interrupts, High Level or Rising Edge

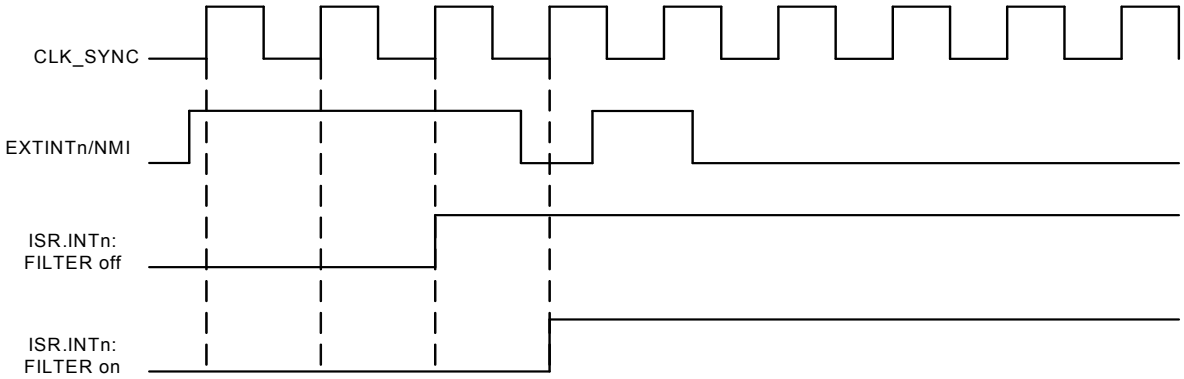
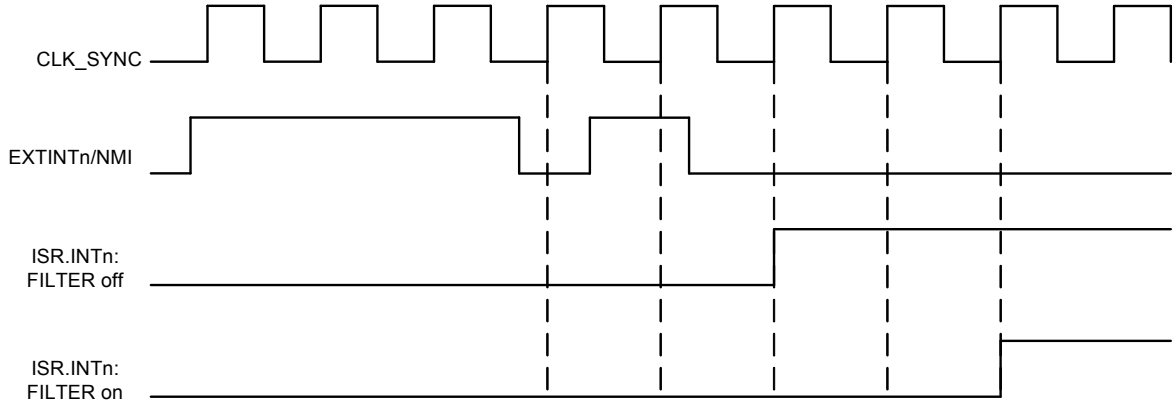


Figure 12-3. Timing Diagram, Synchronous Interrupts, Low Level or Falling Edge



12.6.3 Non-Maskable Interrupt

The NMI supports the same features as the external interrupts, and is accessed through the same registers. The description in Section 12.6.1 should be followed, accessing the NMI bit instead of the INTn bits.

The NMI is non-maskable within the CPU in the sense that it can interrupt any other execution mode. Still, as for the other external interrupts, the actual NMI input can be enabled and disabled by accessing the registers in the EIC.

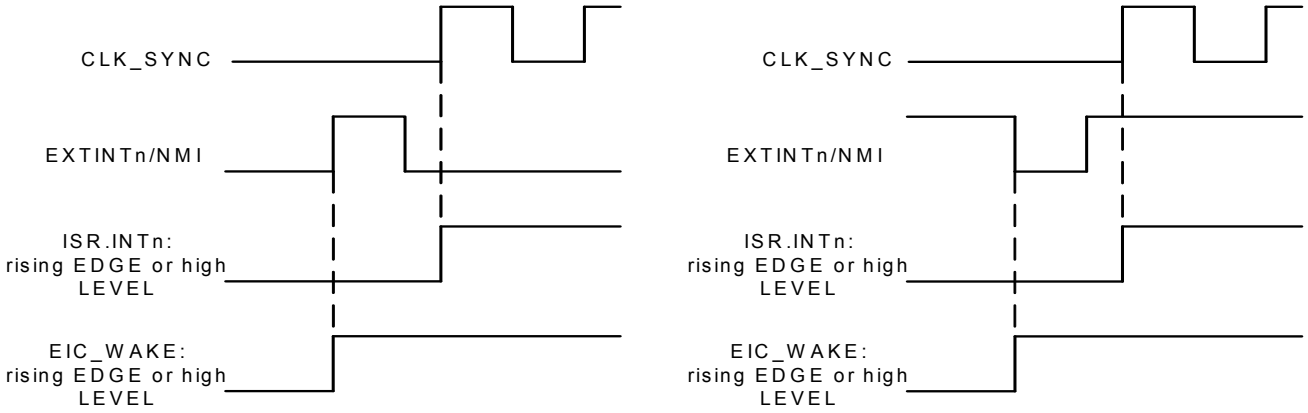
12.6.4 Asynchronous Interrupts

Each external interrupt can be made asynchronous by writing a one to INTn in the ASYNC register. This will route the interrupt signal through the asynchronous path of the module. All edge interrupts will be interpreted as level interrupts and the filter is disabled. If an interrupt is configured as edge triggered interrupt in asynchronous mode, a zero in EDGE.INTn will be interpreted as low level, and a one in EDGE.INTn will be interpreted as high level.

EIC_WAKE will be set immediately after the source triggers the interrupt, while the corresponding bit in ISR and the interrupt to the interrupt controller will be set on the next rising edge of CLK_SYNC. Please refer to Figure 12-4 on page 191 for details.

When CLK_SYNC is stopped only asynchronous interrupts remain active, and any short spike on this interrupt will wake up the device. EIC_WAKE will restart CLK_SYNC and ISR will be updated on the first rising edge of CLK_SYNC.

Figure 12-4. Timing Diagram, Asynchronous Interrupts



12.6.5 Wakeup

The external interrupts can be used to wake up the part from sleep modes. The wakeup can be interpreted in two ways. If the corresponding bit in IMR is one, then the execution starts at the interrupt handler for this interrupt. If the bit in IMR is zero, then the execution starts from the next instruction after the sleep instruction.

12.7 User Interface

Table 12-2. EIC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Interrupt Enable Register	IER	Write-only	0x00000000
0x004	Interrupt Disable Register	IDR	Write-only	0x00000000
0x008	Interrupt Mask Register	IMR	Read-only	0x00000000
0x00C	Interrupt Status Register	ISR	Read-only	0x00000000
0x010	Interrupt Clear Register	ICR	Write-only	0x00000000
0x014	Mode Register	MODE	Read/Write	0x00000000
0x018	Edge Register	EDGE	Read/Write	0x00000000
0x01C	Level Register	LEVEL	Read/Write	0x00000000
0x020	Filter Register	FILTER	Read/Write	0x00000000
0x024	Test Register	TEST	Read/Write	0x00000000
0x028	Asynchronous Register	ASYNC	Read/Write	0x00000000
0x030	Enable Register	EN	Write-only	0x00000000
0x034	Disable Register	DIS	Write-only	0x00000000
0x038	Control Register	CTRL	Read-only	0x00000000
0x3FC	Version Register	VERSION	Read-only	- (1)

Note: 1. The reset value is device specific. Please refer to the Module Configuration section at the end of this chapter.

12.7.1 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x000

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the corresponding bit in IMR.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Wrting a one to this bit will set the corresponding bit in IMR.

12.7.2 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in IMR.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in IMR.

12.7.3 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x008

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  0: The corresponding interrupt is disabled.
  1: The corresponding interrupt is enabled.
  This bit is cleared when the corresponding bit in IDR is written to one.
  This bit is set when the corresponding bit in IER is written to one.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is disabled.
  1: The Non-Maskable Interrupt is enabled.
  This bit is cleared when the corresponding bit in IDR is written to one.
  This bit is set when the corresponding bit in IER is written to one.

12.7.4 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x00C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  0: An interrupt event has not occurred.
  1: An interrupt event has occurred.
  This bit is cleared by writing a one to the corresponding bit in ICR.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  0: An interrupt event has not occurred.
  1: An interrupt event has occurred.
  This bit is cleared by writing a one to the corresponding bit in ICR.

12.7.5 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x010

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in ISR.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in ISR.

12.7.6 Mode Register

Name: MODE

Access Type: Read/Write

Offset: 0x014

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  0: The external interrupt is edge triggered.
  1: The external interrupt is level triggered.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is edge triggered.
  1: The Non-Maskable Interrupt is level triggered.

12.7.7 Edge Register

Name: EDGE

Access Type: Read/Write

Offset: 0x018

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  0: The external interrupt triggers on falling edge.
  1: The external interrupt triggers on rising edge.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt triggers on falling edge.
  1: The Non-Maskable Interrupt triggers on rising edge.

12.7.8 Level Register

Name: LEVEL

Access Type: Read/Write

Offset: 0x01C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  0: The external interrupt triggers on low level.
  1: The external interrupt triggers on high level.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt triggers on low level.
  1: The Non-Maskable Interrupt triggers on high level.

12.7.9 Filter Register

Name: FILTER

Access Type: Read/Write

Offset: 0x020

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  0: The external interrupt is not filtered.
  1: The external interrupt is filtered.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is not filtered.
  1: The Non-Maskable Interrupt is filtered.

12.7.10 Test Register

Name: TEST

Access Type: Read/Write

Offset: 0x024

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
TESTEN	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **TESTEN: Test Enable**
  0: This bit disables external interrupt test mode.
  1: This bit enables external interrupt test mode.
- **INTn: External Interrupt n**
  Writing a zero to this bit will set the input value to INTn to zero, if test mode is enabled.
  Writing a one to this bit will set the input value to INTn to one, if test mode is enabled.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit will set the input value to NMI to zero, if test mode is enabled.
  Writing a one to this bit will set the input value to NMI to one, if test mode is enabled.
  If TESTEN is 1, the value written to this bit will be the value to the interrupt detector and the value on the pad will be ignored.

12.7.11 Asynchronous Register

Name: ASYNC

Access Type: Read/Write

Offset: 0x028

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  0: The external interrupt is synchronized to CLK_SYNC.
  1: The external interrupt is asynchronous.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is synchronized to CLK_SYNC.
  1: The Non-Maskable Interrupt is asynchronous.

12.7.12 Enable Register

Name: EN

Access Type: Write-only

Offset: 0x030

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will enable the corresponding external interrupt.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will enable the Non-Maskable Interrupt.

12.7.13 Disable Register

Name: DIS

Access Type: Write-only

Offset: 0x034

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will disable the corresponding external interrupt.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will disable the Non-Maskable Interrupt.

12.7.14 Control Register

Name: CTRL

Access Type: Read-only

Offset: 0x038

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	INT30	INT29	INT28	INT27	INT26	INT25	INT24
23	22	21	20	19	18	17	16
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	NMI

- **INTn: External Interrupt n**
  0: The corresponding external interrupt is disabled.
  1: The corresponding external interrupt is enabled.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  0: The Non-Maskable Interrupt is disabled.
  1: The Non-Maskable Interrupt is enabled.

12.7.15 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x3FC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VERSION: Version number**
  Version number of the module. No functionality associated.

12.8 Module Configuration

The specific configuration for each EIC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 12-3. Module Configuration

Feature	EIC
Number of external interrupts, including NMI	9

Table 12-4. Module Clock Name

Module name	Clock Name	Description
EIC	CLK_EIC	Peripheral Bus clock from the PBA clock domain

Table 12-5. Register Reset Values

Register	Reset Value
VERSION	0x00000302

13. Frequency Meter (FREQM)

Rev: 3.1.0.1

13.1 Features

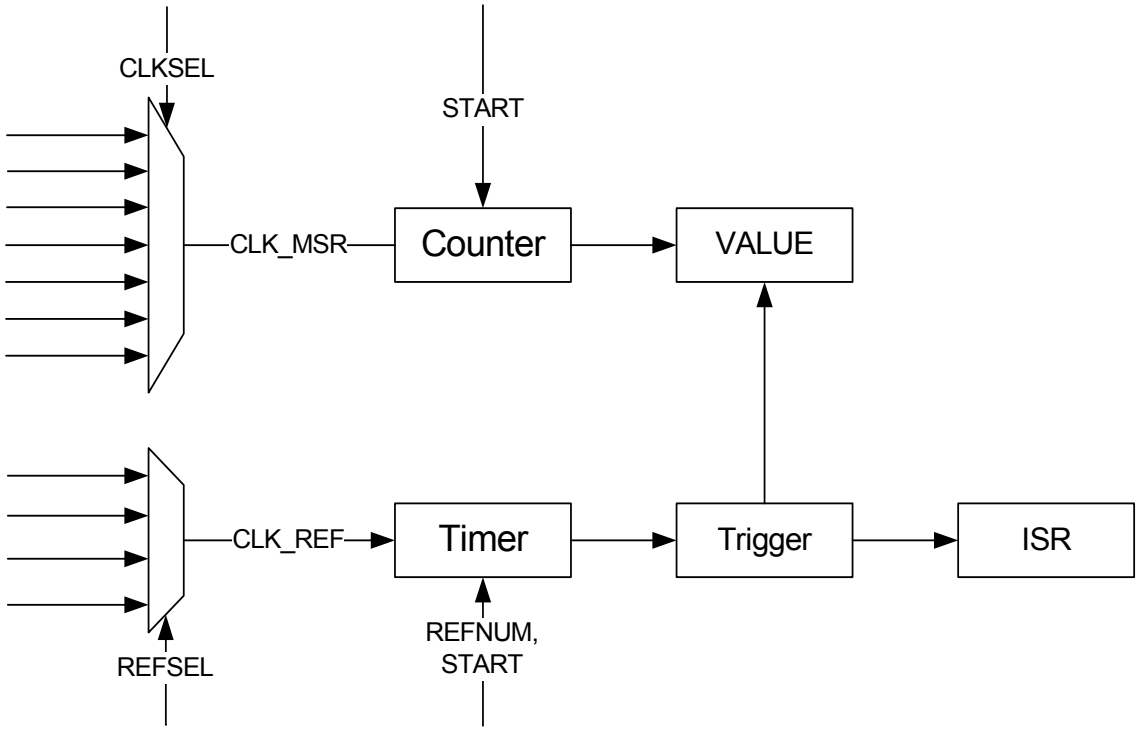
- **Accurately measures a clock frequency**
- **Selectable reference clock**
- **A selectable clock can be measured**
- **Ratio can be measured with 24-bit accuracy**

13.2 Overview

The Frequency Meter (FREQM) can be used to accurately measure the frequency of a clock by comparing it to a known reference clock.

13.3 Block Diagram

Figure 13-1. Frequency Meter Block Diagram



13.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

13.4.1 Power Management

The device can enter a sleep mode while a measurement is ongoing. However, make sure that neither CLK_MSR nor CLK_REF is stopped in the actual sleep mode. FREQM interrupts can wake up the device from sleep modes when the measurement is done, but only from sleep modes where CLK_FREQM is running. Please refer to the Power Manager chapter for details.

13.4.2 Clocks

The clock for the FREQM bus interface (CLK_FREQM) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the FREQM before disabling the clock, to avoid freezing the FREQM ia an undefined state.

A set of clocks can be selected as reference (CLK_REF) and another set of clocks can be selected for measurement (CLK_MSR). Please refer to the CLKSEL and REFSEL tables in the Module Configuration section for details.

13.4.3 Debug Operation

When an external debugger forces the CPU into debug mode, the FREQM continues normal operation. If the FREQM is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

13.4.4 Interrupts

The FREQM interrupt request line is connected to the internal source of the interrupt controller. Using the FREQM interrupt requires the interrupt controller to be programmed first.

13.5 Functional Description

The FREQM accuratly measures the frequency of a clock by comparing the frequency to a known frequency:

$$f_{CLK\_MSR} = (VALUE/REFNUM)*f_{CLK\_REF}$$

13.5.1 Reference Clock

The Reference Clock Selection (REFSEL) field in the Mode Register (MODE) selects the clock source for CLK_REF. The reference clock is enabled by writing a one to the Reference Clock Enable (REFCEN) bit in the Mode Register. This clock should have a known frequency.

CLK_REF needs to be disabled before switching to another clock. The RCLKBUSY bit in the Status Register (SR) indicates whether the clock is busy or not. This bit is set when the MODE.REFCEN bit is written.

To change CLK_REF:

- Write a zero to the MODE.REFCEN bit to disable the clock, without changing the other bits/fields in the Mode Register.
- Wait until the SR.RCLKBUSY bit reads as zero.
- Change the MODE.REFSEL field.
- Write a one to the MODE.REFCEN bit to enable the clock, without changing the other bits/fields in the Mode Register.
- Wait until the SR.RCLKBUSY bit reads as zero.

To enable CLK_REF:

- Write the correct value to the MODE.REFSEL field.
- Write a one to the MODE.REFCEN to enable the clock, without changing the other bits/fields in the Mode Register.
- Wait until the SR.RCLKBUSY bit reads as zero.

To disable CLK_REF:

- Write a zero to the MODE.REFCEN to disable he clock, without changing the other bits/fields in the Mode register.
- Wait until the SR.RCLKBUSY bit reads as zero.

13.5.1.1 Cautionary note

Note that if clock selected as source for CLK_REF is stopped during a measurement, this will not be detected by the FREQM. The BUSY bit in the STATUS register will never be cleared, and the DONE interrupt will never be triggered. If the clock selected as soruce for CLK_REF is stopped, it will not be possible to change the source for the reference clock as long as the selected source is not running.

13.5.2 Measurement

In the Mode Register the Clock Source Selection (CLKSEL) field selects CLK_MSR and the Number of Reference Clock Cycles (REFNUM) field selects the duration of the measurement. The duration is given in number of CLK_REF periodes.

Writing a one to the START bit in the Control Register (CTRL) starts the measurement. The BUSY bit in SR is cleared when the measurement is done.

The result of the measurement can be read from the Value Register (VALUE). The frequency of the measured clock CLK_MSR is then:

$$f_{CLK\_MSR} = (VALUE/REFNUM)*f_{CLK\_REF}$$

13.5.3 Interrupts

The FREQM has two interrupt sources:

- DONE: A frequency measurement is done
- RCLKRDY: The reference clock is ready

These will generate an interrupt request if the corresponding bit in the Interrupt Mask Register (IMR) is set. The interrupt sources are ORed together to form one interrupt request. The FREQM will generate an interrupt request if at least one of the bits in the Interrupt Mask Register (IMR) is set. Bits in IMR are set by writing a one to the corresponding bit in the Interrupt Enable Register (IER) and cleared by writing a one to this bit in the Interrupt Disable Register (IDR). The interrupt request remains active until the corresponding bit in the Interrupt Status Register (ISR) is cleared by writing a one to this bit in the Interrupt Clear Register (ICR). Because all the interrupt sources are ORed together, the interrupt request from the FREQM will remain active until all the bits in ISR are cleared.

13.6 User Interface

Table 13-1. FREQM Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Control Register	CTRL	Write-only	0x00000000
0x004	Mode Register	MODE	Read/Write	0x00000000
0x008	Status Register	STATUS	Read-only	0x00000000
0x00C	Value Register	VALUE	Read-only	0x00000000
0x010	Interrupt Enable Register	IER	Write-only	0x00000000
0x014	Interrupt Disable Register	IDR	Write-only	0x00000000
0x018	Interrupt Mask Register	IMR	Read-only	0x00000000
0x01C	Interrupt Status Register	ISR	Read-only	0x00000000
0x020	Interrupt Clear Register	ICR	Write-only	0x00000000
0x3FC	Version Register	VERSION	Read-only	-(1)

Note: 1. The reset value for this register is device specific. Please refer to the Module Configuration section at the end of this chapter.

13.6.1 Control Register

Name: CTRL

Access Type: Write-only

Offset: 0x000

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	START

- **START**

  Writing a zero to this bit has no effect.

  Writing a one to this bit will start a measurement.

13.6.2 Mode Register

Name: MODE

Access Type: Read/Write

Offset: 0x004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
REFCEN	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	CLKSEL				

15	14	13	12	11	10	9	8
REFNUM							

7	6	5	4	3	2	1	0
-	-	-	-	-	REFSEL		

- **REFCEN: Reference Clock Enable**
  0: The reference clock is disabled
  1: The reference clock is enabled
- **CLKSEL: Clock Source Selection**
  Selects the source for CLK_MSR. See table in Module Configuration chapter for details.
- **REFNUM: Number of Reference Clock Cycles**
  Selects the duration of a measurement, given in number of CLK_REF cycles.
- **REFSEL: Reference Clock Selection**
  Selects the source for CLK_REF. See table in Module Configuration chapter for details.

13.6.3 Status Register

Name: STATUS

Access Type: Read-only

Offset: 0x008

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	RCLKBUSY	BUSY

- **RCLKBUSY: FREQM Reference Clock Status**
  0: The FREQM ref clk is ready, so a measurement can start.
  1: The FREQM ref clk is not ready, so a measurement should not be started.
- **BUSY: FREQM Status**
  0: The Frequency Meter is idle.
  1: Frequency measurement is on-going.

13.6.4 Value Register

Name: VALUE

Access Type: Read-only

Offset: 0x00C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
VALUE[23:16]							

15	14	13	12	11	10	9	8
VALUE[15:8]							

7	6	5	4	3	2	1	0
VALUE[7:0]							

- **VALUE:**
  Result from measurement.

13.6.5 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x010

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	RCLKRDY	DONE

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

13.6.6 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x014

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	RCLKRDY	DONE

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

13.6.7 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x018

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	RCLKRDY	DONE

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

13.6.8 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x01C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	RCLKRDY	DONE

0: The corresponding interrupt is cleared.
1: The corresponding interrupt is pending.
A bit in this register is set when the corresponding bit in STATUS has a one to zero transition.
A bit in this register is cleared when the corresponding bit in ICR is written to one.

13.6.9 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x020

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	RCLKRDY	DONE

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in ISR and the corresponding interrupt request.

13.6.10 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x3FC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

13.7 Module Configuration

The specific configuration for each FREQM instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 13-2. Module Clock Name

Module Name	Clock Name	Description
FREQM	CLK_FREQM	Peripheral Bus clock from the PBA clock domain
	CLK_MSR	Measured clock
	CLK_REF	Reference clock

Table 13-3. Register Reset Values

Register	Reset Value
VERSION	0x00000310

Table 13-4. Clock Sources for CLK_MSR

CLKSEL	Clock/Oscillator	Description
0	CLK_CPU	The clock the CPU runs on
1	CLK_HSB	High Speed Bus clock
2	CLK_PBA	Peripheral Bus A clock
3	CLK_PBB	Peripheral Bus B clock
4	CLK_PBC	Peripheral Bus C clock
5	OSC0	Output clock from Oscillator 0
6	OSC1	Output clock from Oscillator 1
7	OSC32K	Output clock from OSC32K
8	RCSYS	Output clock from RCSYS Oscillator
9	RC8M	Output clock from 8MHz / 1MHz RC Oscillator
13-24	GCLK0-11	Generic clock 0 through 11
25	RC120M	Output clock from RC120M
26-31	Reserved	

Table 13-5. Clock Sources for CLK_REF

REFSEL	Clock/Oscillator	Description
0	RCSYS	System 115 KHz RC oscillator clock
1	OSC32K	Output clock from OSC32K
2	OSC0	Output clock from Oscillator 0

Table 13-5. Clock Sources for CLK_REF

REFSEL	Clock/Oscillator	Description
3	OSC1	Output clock from Oscillator 1
4	RC8M	Output clock from 8MHz / 1MHz RC Oscillator
5	GCLK9	Generic Clock 9
6-7	Reserved	

14. Peripheral Event Controller (PEVC)

Rev: 1.0.0.0

14.1 Features

- **Direct peripheral to peripheral communication system**
- **Allows peripherals to receive, react to, and send peripheral events without CPU intervention**
- **Cycle deterministic event communication**
- **Asynchronous interrupts allow advanced peripheral operation in low power sleep modes**

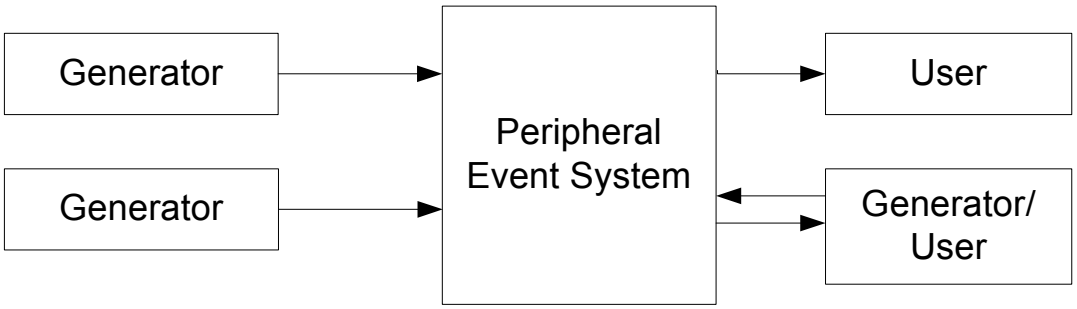
14.2 Overview

Many peripheral modules can be configured to emit or respond to signals known as peripheral events. The exact condition to trigger a peripheral event, or the action taken upon receiving a peripheral event, is specific to each module. Peripherals that respond to events are called users and peripherals that emit events are called generators. A module may be both a generator and user.

The peripheral event generators and users are interconnected by a network known as the Peripheral Event System. The Peripheral Event Controller (PEVC) controls the interconnection parameters, such as generator-to-user multiplexing and peripheral event enable/disable.

The Peripheral Event System allows low latency peripheral-to-peripheral signalling without CPU intervention, and without consuming system resources such as bus or RAM bandwidth. This offloads the CPU and system resources compared to a traditional interrupt-based software driven system.

Figure 14-1. Peripheral Event System Overview



14.3 Block Diagram

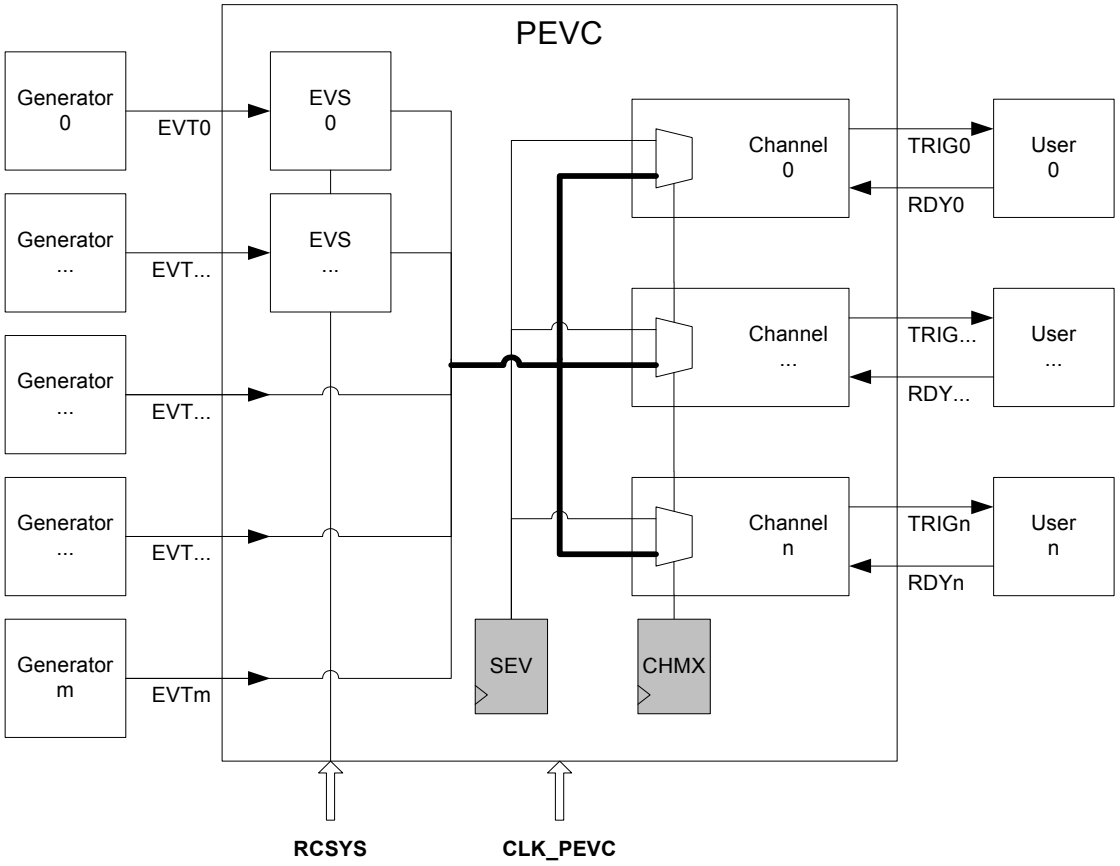
The main building blocks of the PEVC are:

- Event Shapers (EVS): Instantiated for some generators, in case filtering and/or rising/falling edge detection is needed prior to peripheral event propagation
- Channels: One channel per user, to propagate events and follow-up the user status

To help distinguish the different signalling stages, these naming conventions are used:

- Generators generate events
- PEVC multiplexes these incoming events
- PEVC outputs Triggers to users

Figure 14-2. PEVC Block Diagram



The maximum number of generators, Event Shapers, channels, and users supported by the Peripheral Event Controller is 64. Please refer to the Module Configuration section at the end of this chapter for the device-specific configuration.

14.4 I/O Lines Description

Table 14-1. I/O Lines Description

Pin Name	Pin Description	Type
PAD_EVT[n]	External Event Inputs	Input

14.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

14.5.1 I/O Lines

Multiplexed I/O lines can be used as event generators. To generate a peripheral event from an external source the source pin must be configured as an input pin by the I/O Controller. It is also possible to trigger a peripheral event by driving these pins from registers in the I/O Controller, or another peripheral output connected to the same pin.

14.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the PEVC, the PEVC will stop functioning and resume operation after the system wakes up from sleep mode. Peripheral events do not require CPU intervention, and are therefore available in sleep modes where the CPU sleeps.

14.5.3 Clocks

The PEVC has two bus clocks connected: One Peripheral Bus clock (CLK_PEVC) and the system RC oscillator clock (CLK_RCSYS). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by writing to the Power Manager.

CLK_RCSYS is used for glitch filtering.

14.5.4 Interrupts

PEVC can generate an interrupt request in case of trigger generation or trigger overrun. The PEVC interrupt request lines are connected to the interrupt controller. Using the PEVC interrupts requires the interrupt controller to be programmed first.

14.5.5 Debug Operation

PEVC is frozen during debug operation, unless the Run In Debug bit in the Development Control Register is set and the bit corresponding to the PEVC is set in the Peripheral Debug Register (PDBG). Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual, and the OCD Module Configuration section, for details.

14.6 Functional Description

14.6.1 PEVC Channel Operation

PEVC routes incoming events to users by means of one channel per user. Channels operate in parallel, allowing multiple users to listen to the same generator.

14.6.1.1 Channel Setup

The Channel Multiplexer Register (CHMXn) is written to allocate a generator to a given channel. The Event Multiplexer field (EVMX) selects between the different generators, while the Software Event Multiplexer bit (SMX) selects Software Events.

The channel is then enabled by writing a one to the appropriate bit in the Channel Enable Register (CHER). It is disabled by writing a one to the appropriate bit in the Channel Disable Register (CHDR).

To safely program a channel, user software must:

- disable the channel by writing a one to CHDR
- program CHMXn
- enable the channel by writing a one to CHER

14.6.1.2 Channel Operation

When the channel is enabled, the user signals its busy/ready state to the channel, to determine how an incoming event will be handled:

- If the user is ready, an incoming event is forwarded. The corresponding Trigger Status Register (TRISR) flag is set allowing an interrupt to be generated for tracking PEVC operations.
- If the user is busy (because of a previous event, or for some other cause), the new event is not forwarded. The corresponding Overrun Status Register (OVSR) flag is set allowing an interrupt to be generated.

The Busy Register (BUSY) is used to determine the current activity of a channel/user. A busy status has one of two causes:

- A peripheral event is being relayed by the channel and handled by the user,
- No event relayed, but user is not ready (e.g. not initialized, or handling some other request).

14.6.1.3 Software Event

A Software Event can be initiated by software writing to the Software Event Register (SEV). This is intended for application debugging.

The channel must first be programmed by writing a one to the Software Event Multiplexer bit (SMX) of CHMXn.

Writing a one to the appropriate bit of SEV will then trigger a Software Event on the channel.

14.6.2 Event Shaper (EVS) Operation

PEVC contains Event Shapers (EVS) for certain types of generators:

- Asynchronous generators and/or external input
- General-purpose waveforms like timer outputs or Generic Clocks

Each Event Shaper is responsible of shaping one input, prior to going through a PEVC channel:

- Synchronize asynchronous external inputs
- Apply any additional glich-filtering
- Detect rise, fall, or both edges of the incoming signal

14.6.2.1 Input Glitch Filter (IGF)

Input Glitch Filtering can be turned on or off by writing to the Input Glitch Filter (IGF) field of the corresponding Event Shaper Register (EVS).

When IGF is on, the incoming event is sampled periodically. The sampling clock is divided from CLK_RCSYS by the value of the Input Glitch Filter Divider Register (IGFDR). IGF will filter out spikes and propagate only incoming events that respect one of the following two conditions :

- rise event : 2 samples low, followed by 0+ changes, followed by 2 samples high
- fall event : 2 samples high, followed by 0+ changes, followed by 2 samples low

Both CLK_RCSYS and CLK_PEVC must be enabled to use Input Glitch Filtering.

14.6.3 Event Propagation Latency

Once a channel is setup, incoming peripheral events are relayed by hardware. Event progation latency is therefore cycle deterministic. However, its value depends on the exact settings that apply to a given channel.

When the channel multiplexer CHMXn.EVMX selects a generator without Event Shaper, event propagation latency is 0 cycle. Software event is a particular case of 0 cycle propagation.

When the channel multiplexer CHMXn.EVMX selects a generator with Event Shaper, event propagation latency depends on Input Glitch Filter setting EVSm.IGF :

- IGF off : event propagation latency is lesser or equal to 2 CLK_PEVC cycles
- IGF on : event propagation latency is lesser or equal to $3 * 2^{IGFDR+1} *$ CLK_RCSYS cycles

Table 14-2. Event Propagation Latency

Generator CHMXn.EVMX	Input Glitch Filter EVSm.IGF	Latency	Clock
Generator without Event Shaper	-	0	-
Software event	-	0	-
Generator with Event Shaper	Off	2	CLK_PEVC
Generator with Event Shaper	On	$3 * 2^{IGFDR+1}$	CLK_RCSYS

Please refer to the Module Configuration section at the end of this chapter for the list of generators implementing Event Shapers.

14.7 User Interface

PEVC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Version	VERSION	Read-only	-[1]
0x004	Parameter	PARAMETER	Read-only	-[1]
0x008	Input Glitch Filter Divider Register	IGFDR	Read/Write	0x00000000
0x010 - 0x014	Channel Status Register	CHSRi[2]	Read-only	0x00000000
0x020 - 0x024	Channel Enable Register	CHERi[2]	Write-only	-
0x030 - 0x034	Channel Disable Register	CHDRi[2]	Write-only	-
0x040 - 0x044	Software Event	SEVi[2]	Write-only	-
0x050 - 0x054	Channel / User Busy	BUSYi[2]	Read-only	-[1]
0x060 - 0x064	Trigger Status Register	TRSRi[2]	Read-only	0x00000000
0x070 - 0x074	Trigger Status Clear Register	TRSCRi[2]	Write-only	-
0x080 - 0x084	Trigger Interrupt Mask Register	TRIMRi[2]	Read-only	0x00000000
0x090 - 0x094	Trigger Interrupt Mask Enable Register	TRIERi[2]	Write-only	-
0x0A0 - 0x0A4	Trigger Interrupt Mask Disable Register	TRIDRi[2]	Write-only	-
0x0B0 - 0x0B4	Overrun Status Register	OVSRi[2]	Read-only	0x00000000
0x0C0 - 0x0C4	Overrun Status Clear Register	OVSCRi[2]	Write-only	-
0x0D0 - 0x0D4	Overrun Interrupt Mask Register	OVIMRi[2]	Read-only	0x00000000
0x0E0 - 0x0E4	Overrun Interrupt Mask Enable Register	OVIERi[2]	Write-only	-
0x0F0 - 0x0F4	Overrun Interrupt Mask Disable Register	OVIDRi[2]	Write-only	-
0x100	Channel Multiplexer 0	CHMX0	Read/Write	0x00000000
0x100 + n*0x004	Channel Multiplexer n	CHMXn	Read/Write	0x00000000
0x1FC	Channel Multiplexer 63	CHMX63	Read/Write	0x00000000
0x200	Event Shaper 0	EVS0	Read/Write	0x00000000
0x200 + m*0x004	Event Shaper m	EVSm	Read/Write	0x00000000
0x2FC	Event Shaper 63	EVS63	Read/Write	0x00000000

Notes:
1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.
2. i={0,1}, where i=0 contains the lowest 32 channels, and i=1 contains the highest 32 channels. The lowest address contains register 0, the highest address contains register 1. Register 1 is only implemented if the device has more than 32 channels implemented. Please refer to the Module Configuration section at the end of this chapter for details.

14.7.1 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x000

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Variant number of the module. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

14.7.2 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x004

Reset Value: -

31	30	29	28	27	26	25	24
TRIGOUT							

23	22	21	20	19	18	17	16
EVMX							

15	14	13	12	11	10	9	8
EVS							

7	6	5	4	3	2	1	0
EVIN							

- **TRIGOUT: Number of Trigger Outputs / Channels / Users**
  Number of trigger outputs / channels implemented. No functionality associated.
- **EVMX: Number of Bits to control EVMX field in CHMXn Registers**
  Number of Multiplexers control bits, derived from EVIN. No functionality associated.
- **EVS: Number of Event Shapers**
  Number of Event Shapers implemented. No functionality associated.
- **EVIN: Number of Event Inputs / Generators**
  Number of event inputs. No functionality associated.

14.7.3 Input Glitch Filter Divider Register

Name: IGFDR

Access Type: Read/Write

Offset: 0x008

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	IGFDR			

- **IGFDR: Input Glitch Filter Divider**
  Selects prescaler division ratio for the system RC clock used for glitch filtering.

IGFDR	Division Ratio
0x0	2
0x1	4
0x2	8
0xn	2^{n+1}
0xF	65536

14.7.4 Channel Status Register

Name: CHSR0 - CHSR1

Access Type: Read-only

Offset: 0x010 - 0x014

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CHS							

23	22	21	20	19	18	17	16
CHS							

15	14	13	12	11	10	9	8
CHS							

7	6	5	4	3	2	1	0
CHS							

- **CHS: Channel Status**
  0: The corresponding channel is disabled.
  1: The corresponding channel is enabled.
  This bit is cleared when the corresponding bit in CHDR is written to one.
  This bit is set when the corresponding bit in CHER is written to one.

  Note:
  Channels 0 to 31 are controlled by CHSR0.
  Channels 32 to 63 are controlled by CHSR1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.5 Channel Enable Register

Name: CHER0 - CHER1

Access Type: Write-only

Offset: 0x020 - 0x024

Reset Value: -

31	30	29	28	27	26	25	24
CHE							

23	22	21	20	19	18	17	16
CHE							

15	14	13	12	11	10	9	8
CHE							

7	6	5	4	3	2	1	0
CHE							

- **CHE: Channel Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the corresponding bit in CHSR.

  Note:
  Channels 0 to 31 are controlled by CHER0.
  Channels 32 to 63 are controlled by CHER1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.6 Channel Disable Register

Name: CHDR0 - CHDR1

Access Type: Write-only

Offset: 0x030 - 0x034

Reset Value: -

31	30	29	28	27	26	25	24
CHD							

23	22	21	20	19	18	17	16
CHD							

15	14	13	12	11	10	9	8
CHD							

7	6	5	4	3	2	1	0
CHD							

- **CHD: Channel Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in CHSR.

  Note:
  Channels 0 to 31 are controlled by CHER0.
  Channels 32 to 63 are controlled by CHER1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.7 Software Event Register

Name: SEV0 - SEV1

Access Type: Write-only

Offset: 0x040 - 0x044

Reset Value: -

31	30	29	28	27	26	25	24
SEV							

23	22	21	20	19	18	17	16
SEV							

15	14	13	12	11	10	9	8
SEV							

7	6	5	4	3	2	1	0
SEV							

- **SEV: Software Event**

  Writing a zero to this bit has no effect.
  Writing a one to this bit will trigger a Software Event for the corresponding channel.

  Note:
  Channels 0 to 31 are controlled by SEV0.
  Channels 32 to 63 are controlled by SEV1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.8 Channel / User Busy

Name: BUSY0 - BUSY1

Access Type: Read-only

Offset: 0x050 - 0x054

Reset Value: -

31	30	29	28	27	26	25	24
BUSY							

23	22	21	20	19	18	17	16
BUSY							

15	14	13	12	11	10	9	8
BUSY							

7	6	5	4	3	2	1	0
BUSY							

- **BUSY: Channel Status**
  0: The corresponding channel and user are idle.
  1: The corresponding channel and user are busy.

  Note:
  Channels 0 to 31 are controlled by BUSY0.
  Channels 32 to 63 are controlled by BUSY1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.9 Trigger Status Register

Name: TRSR0 - TRSR1

Access Type: Read-only

Offset: 0x060 - 0x064

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
TRS							

23	22	21	20	19	18	17	16
TRS							

15	14	13	12	11	10	9	8
TRS							

7	6	5	4	3	2	1	0
TRS							

- **TRS: Trigger Interrupt Status**
  0: An interrupt event has not occurred
  1: An interrupt event has occurred
  This bit is cleared by writing a one to the corresponding bit in TRSCR.

  Note:
  Channels 0 to 31 are controlled by TRSR0.
  Channels 32 to 63 are controlled by TRSR1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.10 Trigger Status Clear Register

Name: TRSCR0 - TRSCR1

Access Type: Write-only

Offset: 0x070 - 0x074

Reset Value: -

31	30	29	28	27	26	25	24
TRSC							

23	22	21	20	19	18	17	16
TRSC							

15	14	13	12	11	10	9	8
TRSC							

7	6	5	4	3	2	1	0
TRSC							

- **TRSC: Trigger Interrupt Status Clear**

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the corresponding bit in TRSR.

  Note:

  Channels 0 to 31 are controlled by TRSCR0.

  Channels 32 to 63 are controlled by TRSCR1.

  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.11 Trigger Interrupt Mask Register

Name: TRIMR0 - TRIMR1

Access Type: Read-only

Offset: 0x080 - 0x084

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
TRIM							

23	22	21	20	19	18	17	16
TRIM							

15	14	13	12	11	10	9	8
TRIM							

7	6	5	4	3	2	1	0
TRIM							

- **TRIM: Trigger Interrupt Mask**

  0: The corresponding interrupt is disabled.

  1: The corresponding interrupt is enabled.

  This bit is cleared when the corresponding bit in TRIDR is written to one.

  This bit is set when the corresponding bit in TRIER is written to one.

  Note:

  Channels 0 to 31 are controlled by TRIMR0.

  Channels 32 to 63 are controlled by TRIMR1.

  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.12 Trigger Interrupt Enable Register

Name: TRIER0 - TRIER1

Access Type: Write-only

Offset: 0x090 - 0x094

Reset Value: -

31	30	29	28	27	26	25	24
TRIE							

23	22	21	20	19	18	17	16
TRIE							

15	14	13	12	11	10	9	8
TRIE							

7	6	5	4	3	2	1	0
TRIE							

- **TRIE: Trigger Interrupt Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the corresponding bit in TRIMR.

  Note:
  Channels 0 to 31 are controlled by TRIER0.
  Channels 32 to 63 are controlled by TRIER1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.13 Trigger Interrupt Disable Register

Name: TRIDR0 - TRIDR1

Access Type: Write-only

Offset: 0x0A0 - 0x0A4

Reset Value: -

31	30	29	28	27	26	25	24
TRID							

23	22	21	20	19	18	17	16
TRID							

15	14	13	12	11	10	9	8
TRID							

7	6	5	4	3	2	1	0
TRID							

- **TRID: Trigger Interrupt Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in IMR.

  Note:
  Channels 0 to 31 are controlled by TRIDR0.
  Channels 32 to 63 are controlled by TRIDR1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.14 Overrun Status Register

Name: OVSR0 - OVSR1

Access Type: Read-only

Offset: 0x0B0 - 0x0B4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
OVS							

23	22	21	20	19	18	17	16
OVS							

15	14	13	12	11	10	9	8
OVS							

7	6	5	4	3	2	1	0
OVS							

- **OVS: Overrun Interrupt Status**
  0: An interrupt event has not occurred
  1: An interrupt event has occurred
  This bit is cleared by writing a one to the corresponding bit in OVSCR.

  Note:
  Channels 0 to 31 are controlled by OVSR0.
  Channels 32 to 63 are controlled by OVSR1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.15 Overrun Status Clear Register

Name: OVSCR0 - OVSCR1

Access Type: Write-only

Offset: 0x0C0 - 0x0C4

Reset Value: -

31	30	29	28	27	26	25	24
OVSC							

23	22	21	20	19	18	17	16
OVSC							

15	14	13	12	11	10	9	8
OVSC							

7	6	5	4	3	2	1	0
OVSC							

- **OVSC: Overrun Interrupt Status Clear**

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the corresponding bit in OVSR.

  Note:

  Channels 0 to 31 are controlled by OVSCR0.

  Channels 32 to 63 are controlled by OVSCR1.

  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.16 Overrun Interrupt Mask Register

Name: OVIMR0 - OVIMR1

Access Type: Read-only

Offset: 0x0D0 - 0x0D4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
OVIM							

23	22	21	20	19	18	17	16
OVIM							

15	14	13	12	11	10	9	8
OVIM							

7	6	5	4	3	2	1	0
OVIM							

- **OVIM: Overrun Interrupt Mask**

  0: The corresponding interrupt is disabled.

  1: The corresponding interrupt is enabled.

  This bit is cleared when the corresponding bit in OVIDR is written to one.

  This bit is set when the corresponding bit in OVIER is written to one.

  Note:

  Channels 0 to 31 are controlled by OVIMR0.

  Channels 32 to 63 are controlled by OVIMR1.

  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.17 Overrun Interrupt Enable Register

Name: OVIER0 - OVIER1

Access Type: Write-only

Offset: 0x0E0 - 0x0E4

Reset Value: -

31	30	29	28	27	26	25	24
OVIE							

23	22	21	20	19	18	17	16
OVIE							

15	14	13	12	11	10	9	8
OVIE							

7	6	5	4	3	2	1	0
OVIE							

- **OVIE: Overrun Interrupt Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the corresponding bit in OVIMR.

  Note:
  Channels 0 to 31 are controlled by OVIER0.
  Channels 32 to 63 are controlled by OVIER1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.18 Overrun Interrupt Disable Register

Name: OVIDR0 - OVIDR1

Access Type: Write-only

Offset: 0x0F0 - 0x0F4

Reset Value: -

31	30	29	28	27	26	25	24
OVID							

23	22	21	20	19	18	17	16
OVID							

15	14	13	12	11	10	9	8
OVID							

7	6	5	4	3	2	1	0
OVID							

- **OVID: Overrun Interrupt Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in IMR.

  Note:
  Channels 0 to 31 are controlled by OVIDR0.
  Channels 32 to 63 are controlled by OVIDR1.
  Please refer to the Module Configuration section at the end of this Chapter for device-specific channel mapping information.

14.7.19 Channel Multiplexer Register

Name: CHMXn

Access Type: Read/Write

Offset: 0x100 + n*0x004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	SMX

7	6	5	4	3	2	1	0
-	-	EVMX					

- **EVMX: Event Multiplexer**
  Select input event / generator.
- **SMX: Software Event Multiplexer**
  0: The Software Event is not selected. Event / generator is selected by EVMX.
  1: The Software Event is selected. EVMX is not considered.

EVMX	SMX	Channel Input
0x00	0	EVT0
0x01	0	EVT1
0xm	0	EVTm
> TRIGOUT	0	None
Any	1	Software Event

14.7.20 Event Shaper Register

Name: EVSm

Access Type: Read/Write

Offset: 0x200 + m*0x004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	IGF	EVF	EVR

- **EVR: Event Rise**
  0: No event detection on rising edge.
  1: Event detection on rising edge.
- **EVF: Event Fall**
  0: No event detection on falling edge.
  1: Event detection on falling edge.
- **IGF: Input Glitch Filter**
  0: Input glitch filter is off.
  1: Input glitch filter is on.

14.8 Module Configuration

The specific configuration for each PEVC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Refer to the Power Manager chapter for details.

Table 14-3. Module Configuration

Feature	Parameter	PEVC
Number of Generators	EVIN	34
Number of Event Shapers	EVS	24
Number of Channels / Users	TRIGOUT	22

Table 14-4. Module Clock Name

Module name	Clock name	Description
PEVC	CLK_PEVC	HSB clock

Table 14-5. Register Reset Values

Register	Reset Value
BUSY0	0x0000FFFF
VERSION	0x00000100
PARAMETER	Refer to Table 14-3

The BUSY0[21:16] field will always be read as 0x3F.

The PEVC routes events from event generator to trigger an action in the event user. The following tables defines the corresponding input event generator in EVMx registers and if an event shaper is implemented for this generator.

Table 14-6. PEVC event numbers

Event Number (EVMx)	Event Generator - Event source	Event Shaper
[15:0]	PAD_EVT[15:0] - change on input pins	Yes
16	the generic clock GCLK7	Yes
17	the generic clock GCLK8	Yes
18	TC0 - A0 rising edge	Yes
19	TC0 - A1 rising edge	Yes
20	TC0 - A2 rising edge	Yes
21	TC0 - B0 rising edge	Yes
22	TC0 - B1 rising edge	Yes
23	TC0 - B2 rising edge	Yes
24	ACIFA0 - event 0	
25	ACIFA0 - event 1	

Table 14-6. PEVC event numbers

Event Number (EVMx)	Event Generator - Event source	Event Shaper
26	ACIFA1 - event 0	
27	ACIFA1 - event 1	
28	AST - alarm event 0	
29	AST - period event 0	
30	PWM - compare match on event 0	
31	PWM - compare match on event 1	
32	QDEC0 - compare match	
33	QDEC1 - compare match	

The following tables defines the triggered action for each PEVC channel.

Table 14-7. PEVC channel

channel Number	Event User - Triggered Event
0	ADCIFA - sequencer 0 start of conversion
1	ADCIFA - sequencer 1 start of conversion
2	DACIFB0 - start of conversion chA
3	DACIFB0 - start of conversion chB
4	DACIFB1 - start of conversion chA
5	DACIFB1 - start of conversion chB
6	PDCA - start of transfer channel 0
7	PDCA - start of transfer channel 1
8	PWM - fault input 0
9	PWM - fault input 1
10	QDEC0 - capture
11	QDEC0 - toggle direction of counting
12	QDEC0- trigger
13	QDEC1 - capture
14	QDEC1 - toggle direction of counting
15	QDEC1- trigger
16	TC0 - input capture A0
17	TC0 - input capture A1
18	TC0 - input capture A2
19	TC0 - input capture B0
20	TC0 - input capture B1
21	TC0 - input capture B2

15. Flash Controller (FLASHC)

Rev: 3.0.2.2

15.1 Features

- **Controls flash block with dual read ports allowing staggered reads.**
- **Supports 0 and 1 wait state bus access.**
- **Allows interleaved burst reads for systems with one wait state, outputting one 32-bit word per clock cycle.**
- **32-bit HSB interface for reads from flash array and writes to page buffer.**
- **32-bit PB interface for issuing commands to and configuration of the controller.**
- **16 lock bits, each protecting a region consisting of (total number of pages in the flash block / 16) pages.**
- **Regions can be individually protected or unprotected.**
- **Additional protection of the Boot Loader pages.**
- **Supports reads and writes of general-purpose NVM bits.**
- **Supports reads and writes of additional NVM pages.**
- **Supports device protection through a security bit.**
- **Dedicated command for chip-erase, first erasing all on-chip volatile memories before erasing flash and clearing security bit.**
-

15.2 Overview

The Flash Controller (FLASHC) interfaces the on-chip flash memory with the 32-bit internal HSB bus. The controller manages the reading, writing, erasing, locking, and unlocking sequences.

15.3 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

15.3.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the FLASHC, the FLASHC will stop functioning and resume operation after the system wakes up from sleep mode.

15.3.2 Clocks

The FLASHC has two bus clocks connected: One High Speed Bus clock (CLK_FLASHC_HSB) and one Peripheral Bus clock (CLK_FLASHC_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by writing to the Power Manager. The user has to ensure that CLK_FLASHC_HSB is not turned off before reading the flash or writing the pagebuffer and that CLK_FLASHC_PB is not turned off before accessing the FLASHC configuration and control registers. Failing to do so may deadlock the bus.

15.3.3 Interrupt

The FLASHC interrupt request lines are connected to the interrupt controller. Using the FLASHC interrupts requires the interrupt controller to be programmed first.

15.3.4 Debug Operation

When an external debugger forces the CPU into debug mode, the FLASHC continues normal operation. If the FLASHC is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

15.4 Functional description

15.4.1 Bus Interfaces

The FLASHC has two bus interfaces, one High-Speed Bus (HSB) interface for reads from the flash memory and writes to the page buffer, and one Peripheral Bus (PB) interface for issuing commands and reading status from the controller.

15.4.2 Memory Organization

The flash memory is divided into a set of pages. A page is the basic unit addressed when programming the flash. A page consists of several words. The pages are grouped into 16 regions of equal size. Each of these regions can be locked by a dedicated fuse bit, protecting it from accidental modification.

- *p* pages (*FLASH_P*)
- w bytes in each page and in the page buffer (*FLASH_W*)
- *pw* bytes in total (*FLASH_PW*)
- *f* general-purpose fuse bits (*FLASH_F*), used as region lock bits and for other device-specific purposes
- 1 security fuse bit
- 1 Factory Page
- 1 User Page

15.4.3 User Page

The User page is an additional page, outside the regular flash array, that can be used to store various data, such as calibration data and serial numbers. This page is not erased by regular chip erase. The User page can only be written and erased by a special set of commands. Read accesses to the User page are performed just as any other read accesses to the flash. The address map of the User page is given in Figure 15-1.

15.4.4 Factory page

The Factory page is an additional page, outside the regular flash array, that can be used to store various data, such as calibration data and serial numbers. This page is not erased by regular chip erase. Read accesses to the Factory page is performed just as any other read access to the flash. The address map of the Factory page is given in Figure 15-1.

15.4.5 Read Operations

The on-chip flash memory is typically used for storing instructions to be executed by the CPU. The CPU will address instructions using the HSB bus, and the FLASHC will access the flash memory and return the addressed 32-bit word.

In systems where the HSB clock period is slower than the access time of the flash memory, the FLASHC can operate in 0 wait state mode, and output one 32-bit word on the bus per clock cycle. If the clock frequency allows, the user should use 0 wait state mode, because this gives the highest performance as no stall cycles are encountered.

The FLASHC can also operate in systems where the HSB bus clock period is faster than the access speed of the flash memory. Wait state support and a read granularity of 64 bits ensure efficiency in such systems.

Performance for systems with high clock frequency is increased since the flash internally is configured as two separate banks of 32 bits. Each bank has its own read port. In 0ws mode, only one of the two flash read ports is accessed. The other flash read port is idle. In 1ws mode, both flash read ports are active. One read port reading the addressed word, and the other reading the next sequential word.

The programmer can select the wait states required by writing to the FWS field in the Flash Control Register (FCR). It is the responsibility of the programmer to select a number of wait states compatible with the clock frequency and timing characteristics of the flash memory.

In 0ws mode, no wait states are encountered on any flash read operations. In 1 ws mode, one stall cycle is encountered on the first access in a single or burst transfer.

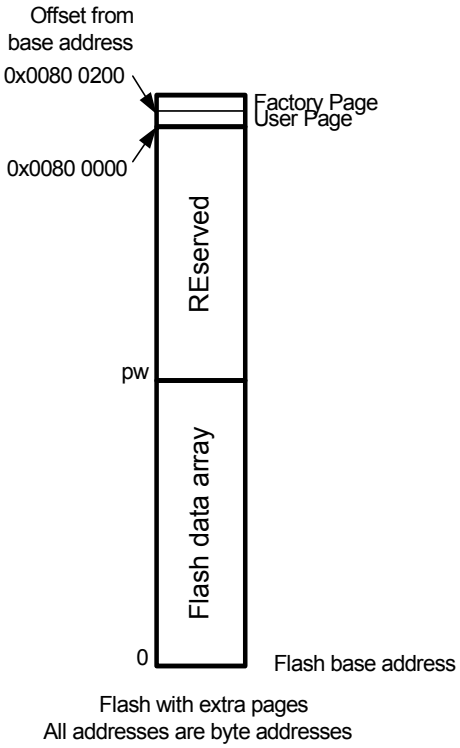
If the clock frequency allows, the user should use 0ws mode, because this gives the lowest power consumption for low-frequency systems as only one flash read port is read. Using 1ws mode has a power/performance ratio approaching 0ws mode as the clock frequency approaches twice the max frequency of 0ws mode. Using two flash read ports use twice the power, but also give twice the performance.

The Flash Controller address space is displayed in Figure 15-1. The memory space between address *pw* and the User page is reserved, and reading addresses in this space returns an undefined result. The User page is permanently mapped to an offset of 0x0080 0000 from the start address of the flash memory. The Factory page is permanently mapped to an offset of 0x0080 0200 from the start address of the flash memory.

Table 15-1. User page Addresses

Memory type	Start address, byte sized	Size
Main array	0	*pw* bytes
User	0x0080 0000	w bytes
Factory Page	0x0080 0200	w bytes

Figure 15-1. Memory Map for the Flash Memories2

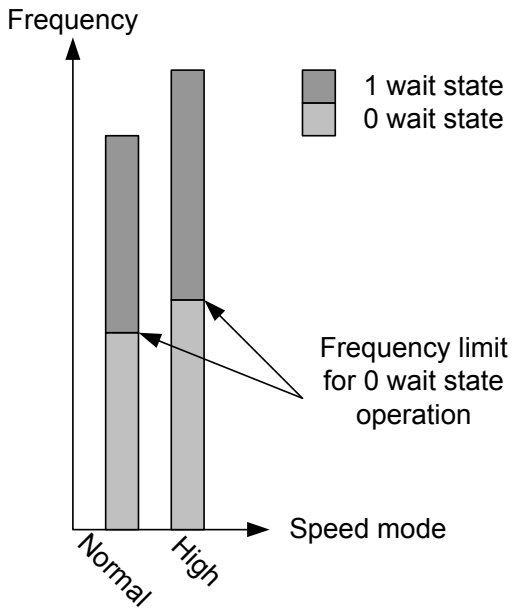


15.4.6 High Speed Read Mode

The flash provides a High Speed Read Mode, offering slightly higher flash read speed at the cost of higher power consumption. Two dedicated commands, High Speed Read Mode Enable (HSEN) and High Speed Read Mode Disable (HSDIS) control the speed mode. The High Speed Mode (HSMODE) in Flash Status Register (FSR) shows which mode the flash is in. After reset, the High Speed Mode is disabled, and must be manually enabled if the user wants to.

Refer to the Electrical Characteristics chapter at the end of this datasheet for details on the maximum clock frequencies in Normal and High Speed Read Mode.

Figure 15-2. High Speed Mode



15.4.7 Quick Page Read

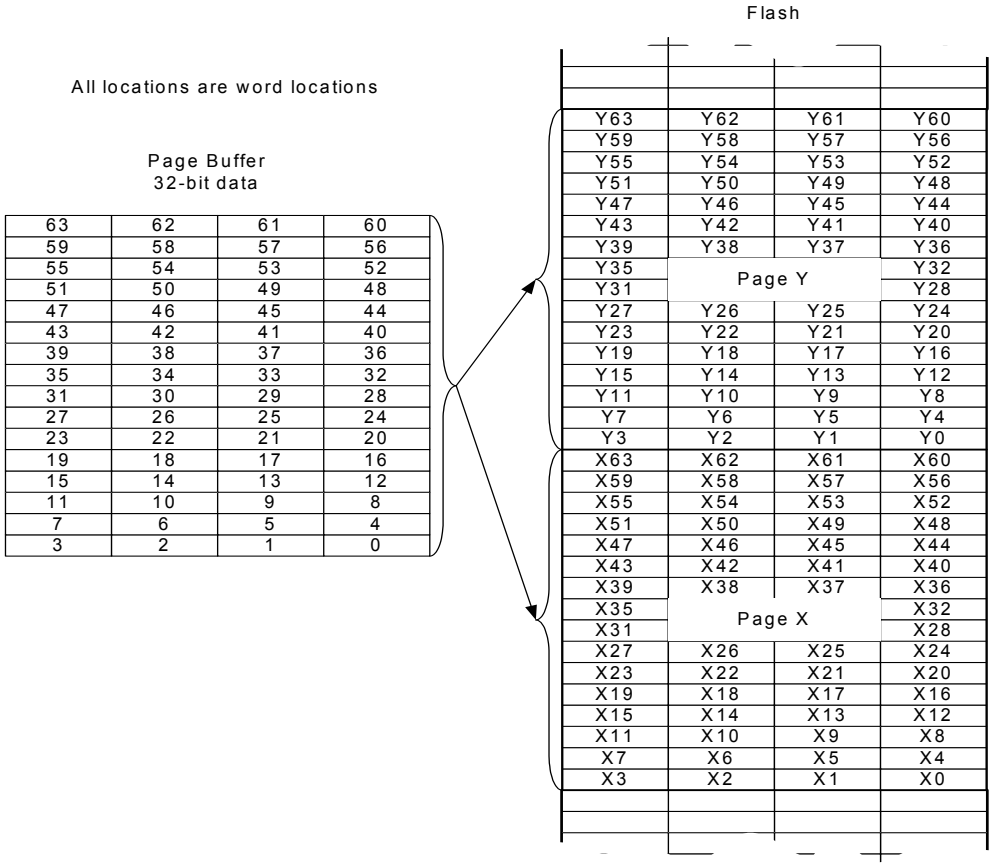
A dedicated command, Quick Page Read (QPR), is provided to read all words in an addressed page. All bits in all words in this page are AND'ed together, returning a 1-bit result. This result is placed in the Quick Page Read Result (QPRR) bit in Flash Status Register (FSR). The QPR command is useful to check that a page is in an erased state. The QPR instruction is much faster than performing the erased-page check using a regular software subroutine.

15.4.8 Page Buffer Operations

The flash memory has a write and erase granularity of one page; data are written and erased in chunks of one page. When programming a page, the user must first write the new data into the Page Buffer. The contents of the entire Page Buffer is copied into the desired page in flash memory when the user issues the Write Page command, See Section "15.5.1" on page 259.

In order to program data into flash page Y, write the desired data to locations Y0 to 63 in the regular flash memory map. Writing to an address A in the flash memory map will not update the flash memory, but will instead update location A%64 in the page buffer. The PAGEN field in the Flash Command (FCMD) register will at the same time be updated with the value A/64.

Figure 15-3. Mapping from Page Buffer to Flash



The page buffer is word-addressable and should only be written with aligned word transfers, never with byte or halfword transfers. The page buffer can not be read.

The page buffer is also used for writes to the User page.

Page buffer write operations are performed with 4 wait states. Any accesses attempted to the FLASHC on the HSB bus during these cycles will be automatically stalled.

Writing to the page buffer can only change page buffer bits from one to zero, i.e. writing 0xAAAAAAAA to a page buffer location that has the value 0x00000000 will not change the page buffer value. The only way to change a bit from zero to one is to erase the entire page buffer with the Clear Page Buffer command.

The page buffer is not automatically reset after a page write. The programmer should do this manually by issuing the Clear Page Buffer flash command. This can be done after a page write, or before the page buffer is loaded with data to be stored to the flash page.

15.4.9 Writing Words to a Page that is not Completely Erased

This can be used for EEPROM emulation, i.e. writes with granularity of one word instead of an entire page. Only words that are in an completely erased state (0xFFFFFFFF) can be changed. The procedure is as follows:

1. Clear page buffer
2. Write to the page buffer the result of the logical bitwise AND operation between the contents of the flash page and the new data to write. Only bits that were in an erased state can be changed from the original page.
3. Write Page.

15.5 Flash Commands

The FLASHC offers a command set to manage programming of the flash memory, locking and unlocking of regions, and full flash erasing. See Section 15.8.2 for a complete list of commands.

To run a command, the field CMD of the Flash Command Register (FCMD) has to be written with the command number. As soon as the FCMD register is written, the FRDY bit is automatically cleared. Once the current command is complete, the FRDY bit is automatically set. If an interrupt has been enabled by writing a one to FCR.FRDY, the interrupt request line of the Flash Controller is activated. All flash commands except for Quick Page Read (QPR) will generate an interrupt request upon completion if FRDY is written to one.

Any HSB bus transfers attempting to read flash memory when the FLASHC is busy executing a flash command will be stalled, and allowed to continue when the flash command is complete.

After a command has been written to FCMD, the programming algorithm should wait until the command has been executed before attempting to read instructions or data from the flash or writing to the page buffer, as the flash will be busy. The waiting can be performed either by polling the Flash Status Register (FSR) or by waiting for the flash ready interrupt. The command written to FCMD is initiated on the first clock cycle where the HSB bus interface in FLASHC is IDLE. The user must make sure that the access pattern to the FLASHC HSB interface contains an IDLE cycle so that the command is allowed to start. Make sure that no bus masters such as DMA controllers are performing endless burst transfers from the flash. Also, make sure that the CPU does not perform endless burst transfers from flash. This is done by letting the CPU enter sleep mode after writing to FCMD, or by polling FSR for command completion. This polling will result in an access pattern with IDLE HSB cycles.

All the commands are protected by the same keyword, which has to be written in the eight highest bits of the FCMD register. Writing FCMD with data that does not contain the correct key and/or with an invalid command has no effect on the flash memory; however, the PROGE bit is set in the Flash Status Register (FSR). This bit is automatically cleared by a read access to the FSR register.

Writing a command to FCMD while another command is being executed has no effect on the flash memory; however, the PROGE bit is set in the Flash Status Register (FSR). This bit is automatically cleared by a read access to the FSR register.

If the current command writes or erases a page in a locked region, or a page protected by the BOOTPROT fuses, the command has no effect on the flash memory; however, the LOCKE bit is set in the FSR register. This bit is automatically cleared by a read access to the FSR register.

15.5.1 Write/Erase Page Operation

Flash technology requires that an erase must be done before programming. The entire flash can be erased by an Erase All command. Alternatively, pages can be individually erased by the Erase Page command.

The User page can be written and erased using the mechanisms described in this chapter.

After programming, the page can be locked to prevent miscellaneous write or erase sequences. Locking is performed on a per-region basis, so locking a region locks all pages inside the region. Additional protection is provided for the lowermost address space of the flash. This address space is allocated for the Boot Loader, and is protected both by the lock bit(s) corresponding to this address space, and the BOOTPROT[2:0] fuses.

Data to be written are stored in an internal buffer called page buffer. The page buffer contains *w* words. The page buffer wraps around within the internal memory area address space and appears to be repeated by the number of pages in it. Writing of 8-bit and 16-bit data to the page buffer is not allowed and may lead to unpredictable data corruption.

Data must be written to the page buffer before the programming command is written to the Flash Command Register FCMD. The sequence is as follows:

- Reset the page buffer with the Clear Page Buffer command.
- Fill the page buffer with the desired contents as described in Section 15.4.8 on page 257.
- Programming starts as soon as the programming key and the programming command are written to the Flash Command Register. The PAGEN field in the Flash Command Register (FCMD) must contain the address of the page to write. PAGEN is automatically updated when writing to the page buffer, but can also be written to directly. The FRDY bit in the Flash Status Register (FSR) is automatically cleared when the page write operation starts.
- When programming is completed, the bit FRDY in the Flash Status Register (FSR) is set. If an interrupt was enabled by writing a one to FCR.FRDY, the interrupt line of the Flash Controller is set.

Two errors can be detected in the FSR register after a programming sequence:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.
- Lock Error: Can have two different causes:
  - The page to be programmed belongs to a locked region. A command must be executed to unlock the corresponding region before programming can start.
  - A bus master without secure status attempted to program a page requiring secure privileges.

15.5.2 Erase All Operation

The entire memory is erased if the Erase All command (EA) is written to the Flash Command Register (FCMD). Erase All erases all bits in the flash array. The User page is not erased. All flash memory locations, the general-purpose fuse bits, and the security bit are erased (reset to 0xFF) after an Erase All.

The EA command also ensures that all volatile memories, such as register file and RAMs, are erased before the security bit is erased.

Erase All operation is allowed only if no regions are locked, and the BOOTPROT fuses are configured with a BOOTPROT region size of 0. Thus, if at least one region is locked, the bit LOCKE in FSR is set and the command is cancelled. If the bit LOCKE has been written to 1 in FCR, the interrupt request line is set.

When the command is complete, the FRDY bit in the Flash Status Register (FSR) is set. If an interrupt has been enabled by writing a one to FCR.FRDY, the interrupt line of the Flash Controller is set. Two errors can be detected in the FSR register after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.
- Lock Error: At least one lock region is protected, or BOOTPROT is different from 0. The erase command has been aborted and no page has been erased. A “Unlock region containing given page” (UP) command must be executed to unlock any locked regions.

15.5.3 Region Lock Bits

The flash memory has *p* pages, and these pages are grouped into 16 lock regions, each region containing *p*/16 pages. Each region has a dedicated lock bit preventing writing and erasing pages in the region. After production, the device may have some regions locked. These locked regions are reserved for a boot or default application. Locked regions can be unlocked to be erased and then programmed with another application or other data.

To lock or unlock a region, the commands Lock Region Containing Page (LP) and Unlock Region Containing Page (UP) are provided. Writing one of these commands, together with the number of the page whose region should be locked/unlocked, performs the desired operation.

One error can be detected in the FSR register after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.

The lock bits are implemented using the lowest 16 general-purpose fuse bits. This means that lock bits can also be set/cleared using the commands for writing/erasing general-purpose fuse bits, see Section 15.6. The general-purpose bit being in an erased (1) state means that the region is unlocked.

The lowermost pages in the Flash can additionally be protected by the BOOTPROT fuses, see Section 15.6.

15.6 General-purpose fuse bits

The flash memory has a number of general-purpose fuse bits that the application programmer can use freely. The fuse bits can be written and erased using dedicated commands, and read

through a dedicated Peripheral Bus address. Some of the general-purpose fuse bits are reserved for special purposes, and should not be used for other functions.:

Table 15-2. General-purpose Fuses with Special Functions

General-purpose fuse number	Name	Usage
15:0	LOCK	Region lock bits.
16	EPFL	External Privileged Fetch Lock. Used to prevent the CPU from fetching instructions from external memories when in privileged mode. This bit can only be changed when the security bit is cleared. The address range corresponding to external memories is device-specific, and not known to the Flash Controller. This fuse bit is simply routed out of the CPU or bus system, the Flash Controller does not treat this fuse in any special way, except that it can not be altered when the security bit is set. If the security bit is set, only an external JTAG Chip Erase can clear EPFL. No internal commands can alter EPFL if the security bit is set. When the fuse is erased (i.e. "1"), the CPU can execute instructions fetched from external memories. When the fuse is programmed (i.e. "0"), instructions can not be executed from external memories. This fuse has no effect in devices with no External Memory Interface (EBI).
19:17	BOOTPROT	Used to select one of eight different bootloader sizes. Pages included in the bootloader area can not be erased or programmed except by a JTAG chip erase. BOOTPROT can only be changed when the security bit is cleared. If the security bit is set, only an external JTAG Chip Erase can clear BOOTPROT, and thereby allow the pages protected by BOOTPROT to be programmed. No internal commands can alter BOOTPROT or the pages protected by BOOTPROT if the security bit is set.
21:20	SECURE	Used to configure secure state and secure state debug capabilities. Refer to the AVR32 Architecture Manual and the AVR32UC Technical Reference Manual for more details.
22	UPROT	If programmed (i.e. “1”), the JTAG USER PROTECTION feature is enabled. If this fuse is programmed some HSB addresses will be accessible by JTAG access even if the flash security fuse is programmed. Refer to the JTAG documentation for more information on this functionality. This bit can only be changed when the security bit is cleared.

The BOOTPROT fuses protects the following address space for the Boot Loader:

Table 15-3. Boot Loader Area Specified by BOOTPROT

BOOTPROT	Pages protected by BOOTPROT	Size of protected memory
7	None	0
6	0-1	1kByte
5	0-3	2kByte
4	0-7	4kByte
3	0-15	8kByte
2	0-31	16kByte
1	0-63	32kByte
0	0-127	64kByte

The SECURE fuses have the following functionality:

Table 15-4. Secure state configuration

SECURE	Functionality
00	Secure state disabled
01	Secure enabled, secure state debug enabled
10	Secure enabled, secure state debug disabled
11	Secure state disabled

To erase or write a general-purpose fuse bit, the commands Write General-Purpose Fuse Bit (WGPB) and Erase General-Purpose Fuse Bit (EGPB) are provided. Writing one of these commands, together with the number of the fuse to write/erase, performs the desired operation.

An entire general-purpose fuse byte can be written at a time by using the Program GP Fuse Byte (PGPFB) instruction. A PGPFB to GP fuse byte 2 is not allowed if the flash is locked by the security bit. The PFB command is issued with a parameter in the PAGEN field:

- PAGEN[2:0] - byte to write
- PAGEN[10:3] - Fuse value to write

All general-purpose fuses can be erased by the Erase All General-Purpose fuses (EAGP) command. An EAGP command is not allowed if the flash is locked by the security bit.

Two errors can be detected in the FSR register after issuing these commands:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.
- Lock Error:
  - A write or erase of the BOOTPROT or EPFL or UPROT fuse bits was attempted while the flash is locked by the security bit.
  - A write or erase of the SECURE fuse bits was attempted when SECURE mode was enabled.

The lock bits are implemented using the lowest 16 general-purpose fuse bits. This means that the 16 lowest general-purpose fuse bits can also be written/erased using the commands for locking/unlocking regions, see Section 15.5.3.

15.7 Security bit

The security bit allows the entire chip to be locked from external JTAG or other debug access for code security. The security bit can be written by a dedicated command, Set Security Bit (SSB). Once set, the only way to clear the security bit is through the JTAG Chip Erase command.

Once the Security bit is set, the following Flash controller commands will be unavailable and return a lock error if attempted:

- Write General-Purpose Fuse Bit (WGPB) to BOOTPROT or EPFL fuses
- Erase General-Purpose Fuse Bit (EGPB) to BOOTPROT or EPFL fuses
- Program General-Purpose Fuse Byte (PGPFB) of fuse byte 2
- Erase All General-Purpose Fuses (EAGPF)

One error can be detected in the FSR register after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.

15.8 User interface

Table 15-5. FLASHC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x0	Flash Control Register	FCR	Read/Write	0x00000000
0x4	Flash Command Register	FCMD	Read/Write	0x00000000
0x8	Flash Status Register	FSR	Read/Write	0(1)
0xc	Flash Parameter Register	PR	Read-only	0(2)
0x10	Flash Version Register	VR	Read-only	0(2)
0x14	Flash General Purpose Fuse Register Hi	FGPFRHI	Read-only	NA(1)
0x18	Flash General Purpose Fuse Register Lo	FGPFRLO	Read-only	NA(1)

Note: 1. The value of the Lock bits is dependent of their programmed state. All other bits in FSR are 0. All bits in FGPFR are dependent on the programmed state of the fuses they map to. Any bits in these registers not mapped to a fuse read 0.

2. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

15.8.1 Flash Control Register

Name: FCR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	FWS	-	-	PROGE	LOCKE	-	FRDY

- **FWS: Flash Wait State**
  0: The flash is read with 0 wait states.
  1: The flash is read with 1 wait state.

- **PROGE: Programming Error Interrupt Enable**
  0: Programming Error does not generate an interrupt.
  1: Programming Error generates an interrupt.
- **LOCKE: Lock Error Interrupt Enable**
  0: Lock Error does not generate an interrupt.
  1: Lock Error generates an interrupt.
- **FRDY: Flash Ready Interrupt Enable**
  0: Flash Ready does not generate an interrupt.
  1: Flash Ready generates an interrupt.

15.8.2 Flash Command Register

Name: FCMD

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

The FCMD can not be written if the flash is in the process of performing a flash command. Doing so will cause the FCR write to be ignored, and the PROGE bit to be set.

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
PAGEN [15:8]							

15	14	13	12	11	10	9	8
PAGEN [7:0]							

7	6	5	4	3	2	1	0
-	-	CMD					

- **KEY: Write Protection Key**
  This field should be written with the value 0xA5 to enable the command defined by the bits of the register. If the field is written with a different value, the write is not performed and no action is started.
  This field always reads as 0.
- **PAGEN: Page Number**
  The PAGEN field is used to address a page or fuse bit for certain operations. In order to simplify programming, the PAGEN field is automatically updated every time the page buffer is written to. For every page buffer write, the PAGEN field is updated with the page number of the address being written to. Hardware automatically masks writes to the PAGEN field so that only bits representing valid page numbers can be written, all other bits in PAGEN are always 0. As an example, in a flash with 1024 pages (page 0 - page 1023), bits 15:10 will always be 0.

Table 15-6. Semantic of PAGEN Field in Different Commands

Command	PAGEN description
No operation	Not used
Write Page	The number of the page to write
Clear Page Buffer	Not used
Lock region containing given Page	Page number whose region should be locked
Unlock region containing given Page	Page number whose region should be unlocked
Erase All	Not used
Write General-Purpose Fuse Bit	GPFUSE #
Erase General-Purpose Fuse Bit	GPFUSE #
Set Security Bit	Not used

Table 15-6. Semantic of PAGEN Field in Different Commands

Command	PAGEN description
Program GP Fuse Byte	WriteData[7:0], ByteAddress[2:0]
Erase All GP Fuses	Not used
Quick Page Read	Page number
Write User Page	Not used
Erase User Page	Not used
Quick Page Read User Page	Not used
High Speed Mode Enable	Not used
High Speed Mode Disable	Not used

- **CMD: Command**
  This field defines the flash command. Issuing any unused command will cause the Programming Error bit to be set, and the corresponding interrupt to be requested if the PROGE bit in FCR is set.

Table 15-7. Set of Commands

Command	Value	Mnemonic
No operation	0	NOP
Write Page	1	WP
Erase Page	2	EP
Clear Page Buffer	3	CPB
Lock region containing given Page	4	LP
Unlock region containing given Page	5	UP
Erase All	6	EA
Write General-Purpose Fuse Bit	7	WGPB
Erase General-Purpose Fuse Bit	8	EGPB
Set Security Bit	9	SSB
Program GP Fuse Byte	10	PGPFB
Erase All GPFuses	11	EAGPF
Quick Page Read	12	QPR
Write User Page	13	WUP
Erase User Page	14	EUP
Quick Page Read User Page	15	QPRUP
High Speed Mode Enable	16	HSEN
High Speed Mode Disable	17	HSDIS

15.8.3 Flash Status Register

Name: FSR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8
23	22	21	20	19	18	17	16
LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	HSMODE	QPRR	SECURITY	PROGE	LOCKE	-	FRDY

- **LOCKx: Lock Region x Lock Status**
  0: The corresponding lock region is not locked.
  1: The corresponding lock region is locked.
- **HSMODE: High-Speed Mode**
  0: High-speed mode disabled.
  1: High-speed mode enabled.
- **QPRR: Quick Page Read Result**
  0: The result is zero, i.e. the page is not erased.
  1: The result is one, i.e. the page is erased.
- **SECURITY: Security Bit Status**
  0: The security bit is inactive.
  1: The security bit is active.
- **PROGE: Programming Error Status**
  Automatically cleared when FSR is read.
  0: No invalid commands and no bad keywords were written in the Flash Command Register FCMD.
  1: An invalid command and/or a bad keyword was/were written in the Flash Command Register FCMD.
- **LOCKE: Lock Error Status**
  Automatically cleared when FSR is read.
  0: No programming of at least one locked lock region has happened since the last read of FSR.
  1: Programming of at least one locked lock region has happened since the last read of FSR.
- **FRDY: Flash Ready Status**
  0: The Flash Controller is busy and the application must wait before running a new command.
  1: The Flash Controller is ready to run a new command.

15.8.4 Parameter Register

Name: PR

Access Type: Read-only

Offset: 0x0C

Reset Value: 0x00000000(*)

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	PSZ		

7	6	5	4	3	2	1	0
-	-	-	-	FSZ			

- **FSZ: Flash Size**
  The size of the flash. Not all device families will provide all flash sizes indicated in the table.

Table 15-8. Flash size

FSZ	Flash Size	FSZ	Flash Size
0	4 Kbyte	8	192 Kbyte
1	8 Kbyte	9	256 Kbyte
2	16 Kbyte	10	384 Kbyte
3	32 Kbyte	11	512 Kbyte
4	48 Kbyte	12	768 Kbyte
5	64 Kbyte	13	1024 Kbyte
6	96 Kbyte	14	2048 Kbyte
7	128 Kbyte	15	RESERVED

- **PSZ: Page Size**
  The size of a flash page.

Table 15-9. Flash page size

PSZ	Page Size
0	32 words
1	64 words
2	128 words
3	256 words
4	512 words
5	1024 words
6	2048 words
7	4096 words

15.8.5 Version Register

Name: VR

Access Type: Read-only

Offset: 0x10

Reset Value: 0x00000000(*)

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]-			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

15.8.6 Flash General Purpose Fuse Register High

Name: FGPFRHI

Access Type: Read-only

Offset: 0x14

Reset Value: NA(*)

31	30	29	28	27	26	25	24
GPF63	GPF62	GPF61	GPF60	GPF59	GPF58	GPF57	GPF56

23	22	21	20	19	18	17	16
GPF55	GPF54	GPF53	GPF52	GPF51	GPF50	GPF49	GPF48

15	14	13	12	11	10	9	8
GPF47	GPF46	GPF45	GPF44	GPF43	GPF42	GPF41	GPF40

7	6	5	4	3	2	1	0
GPF39	GPF38	GPF37	GPF36	GPF35	GPF34	GPF33	GPF32

This register is only used in systems with more than 32 GP fuses.

- **GPFxx: General Purpose Fuse xx**
  0: The fuse has a written/programmed state.
  1: The fuse has an erased state.

15.8.7 Flash General Purpose Fuse Register Low

Name: FGPFRLO

Access Type: Read-only

Offset: 0x18

Reset Value: NA(*)

31	30	29	28	27	26	25	24
GPF31	GPF30	GPF29	GPF28	GPF27	GPF26	GPF25	GPF24

23	22	21	20	19	18	17	16
GPF23	GPF22	GPF21	GPF20	GPF19	GPF18	GPF17	GPF16

15	14	13	12	11	10	9	8
GPF15	GPF14	GPF13	GPF12	GPF11	GPF10	GPF09	GPF08

7	6	5	4	3	2	1	0
GPF07	GPF06	GPF05	GPF04	GPF03	GPF02	GPF01	GPF00

- **GPFxx: General Purpose Fuse xx**
  0: The fuse has a written/programmed state.
  1: The fuse has an erased state.

15.9 Fuses Settings

The flash block contains 32 general purpose fuses. These 32 fuses can be found in the Flash General Purpose Fuse Register Low. Some of these fuses have defined meanings outside the flash controller and are described in this section.

In addition to the General Purpose fuses parts of the Flash, user page can have a defined meaning outside the flash controller and are described in this section. Note that after writing the user page, values will not be propagated to other modules internally on the device. Other modules will load these values after resetting the chip.

The general purpose fuses are erased by a JTAG or aWire chip erase.

15.9.1 Flash General Purpose Fuse Register (FGPFRLO)

Table 15-10. FGPFR Register Description

31	30	29	28	27	26	25	24
BOD33EN		BODEN		BODHYST	BODLEVEL[3:1]		

23	22	21	20	19	18	17	16
BODLEVEL[0]	UPROT	SECURE		BOOTPROT			EPFL

15	14	13	12	11	10	9	8
LOCK[15:8]							

7	6	5	4	3	2	1	0
LOCK[7:0]							

- **BOD33EN: 3.3V Brown Out Detector Enable**

Table 15-11. BOD33EN Field Description

BOD33EN	Description
0x0	BOD33 disabled
0x1	BOD33 enabled, BOD33 reset enabled
0x2	BOD33 enabled, BOD33 reset disabled
0x3	BOD33 disabled

- **BODEN: 1.8V Brown Out Detector Enable**

Table 15-12. BODEN Field Description

BODEN	Description
0x0	BOD18 disabled
0x1	BOD18 enabled, BOD18 reset enabled
0x2	BOD18 enabled, BOD18 reset disabled
0x3	BOD18 disabled

- **BODHYST: 1.8V Brown Out Detector Hysteresis**
  0: The BOD18 hysteresis is disabled.
  1: The BOD18 hysteresis is enabled.

- **BODLEVEL: 1.8V Brown Out Detector Trigger Level**
  This controls the voltage trigger level for the BOD18.
  When the flash fuse calibration is done (SCIF.BOD.FCD is set):
  SCIF.BOD.LEVEL is loaded to 0x28 if the BODLEVEL fuses are greater than 0xA.
  SCIF.BOD.LEVEL is loaded to (BODLEVEL x 4) if the BODLEVEL fuses are lower or equal to 0xA.
  Refer to See "Electrical Characteristics" on page 1249.

- **UPROT, SECURE, BOOTPROT, EPFL, LOCK**
  These are Flash controller fuses and are described in the FLASHC section.

15.9.2 Default Fuse Value

The devices are shipped with the FGPFRLO register value: 0xF877FFFF:

- BOD33EN fuses set to 11. BOD33 is disabled.
- BODEN fuses set to 11. BOD18 is disabled.
- BODHYST fuse set to 1. The BOD18 hysteresis is enabled.
- BODLEVEL fuses set to 0000. This is the minimum voltage level for BOD18.
- UPROT fuse set to 1
- SECURE fuses set to 11
- BOOTPROT fuses set to 011. The bootloader protected size is 8 KBytes.
- EPFL fuse set to 1. External privileged fetch is not locked.
- LOCK fuses set to 1111111111111111. No region locked.

See also the AT32UC3C Bootloader user guide document.

After the JTAG or aWire chip erase command, the FGPFRLO register value is 0xFFFFFFFF.

15.9.3 Fuses in User Page (address 0x80800000)

15.9.3.1 First word (address 0x80800000)

Table 15-13. User Page Fuse Description

31	30	29	28	27	26	25	24
WDTDISRV	SS_ADRR[14:8]						

23	22	21	20	19	18	17	16
SS_ADRR[7:0]							

15	14	13	12	11	10	9	8
SS_ADRF[15:8]							

7	6	5	4	3	2	1	0
SS_ADRF[7:0]							

- **SS_ADRR: Size of the CPU RAM controlled by the Secure State**
  The section of the CPU RAM controlled by the Secure State is from address 0x00000000 to address (SS_ADRR << 10).

- **SS_ADRF: Size of the Flash controlled by the Secure State**
  The section of the Flash controlled by the Secure State is from address 0x80000000 to address (SS_ADRF << 10).

- **WDTDISRV: WatchDog Timer auto disable at startup**
  0: The WDT is automatically enabled at startup, the WDTAUTO fuse of the WDT is set.
  1: The WDT is not automatically enabled at startup, the WDTAUTO fuse of the watchdog timer is not set.
  Please refer to the WDT chapter for detail about time-out settings when the WDT is automatically enabled.

The devices are shipped with the User page erased (all bits 1).

15.9.4 Bootloader Configuration

The USB/USART bootloader uses two words in the flash user page to store its configuration:

- Configuration word 1 at address 0x808001FC is read first at boot time to know if it should start the ISP process unconditionally and whether it should use the configuration word 2 where further configuration is stored.
- Configuration word 2 at address 0x808001F8 stores the I/O conditions that determine which of the USB DFU ISP and the application to start at the end of the boot process.

Please refer to the bootloader documentation for more information.

15.10 Calibration Settings

Some analog blocks require to be calibrated. The recommended calibration settings are written in the factory page. The base address of the factory page is 0x80800200.

Table 15-14. Calibration Register Map

Offset	Register
0x0000	Oscillator Calibration
0x0004	ADC Core Calibration
0x0008	ADC S/H Calibration
0x000C	DAC0A Channel Calibration
0x0010	DAC0B Channel Calibration
0x0014	DAC1A Channel Calibration
0x0018	DAC1B Channel Calibration

15.10.0.1 Oscillator Calibration (offset 0x0000)

Table 15-15. Oscillator Calibration

31	30	29	28	27	26	25	24
RC1M_CALIB_5V[7:0]							

23	22	21	20	19	18	17	16
RC8M_CALIB_5V[7:0]							

15	14	13	12	11	10	9	8
RC1M_CALIB[7:0]							

7	6	5	4	3	2	1	0
RC8M_CALIB[7:0]							

- **RC1M_CALIB_5V: Calibration of RC8M operating at 1MHz and at 5V**
  This calibration should be used when the RC8M is used at a frequency of 1 MHz and when the voltage of the VDDIN_5 pin is within [4.5V:5.5V]. This value should be written to RCCR8.CALIB field of the SCIF module.

- **RC8M_CALIB_5V: Calibration of RC8M operating at 8MHz and at 5V**
  This calibration should be used when the RC8M is used at a frequency of 8 MHz and when the voltage of the VDDIN_5 pin is within [4.5V:5.5V]. This value should be written to RCCR8.CALIB.

- **RC1M_CALIB: Calibration of RC8M operating at 1MHz and at 3.3V**
  This calibration should be used when the RC8M is used at a frequency of 1 MHz and when the voltage of the VDDIN_5 pin is within [3.0V:3.6V]. This value should be written to RCCR8.CALIB.

- **RC8M_CALIB: Calibration of RC8M operating at 8MHz and at 3.3V**
  This calibration should be used when the RC8M is used at a frequency of 8 MHz and when the voltage of the VDDIN_5 pin is within [3.0V:3.6V]. This value is automatically loaded in RCCR8.CALIB at power-up.

15.10.0.2 ADC Core Calibration (offset 0x0004)

Table 15-16. ADC Core Calibration

31	30	29	28	27	26	25	24
-	-	ADC_OCAL[5:0]					

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
	ADC_GCAL[14:8]						

7	6	5	4	3	2	1	0
ADC_GCAL[7:0]							

- **ADC_OCAL: Offset Calibration of the ADC core**
  This value should be written to the ADCCAL.OCAL field of the ADCIFA module.

- **ADC_GCAL: Gain Calibration of the ADC core**
  This value should be written to the ADCCAL.GCAL field of the ADCIFA module.

15.10.0.3 ADC S/H Calibration (offset 0x0008)

Table 15-17. ADC S/H Calibration

31	30	29	28	27	26	25	24
-	-	-	-	-	-	ADC_GAIN1[9:8]	

23	22	21	20	19	18	17	16
ADC_GAIN1[7:0]							

15	14	13	12	11	10	9	8
-	-	-	-	-	-	ADC_GAIN0[9:8]	

7	6	5	4	3	2	1	0
ADC_GAIN0[7:0]							

- **ADC_GAIN1: Gain Calibration of the ADC S/H1**
  This value should be written to the SHCAL.GAIN1 field of the ADCIFA module.

- **ADC_GAIN0: Gain Calibration of the ADC S/H0**
  This value should be written to the SHCAL.GAIN0 field of the ADCIFA module.

15.10.0.4 DAC0 Channel Calibration (offset 0x000C (DAC0 Channel A), 0x0010 (DAC0 Channel B)),

Table 15-18. DAC0A and DAC0B Channel Calibration

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	DAC_GCAL[8]

23	22	21	20	19	18	17	16
DAC_GCAL[7:0]							

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	DAC_OCAL[8]

7	6	5	4	3	2	1	0
DAC_OCAL[7:0]							

- **DAC_GCAL: Gain Calibration of the DAC Channel**
  This value should be written to the GOC.GCR field of the DACIFB0 module.

- **DAC_OCAL: Offset Calibration of the DAC Channel**
  This value should be written to GOC.OCR field of the DACIFB0 module.

15.10.0.5 DAC1 Channel Calibration (offset 0x0014 (DAC1 Channel A), 0x0018 (DAC1 Channel B)),

Table 15-19. DAC1A and DAC1B Channel Calibration

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	DAC_GCAL[8]

23	22	21	20	19	18	17	16
DAC_GCAL1[7:0]							

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	DAC_OCAL[8]

7	6	5	4	3	2	1	0
DAC_OCAL[7:0]							

- **DAC_GCAL: Gain Calibration of the DAC Channel**
  This value should be written to the GOC.GCR field of the DACIFB1 module.

- **DAC_OCAL: Offset Calibration of the DAC Channel**
  This value should be written to GOC.OCR field of the DACIFB1 module.

15.11 Serial Number

Each device has a unique 120 bits serial number readable from address 0x80800284 to 0x80800292.

15.12 Module Configuration

The specific configuration for each FLASHC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 15-20. Module Configuration

	Flash Size	Number of pages	Page size
AT32UC3C0512C **AT32UC3C1512C** **AT32UC3C2512C**	512Kbytes	1024	128 words
AT32UC3C0256C **AT32UC3C1256C** **AT32UC3C2256C**	256Kbytes	512	128 words
AT32UC3C0128C **AT32UC3C1128C** **AT32UC3C2128C**	128Kbytes	256	128 words
AT32UC3C064C **AT32UC3C164C** **AT32UC3C264C**	64Kbytes	128	128 words

Table 15-21. Module Clock Name

Module name	Clock Name	Description
FLASHC	CLK_FLASHC_HSB	HSB clock
	CLK_FLASHC_PB	Peripheral Bus clock from the PBB clock domain

Table 15-22. Register Reset Values

Register	Reset Value
PR	Refer to Table 15-20
VR	0x00000302

16. HSB Bus Matrix (HMATRIXB)

Rev: 1.3.0.3

16.1 Features

- **User Interface on peripheral bus**
- **Configurable number of masters (up to 16)**
- **Configurable number of slaves (up to 16)**
- **One decoder for each master**
- **Programmable arbitration for each slave**
  - **Round-Robin**
  - **Fixed priority**
- **Programmable default master for each slave**
  - **No default master**
  - **Last accessed default master**
  - **Fixed default master**
- **One cycle latency for the first access of a burst**
- **Zero cycle latency for default master**
- **One special function register for each slave (not dedicated)**

16.2 Overview

The Bus Matrix implements a multi-layer bus structure, that enables parallel access paths between multiple High Speed Bus (HSB) masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects up to 16 HSB Masters to up to 16 HSB Slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency). The Bus Matrix provides 16 Special Function Registers (SFR) that allow the Bus Matrix to support application specific features.

16.3 Product Dependencies

In order to configure this module by accessing the user registers, other parts of the system must be configured correctly, as described below.

16.3.1 Clocks

The clock for the HMATRIX bus interface (CLK_HMATRIX) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager.

16.4 Functional Description

16.4.1 Special Bus Granting Mechanism

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from some masters. This mechanism reduces latency at first access of a burst or single transfer. This bus granting mechanism sets a different default master for every slave.

At the end of the current access, if no other request is pending, the slave remains connected to its associated default master. A slave can be associated with three kinds of default masters: no default master, last access master, and fixed default master.

To change from one kind of default master to another, the Bus Matrix user interface provides the Slave Configuration Registers, one for each slave, that set a default master for each slave. The Slave Configuration Register contains two fields: DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field selects the default master type (no default, last access master, fixed default master), whereas the 4-bit FIXED_DEFMSTR field selects a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Please refer to the Bus Matrix user interface description.

16.4.1.1 No Default Master

At the end of the current access, if no other request is pending, the slave is disconnected from all masters. No Default Master suits low-power mode.

16.4.1.2 Last Access Master

At the end of the current access, if no other request is pending, the slave remains connected to the last master that performed an access request.

16.4.1.3 Fixed Default Master

At the end of the current access, if no other request is pending, the slave connects to its fixed default master. Unlike last access master, the fixed master does not change unless the user modifies it by a software action (field FIXED_DEFMSTR of the related SCFG).

16.4.2 Arbitration

The Bus Matrix provides an arbitration mechanism that reduces latency when conflict cases occur, i.e. when two or more masters try to access the same slave at the same time. One arbiter per HSB slave is provided, thus arbitrating each slave differently.

The Bus Matrix provides the user with the possibility of choosing between 2 arbitration types for each slave:

1. Round-Robin Arbitration (default)
2. Fixed Priority Arbitration

This is selected by the ARBT field in the Slave Configuration Registers (SCFG).

Each algorithm may be complemented by selecting a default master configuration for each slave.

When a re-arbitration must be done, specific conditions apply. This is described in “Arbitration Rules” .

16.4.2.1 Arbitration Rules

Each arbiter has the ability to arbitrate between two or more different master requests. In order to avoid burst breaking and also to provide the maximum throughput for slave interfaces, arbitration may only take place during the following cycles:

1. Idle Cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single Cycles: When a slave is currently doing a single access.
3. End of Burst Cycles: When the current cycle is the last cycle of a burst transfer. For defined length burst, predicted end of burst matches the size of the transfer but is managed differently for undefined length burst. This is described below.
4. Slot Cycle Limit: When the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. This is described below.

- Undefined Length Burst Arbitration

In order to avoid long slave handling during undefined length bursts (INCR), the Bus Matrix provides specific logic in order to re-arbitrate before the end of the INCR transfer. A predicted end of burst is used as a defined length burst transfer and can be selected among the following five possibilities:

1. Infinite: No predicted end of burst is generated and therefore INCR burst transfer will never be broken.
2. One beat bursts: Predicted end of burst is generated at each single transfer inside the INCP transfer.
3. Four beat bursts: Predicted end of burst is generated at the end of each four beat boundary inside INCR transfer.
4. Eight beat bursts: Predicted end of burst is generated at the end of each eight beat boundary inside INCR transfer.
5. Sixteen beat bursts: Predicted end of burst is generated at the end of each sixteen beat boundary inside INCR transfer.

This selection can be done through the ULBT field in the Master Configuration Registers (MCFG).

- Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At the beginning of the burst access, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (SCFG) and decreased at each clock cycle. When the counter reaches zero, the arbiter has the ability to re-arbitrate at the end of the current byte, halfword, or word transfer.

16.4.2.2 Round-Robin Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave in a round-robin manner. If two or more master requests arise at the same time, the master with the lowest number is first serviced, then the others are serviced in a round-robin manner.

There are three round-robin algorithms implemented:

1. Round-Robin arbitration without default master
2. Round-Robin arbitration with last default master
3. Round-Robin arbitration with fixed default master

- Round-Robin Arbitration without Default Master

This is the main algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to dispatch requests from different masters to the same slave in a pure round-robin manner. At the end of the current access, if no other request is pending, the slave is disconnected from all masters. This configuration incurs one latency cycle for the first access of a burst. Arbitration without default master can be used for masters that perform significant bursts.

- Round-Robin Arbitration with Last Default Master

This is a biased round-robin algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. At the end of the cur-

rent transfer, if no other master request is pending, the slave remains connected to the last master that performed the access. Other non privileged masters still get one latency cycle if they want to access the same slave. This technique can be used for masters that mainly perform single accesses.

- Round-Robin Arbitration with Fixed Default Master

This is another biased round-robin algorithm. It allows the Bus Matrix arbiters to remove the one latency cycle for the fixed default master per slave. At the end of the current access, the slave remains connected to its fixed default master. Every request attempted by this fixed default master will not cause any latency whereas other non privileged masters will still get one latency cycle. This technique can be used for masters that mainly perform single accesses.

16.4.2.3 Fixed Priority Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user. If two or more master requests are active at the same time, the master with the highest priority number is serviced first. If two or more master requests with the same priority are active at the same time, the master with the highest number is serviced first.

For each slave, the priority of each master may be defined through the Priority Registers for Slaves (PRAS and PRBS).

16.4.3 Slave and Master assignation

The index number assigned to Bus Matrix slaves and masters are described in the Module Configuration section at the end of this chapter.

16.5 User Interface

Table 16-1. HMATRIX Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Master Configuration Register 0	MCFG0	Read/Write	0x00000002
0x0004	Master Configuration Register 1	MCFG1	Read/Write	0x00000002
0x0008	Master Configuration Register 2	MCFG2	Read/Write	0x00000002
0x000C	Master Configuration Register 3	MCFG3	Read/Write	0x00000002
0x0010	Master Configuration Register 4	MCFG4	Read/Write	0x00000002
0x0014	Master Configuration Register 5	MCFG5	Read/Write	0x00000002
0x0018	Master Configuration Register 6	MCFG6	Read/Write	0x00000002
0x001C	Master Configuration Register 7	MCFG7	Read/Write	0x00000002
0x0020	Master Configuration Register 8	MCFG8	Read/Write	0x00000002
0x0024	Master Configuration Register 9	MCFG9	Read/Write	0x00000002
0x0028	Master Configuration Register 10	MCFG10	Read/Write	0x00000002
0x002C	Master Configuration Register 11	MCFG11	Read/Write	0x00000002
0x0030	Master Configuration Register 12	MCFG12	Read/Write	0x00000002
0x0034	Master Configuration Register 13	MCFG13	Read/Write	0x00000002
0x0038	Master Configuration Register 14	MCFG14	Read/Write	0x00000002
0x003C	Master Configuration Register 15	MCFG15	Read/Write	0x00000002
0x0040	Slave Configuration Register 0	SCFG0	Read/Write	0x00000010
0x0044	Slave Configuration Register 1	SCFG1	Read/Write	0x00000010
0x0048	Slave Configuration Register 2	SCFG2	Read/Write	0x00000010
0x004C	Slave Configuration Register 3	SCFG3	Read/Write	0x00000010
0x0050	Slave Configuration Register 4	SCFG4	Read/Write	0x00000010
0x0054	Slave Configuration Register 5	SCFG5	Read/Write	0x00000010
0x0058	Slave Configuration Register 6	SCFG6	Read/Write	0x00000010
0x005C	Slave Configuration Register 7	SCFG7	Read/Write	0x00000010
0x0060	Slave Configuration Register 8	SCFG8	Read/Write	0x00000010
0x0064	Slave Configuration Register 9	SCFG9	Read/Write	0x00000010
0x0068	Slave Configuration Register 10	SCFG10	Read/Write	0x00000010
0x006C	Slave Configuration Register 11	SCFG11	Read/Write	0x00000010
0x0070	Slave Configuration Register 12	SCFG12	Read/Write	0x00000010
0x0074	Slave Configuration Register 13	SCFG13	Read/Write	0x00000010
0x0078	Slave Configuration Register 14	SCFG14	Read/Write	0x00000010
0x007C	Slave Configuration Register 15	SCFG15	Read/Write	0x00000010
0x0080	Priority Register A for Slave 0	PRAS0	Read/Write	0x00000000
0x0084	Priority Register B for Slave 0	PRBS0	Read/Write	0x00000000
0x0088	Priority Register A for Slave 1	PRAS1	Read/Write	0x00000000

Table 16-1. HMATRIX Register Memory Map (Continued)

Offset	Register	Name	Access	Reset Value
0x008C	Priority Register B for Slave 1	PRBS1	Read/Write	0x00000000
0x0090	Priority Register A for Slave 2	PRAS2	Read/Write	0x00000000
0x0094	Priority Register B for Slave 2	PRBS2	Read/Write	0x00000000
0x0098	Priority Register A for Slave 3	PRAS3	Read/Write	0x00000000
0x009C	Priority Register B for Slave 3	PRBS3	Read/Write	0x00000000
0x00A0	Priority Register A for Slave 4	PRAS4	Read/Write	0x00000000
0x00A4	Priority Register B for Slave 4	PRBS4	Read/Write	0x00000000
0x00A8	Priority Register A for Slave 5	PRAS5	Read/Write	0x00000000
0x00AC	Priority Register B for Slave 5	PRBS5	Read/Write	0x00000000
0x00B0	Priority Register A for Slave 6	PRAS6	Read/Write	0x00000000
0x00B4	Priority Register B for Slave 6	PRBS6	Read/Write	0x00000000
0x00B8	Priority Register A for Slave 7	PRAS7	Read/Write	0x00000000
0x00BC	Priority Register B for Slave 7	PRBS7	Read/Write	0x00000000
0x00C0	Priority Register A for Slave 8	PRAS8	Read/Write	0x00000000
0x00C4	Priority Register B for Slave 8	PRBS8	Read/Write	0x00000000
0x00C8	Priority Register A for Slave 9	PRAS9	Read/Write	0x00000000
0x00CC	Priority Register B for Slave 9	PRBS9	Read/Write	0x00000000
0x00D0	Priority Register A for Slave 10	PRAS10	Read/Write	0x00000000
0x00D4	Priority Register B for Slave 10	PRBS10	Read/Write	0x00000000
0x00D8	Priority Register A for Slave 11	PRAS11	Read/Write	0x00000000
0x00DC	Priority Register B for Slave 11	PRBS11	Read/Write	0x00000000
0x00E0	Priority Register A for Slave 12	PRAS12	Read/Write	0x00000000
0x00E4	Priority Register B for Slave 12	PRBS12	Read/Write	0x00000000
0x00E8	Priority Register A for Slave 13	PRAS13	Read/Write	0x00000000
0x00EC	Priority Register B for Slave 13	PRBS13	Read/Write	0x00000000
0x00F0	Priority Register A for Slave 14	PRAS14	Read/Write	0x00000000
0x00F4	Priority Register B for Slave 14	PRBS14	Read/Write	0x00000000
0x00F8	Priority Register A for Slave 15	PRAS15	Read/Write	0x00000000
0x00FC	Priority Register B for Slave 15	PRBS15	Read/Write	0x00000000
0x0110	Special Function Register 0	SFR0	Read/Write	–
0x0114	Special Function Register 1	SFR1	Read/Write	–
0x0118	Special Function Register 2	SFR2	Read/Write	–
0x011C	Special Function Register 3	SFR3	Read/Write	–
0x0120	Special Function Register 4	SFR4	Read/Write	–
0x0124	Special Function Register 5	SFR5	Read/Write	–
0x0128	Special Function Register 6	SFR6	Read/Write	–

Table 16-1. HMATRIX Register Memory Map (Continued)

Offset	Register	Name	Access	Reset Value
0x012C	Special Function Register 7	SFR7	Read/Write	–
0x0130	Special Function Register 8	SFR8	Read/Write	–
0x0134	Special Function Register 9	SFR9	Read/Write	–
0x0138	Special Function Register 10	SFR10	Read/Write	–
0x013C	Special Function Register 11	SFR11	Read/Write	–
0x0140	Special Function Register 12	SFR12	Read/Write	–
0x0144	Special Function Register 13	SFR13	Read/Write	–
0x0148	Special Function Register 14	SFR14	Read/Write	–
0x014C	Special Function Register 15	SFR15	Read/Write	–

16.5.1 Master Configuration Registers

Name: MCFG0...MCFG15

Access Type: Read/Write

Offset: 0x00 - 0x3C

Reset Value: 0x00000002

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

- **ULBT: Undefined Length Burst Type**

Table 16-2. Undefined Length Burst Type

ULBT	Undefined Length Burst Type	Description
000	Inifinite Length Burst	No predicted end of burst is generated and therefore INCR bursts coming from this master cannot be broken.
001	Single-Access	The undefined length burst is treated as a succession of single accesses, allowing re-arbitration at each beat of the INCR burst.
010	4 Beat Burst	The undefined length burst is split into a four-beat burst, allowing re-arbitration at each four-beat burst end.
011	8 Beat Burst	The undefined length burst is split into an eight-beat burst, allowing re-arbitration at each eight-beat burst end.
100	16 Beat Burst	The undefined length burst is split into a sixteen-beat burst, allowing re-arbitration at each sixteen-beat burst end.

16.5.2 Slave Configuration Registers

Name: SCFG0...SCFG15

Access Type: Read/Write

Offset: 0x40 - 0x7C

Reset Value: 0x00000010

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	ARBT

23	22	21	20	19	18	17	16
–	–	FIXED_DEFMSTR				DEFMSTR_TYPE	

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SLOT_CYCLE							

- **ARBT: Arbitration Type**
  0: Round-Robin Arbitration
  1: Fixed Priority Arbitration
- **FIXED_DEFMSTR: Fixed Default Master**
  This is the number of the Default Master for this slave. Only used if DEFMSTR_TYPE is 2. Specifying the number of a master which is not connected to the selected slave is equivalent to setting DEFMSTR_TYPE to 0.
- **DEFMSTR_TYPE: Default Master Type**
  0: No Default Master
  At the end of the current slave access, if no other master request is pending, the slave is disconnected from all masters.
  This results in a one cycle latency for the first access of a burst transfer or for a single access.
  1: Last Default Master
  At the end of the current slave access, if no other master request is pending, the slave stays connected to the last master having accessed it.
  This results in not having one cycle latency when the last master tries to access the slave again.
  2: Fixed Default Master
  At the end of the current slave access, if no other master request is pending, the slave connects to the fixed master the number that has been written in the FIXED_DEFMSTR field.
  This results in not having one cycle latency when the fixed master tries to access the slave again.
- **SLOT_CYCLE: Maximum Number of Allowed Cycles for a Burst**
  When the SLOT_CYCLE limit is reached for a burst, it may be broken by another master trying to access this slave.
  This limit has been placed to avoid locking a very slow slave when very long bursts are used.
  This limit must not be very small. Unreasonably small values break every burst and the Bus Matrix arbitrates without performing any data transfer. 16 cycles is a reasonable value for SLOT_CYCLE.

16.5.3 Bus Matrix Priority Registers A For Slaves

Register Name: PRAS0...PRAS15

Access Type: Read/Write

Offset: -

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	M7PR		-	-	M6PR	

23	22	21	20	19	18	17	16
-	-	M5PR		-	-	M4PR	

15	14	13	12	11	10	9	8
-	-	M3PR		-	-	M2PR	

7	6	5	4	3	2	1	0
-	-	M1PR		-	-	M0PR	

- **MxPR: Master x Priority**

  Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

16.5.4 Priority Registers B For Slaves

Name: PRBS0...PRBS15

Access Type: Read/Write

Offset: -

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	M15PR		-	-	M14PR	

23	22	21	20	19	18	17	16
-	-	M13PR		-	-	M12PR	

15	14	13	12	11	10	9	8
-	-	M11PR		-	-	M10PR	

7	6	5	4	3	2	1	0
-	-	M9PR		-	-	M8PR	

- **MxPR: Master x Priority**

  Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

16.5.5 Special Function Registers

Name: SFR0...SFR15

Access Type: Read/Write

Offset: 0x110 - 0x14C

Reset Value: -

31	30	29	28	27	26	25	24
SFR							

23	22	21	20	19	18	17	16
SFR							

15	14	13	12	11	10	9	8
SFR							

7	6	5	4	3	2	1	0
SFR							

- **SFR: Special Function Register Fields**

  Those registers are not a HMATRIX specific register. The field of those will be defined where they are used.

16.6 Bus Matrix Connections

Accesses to unused areas returns an error result to the master requesting such an access.

The bus matrix has the several masters and slaves. Each master has its own bus and its own decoder, thus allowing a different memory mapping per master. The master number in the table below can be used to index the HMATRIX control registers. For example, MCFG0 is associated with the CPU Data master interface.

Table 16-3. High Speed Bus masters

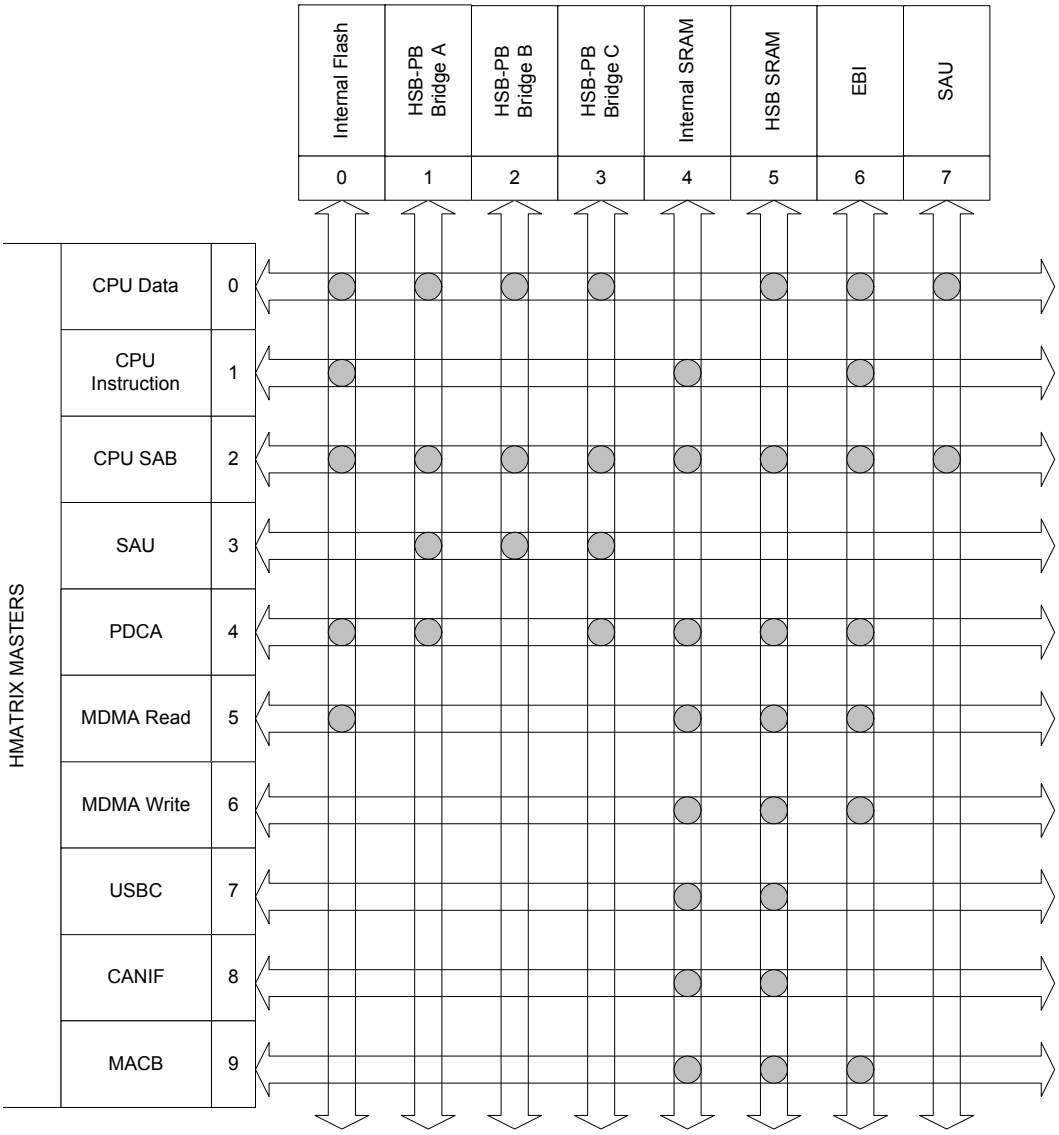
Master 0	CPU Data
Master 1	CPU Instruction
Master 2	CPU SAB
Master 3	SAU
Master 4	PDCA
Master 5	MDMA Read
Master 6	MDMA write
Master 7	USBC
Master 8	CANIF
Master 9	MACB

Each slave has its own arbiter, thus allowing a different arbitration per slave. The slave number in the table below can be used to index the HMATRIX control registers. For example, SCFG3 is associated with the Internal SRAM Slave Interface.

Table 16-4. High Speed Bus slaves

Slave 0	Internal Flash
Slave 1	HSB-PB Bridge A
Slave 2	HSB-PB Bridge B
Slave 3	HSB-PB Bridge C
Slave 4	Internal SRAM
Slave 5	HSB RAM
Slave 6	EBI
Slave 7	SAU

Figure 16-1. HMatrix Master / Slave Connections



17. External Bus Interface (EBI)

Rev.: 1.7.0.2

17.1 Features

- **Optimized for application memory space support**
- **Integrates two external memory controllers:**
  - **Static Memory Controller (SMC)**
  - **SDRAM Controller (SDRAMC)**
- **Optimized external bus:16(or 8)-bit data bus**
  - **Up to 24-bit Address Bus, Up to 16-Mbytes Addressable**
  - **Optimized pin multiplexing to reduce latencies on external memories**
- **Up to 4 Chip Selects, Configurable Assignment:**
  - **Static Memory Controller on Chip Select 0**
  - **SDRAM Controller or Static Memory Controller on Chip Select 1**
  - **Static Memory Controller on Chip Select 2**
  - **Static Memory Controller on Chip Select 3**

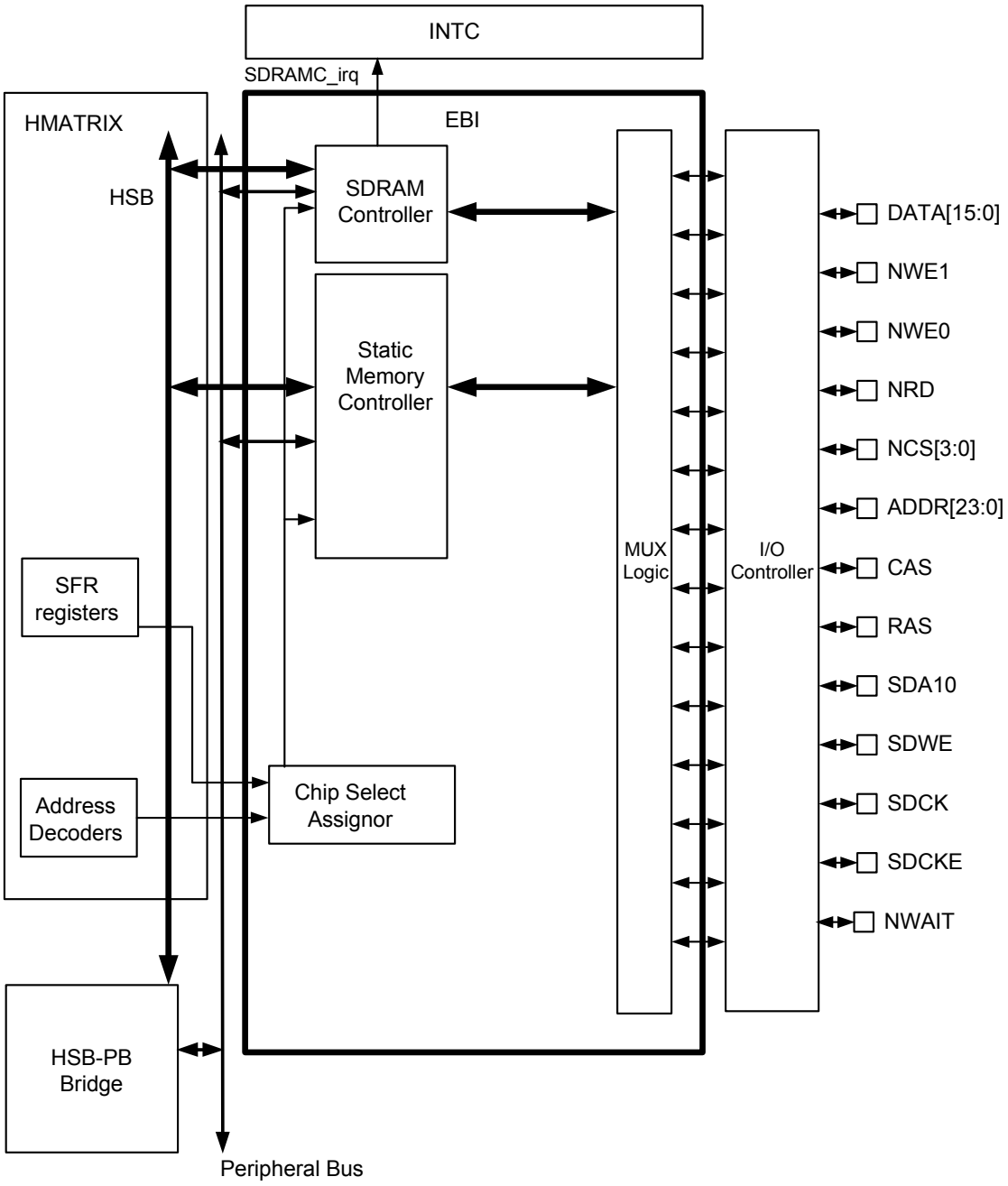
17.2 Overview

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded memory controller of a 32-bit AVR device. The Static Memory and SDRAM Controllers are all featured external memory controllers on the EBI. These external memory controllers are capable of handling several types of external memory and peripheral devices, such as SRAM, PROM, EPROM, EEPROM, Flash, and SDRAM.

The EBI handles data transfers with up to four external devices, each assigned to four address spaces defined by the embedded memory controller. Data transfers are performed through a 16-8)-bit data bus, an address bus of up to 24 bits, up to four chip select lines (NCS[3:0]), and several control pins that are generally multiplexed between the different external memory controllers.

17.3 Block Diagram

Figure 17-1. EBI Block Diagram



17.4 I/O Lines Description

Table 17-1. EBI I/O Lines Description

Pin Name	Alternate Name	Pin Description	Type	Active Level
EBI common lines				
DATA[15:0]		Data Bus	I/O	
SMC dedicated lines				
ADDR[1]		SMC Address Bus Line 1	Output	
ADDR[12]		SMC Address Bus Line 12	Output	
ADDR[15]		SMC Address Bus Line 15	Output	
ADDR[23:18]		SMC Address Bus Line [23:18]	Output	
NCS[0]		SMC Chip Select Line 0	Output	Low
NCS[2]		SMC Chip Select Line 2	Output	Low
NCS[3]		SMC Chip Select Line 3	Output	Low
NRD		SMC Read Signal	Output	Low
NWAIT		SMC External Wait Signal	Input	Low
NWE0	NWE0-NWE	SMC Write Enable1 or Write enable	Output	Low
SDRAMC dedicated lines				
SDCK		SDRAM Clock	Output	
SDCKE		SDRAM Clock Enable	Output	High
SDWE		SDRAM Write Enable	Output	Low
SDA10		SDRAM Address Bus Line 10	Output	Low
RAS - CAS		Row and Column Signal	Output	Low
SMC/SDRAMC shared lines				
ADDR[0]	DQM0 ADDR[0]-NBS0	SDRAMC DQM1 SMC Address Bus Line 0 or Byte Select 0	Output	
ADDR[11:2]	ADDR[9:0] ADDR[11:2]	SDRAMC Address Bus Lines [9:0] SMC Address Bus Lines [11:2]	Output	
ADDR[14:13]	ADDR[9:0] ADDR[14:13]	SDRAMC Address Bus Lines [12:11] SMC Address Bus Lines [14:13]	Output	
ADDR[16]	BA0 ADDR[16]	SDRAMC Bank 0 SMC Address Bus Line 16	Output	
ADDR[17]	BA1 ADDR[17]	SDRAMC Bank 1 SMCAddress Bus Line 17	Output	
NCS[1]	NCS[1] SDCS	SMC Chip Select Line 1 SDRAMC Chip Select	Output	Low
NWE1	DQM1 NWE1-NBS1	SDRAMC DQM1 SMC Write Enable1 or Byte Select 1	Output	

17.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

17.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with I/O Controller lines. The user must first configure the I/O Controller to assign the EBI pins to their peripheral functions.

17.5.2 Power Management

To prevent bus errors EBI operation must be terminated before entering sleep mode.

17.5.3 Clocks

A number of clocks can be selected as source for the EBI. The selected clock must be enabled by the Power Manager.

The following clock sources are available:

- CLK_EBI
- CLK_SDRAMC
- CLK_SMC

Refer to Table 17-2 on page 302 to configure those clocks.

Table 17-2. EBI Clocks Configuration

Clocks name	Clocks type	Type of the Interfaced Device	
		SDRAM	SRAM, PROM, EPROM, EEPROM, Flash
CLK_EBI	HSB	X	X
CLK_SDRAMC	PB	X	
CLK_SMC	PB		X

17.5.4 Interrupts

The EBI interface has one interrupt line connected to the Interrupt Controller:

- SDRAMC_IRQ: Interrupt signal coming from the SDRAMC

Handling the EBI interrupt requires configuring the interrupt controller before configuring the EBI.

17.5.5 HMATRIX

The EBI interface is connected to the HMATRIX Special Function Register 6 (SFR6). The user must first write to this HMATRIX.SFR6 to configure the EBI correctly.

Table 17-3. EBI Special Function Register Fields Description

SFR6 Bit Number	Bit name	Description
[31:2]		Reserved
1	CS1A	0 = Chip Select 1 (NCS[1]) is connected to a Static Memory device. For each access to the NCS[1] memory space, all related pins act as SMC pins 1 = Chip Select 1 (NCS[1]) is connected to a SDRAM device. For each access to the NCS[1] memory space, all related pins act as SDRAM pins (SDCS)
0		Reserved

17.6 Functional Description

The EBI transfers data between the internal HSB bus (handled by the HMATRIX) and the external memories or peripheral devices. It controls the waveforms and the parameters of the external address, data and control busses and is composed of the following elements:

- The Static Memory Controller (SMC)
- The SDRAM Controller (SDRAMC)
- A chip select assignment feature that assigns an HSB address space to the external devices
- A multiplex controller circuit that shares the pins between the different memory controllers

17.6.1 Bus Multiplexing

The EBI offers a complete set of control signals that share the 16(8)-bit data lines, the address lines of up to 24 bits and the control signals through a multiplex logic operating in function of the memory area requests.

Multiplexing is specifically organized in order to guarantee the maintenance of the address and output control lines at a stable state while no external access is being performed. Multiplexing is also designed to respect the data float times defined in the Memory Controllers. Furthermore, refresh cycles of the SDRAM are executed independently by the SDRAMC without delaying the other external memory controller accesses.

17.6.2 Static Memory Controller

For information on the Static Memory Controller, refer to the Static Memory Controller Section.

17.6.3 SDRAM Controller

Writing a one to the HMATRIX.SFR6.CS1A bit enables the SDRAM logic.

For information on the SDRAM Controller, refer to the SDRAM Section.

17.7 Application Example

17.7.1 Hardware Interface

Table 17-4. EBI Pins and External Static Devices Connections

Pins name	Pins of the Interfaced Device		
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device
Controller	**SMC**		
DATA[7:0]	D[7:0]	D[7:0]	D[7:0]
DATA[15:0]	–	D[15:8]	D[15:8]
ADDR[0]	A[0]	–	NBS0(2)
ADDR[1]	A[1]	A[0]	A[0]
ADDR[23:2]	A[23:2]	A[22:1]	A[22:1]
NCS[0] - NCS[3]	CS	CS	CS
NRD	OE	OE	OE
NWE0	WE	WE(1)	WE
NWE1	–	WE(1)	NBS1(2)

Note: 1. NWE1 enables upper byte writes. NWE0 enables lower byte writes.
2. NBS1 enables upper byte writes. NBS0 enables lower byte writes.

Table 17-5. EBI Pins and External Devices Connections

Pins name	Pins of the Interfaced Device
	SDRAM
Controller	**SDRAMC**
DATA[7:0]	D[7:0]
DATA[15:8]	D[15:8]
ADDR[0]	DQM0
ADDR[10:2]	A[8:0]
ADDR[11]	A[9]
SDA10	A[10]
ADDR[14:13]	A[12:11]
ADDR[16]	BA0
ADDR[17]	BA1
NCS[1]	CS
NWE1	DQM1
SDCK	CLK
SDCKE	CKE

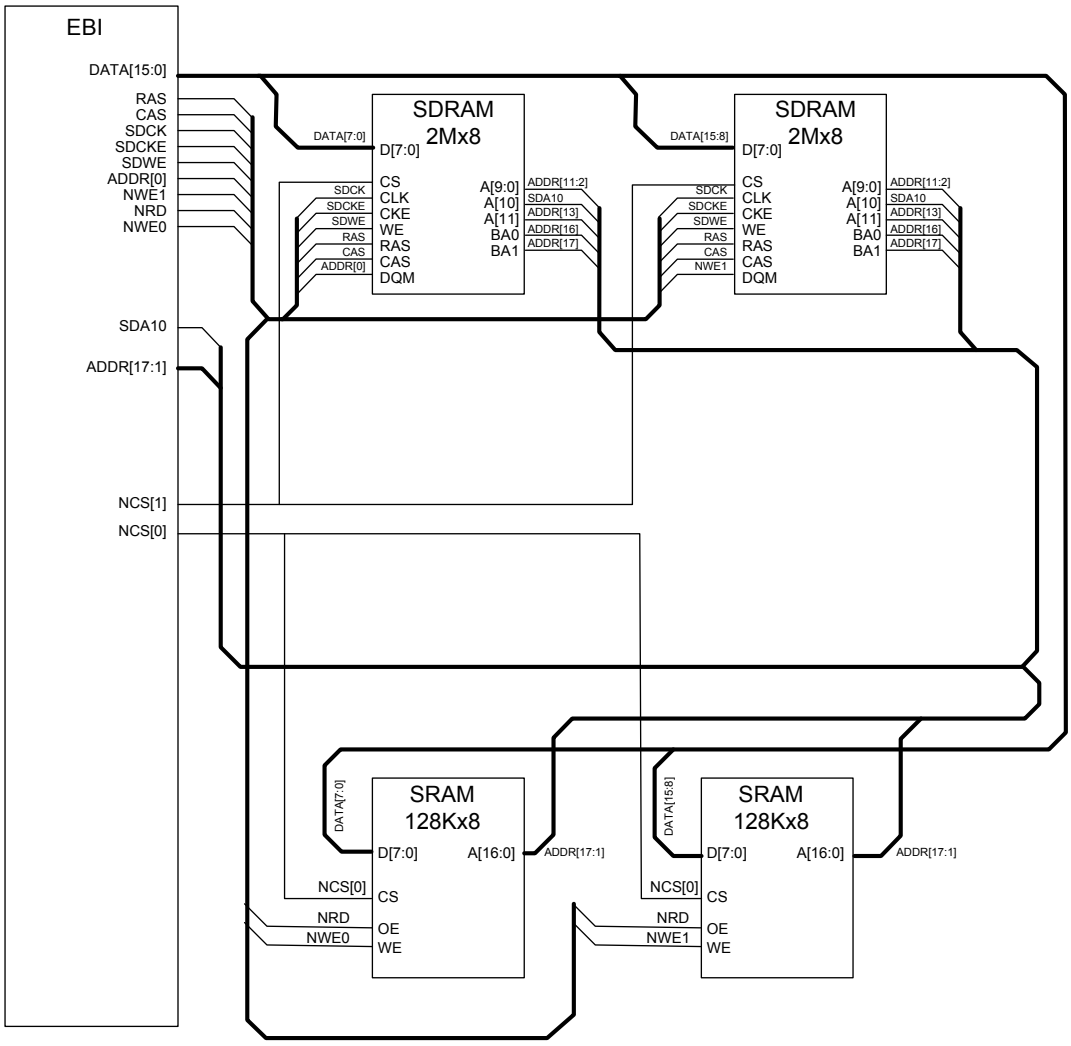
Table 17-5. EBI Pins and External Devices Connections (Continued)

	Pins of the Interfaced Device
Pins name	SDRAM
Controller	**SDRAMC**
RAS	RAS
CAS	CAS
SDWE	WE

17.7.2 Connection Examples

Figure 17-2 on page 306shows an example of connections between the EBI and external devices.

Figure 17-2. EBI Connections to Memory Devices



18. Static Memory Controller (SMC)

Rev. 1.0.6.5

18.1 Features

- **4 chip selects available**
- **16-Mbytes address space per chip select**
- **8- or 16-bit data bus**
- **Word, halfword, byte transfers**
- **Byte write or byte select lines**
- **Programmable setup, pulse and hold time for read signals per chip select**
- **Programmable setup, pulse and hold time for write signals per chip select**
- **Programmable data float time per chip select**
- **Compliant with LCD module**
- **External wait request**
- **Automatic switch to slow clock mode**
- **Asynchronous read in page mode supported: page size ranges from 4 to 32 bytes**

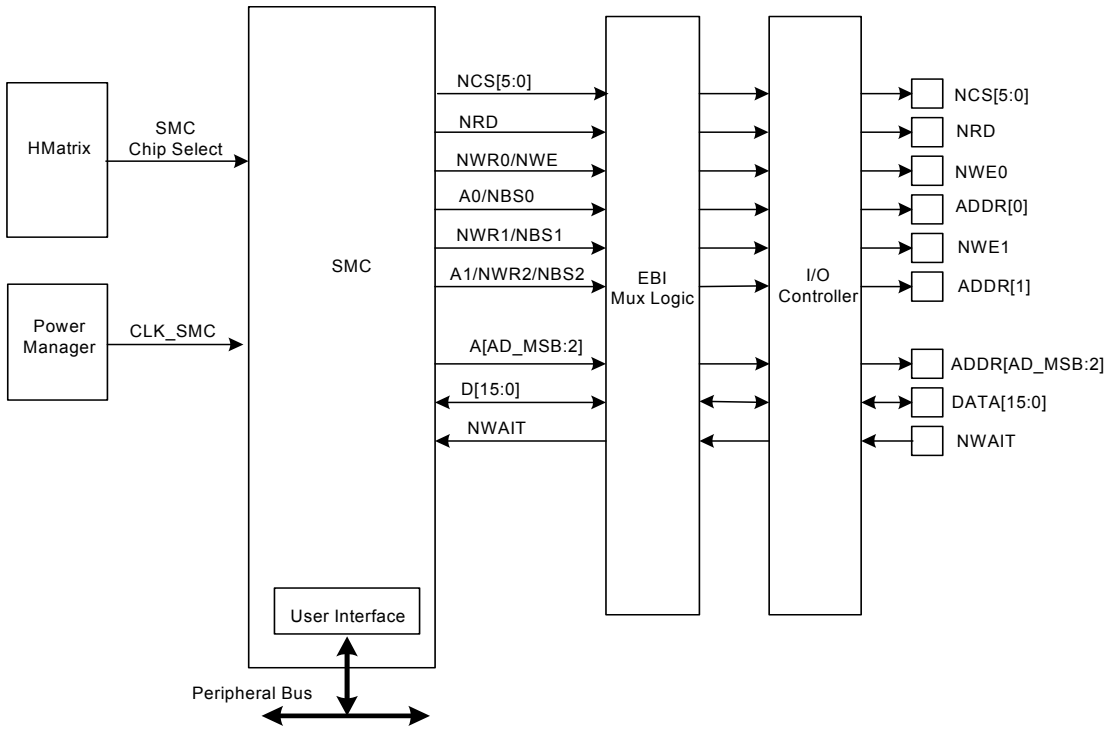
18.2 Overview

The Static Memory Controller (SMC) generates the signals that control the access to the external memory devices or peripheral devices. It has 4 chip selects and a 24-bit address bus. The 16-bit data bus can be configured to interface with 8-16-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully parametrizable.

The SMC can manage wait requests from external devices to extend the current access. The SMC is provided with an automatic slow clock mode. In slow clock mode, it switches from user-programmed waveforms to slow-rate specific waveforms on read and write signals. The SMC supports asynchronous burst read in page mode access for page size up to 32 bytes.

18.3 Block Diagram

Figure 18-1. SMC Block Diagram (AD_MSB=23)



18.4 I/O Lines Description

Table 18-1. I/O Lines Description

Pin Name	Pin Description	Type	Active Level
NCS[3:0]	Chip Select Lines	Output	Low
NRD	Read Signal	Output	Low
NWR0/NWE	Write 0/Write Enable Signal	Output	Low
A0/NBS0	Address Bit 0/Byte 0 Select Signal	Output	Low
NWR1/NBS1	Write 1/Byte 1 Select Signal	Output	Low
A[23:2]	Address Bus	Output	
D[15:0]	Data Bus	Input/Output	
NWAIT	External Wait Signal	Input	Low

18.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

18.5.1 I/O Lines

The SMC signals pass through the External Bus Interface (EBI) module where they are multiplexed. The user must first configure the I/O Controller to assign the EBI pins corresponding to SMC signals to their peripheral function. If the I/O lines of the EBI corresponding to SMC signals are not used by the application, they can be used for other purposes by the I/O Controller.

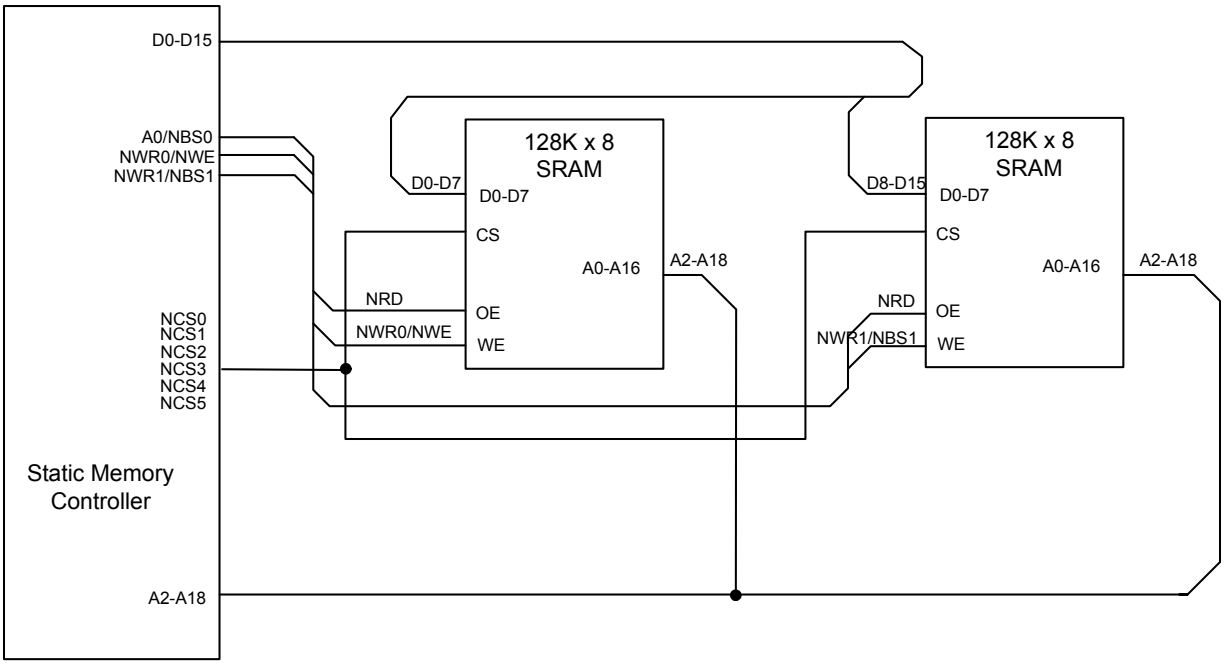
18.5.2 Clocks

The clock for the SMC bus interface (CLK_SMC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the SMC before disabling the clock, to avoid freezing the SMC in an undefined state.

18.6 Functional Description

18.6.1 Application Example

Figure 18-2. SMC Connections to Static Memory Devices



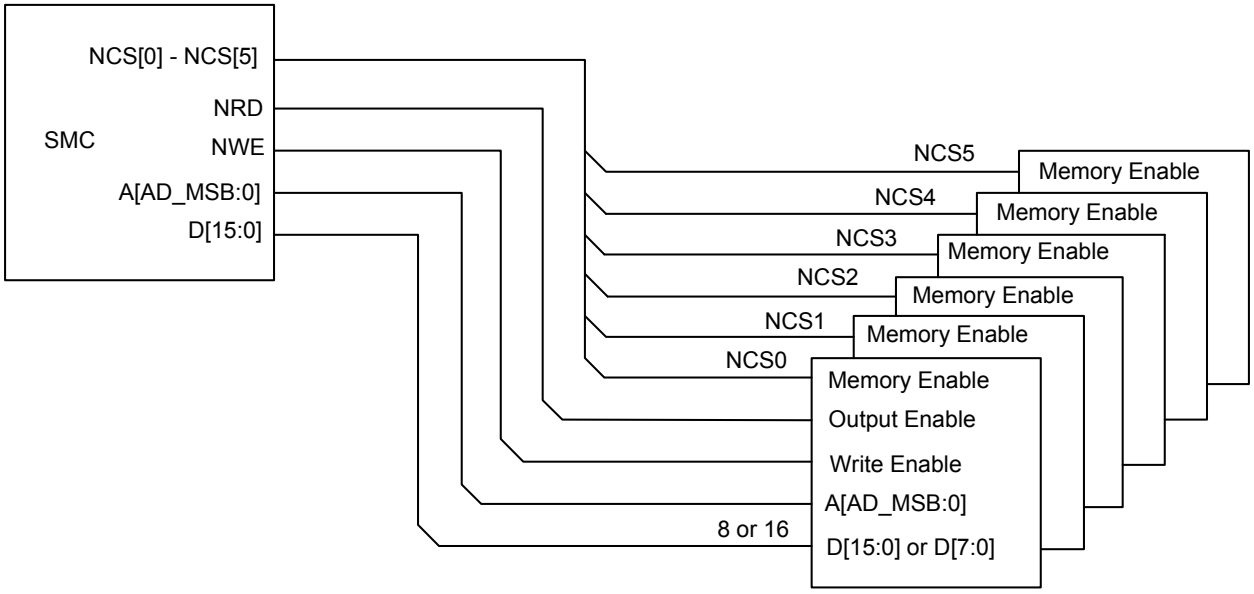
18.6.2 External Memory Mapping

The SMC provides up to 24 address lines, A[23:0]. This allows each chip select line to address up to 16 Mbytes of memory.

If the physical memory device connected on one chip select is smaller than 16 Mbytes, it wraps around and appears to be repeated within this space. The SMC correctly handles any valid access to the memory device within the page (see Figure 18-3 on page 310).

A[23:0] is only significant for 8-bit memory, A[23:1] is used for 16-bit memory23.

Figure 18-3. Memory Connections for Six External Devices



18.6.3 Connection to External Devices

18.6.3.1 Data bus width

A data bus width of 8 or 16 bits can be selected for each chip select. This option is controlled by the Data Bus Width field in the Mode Register (MODE.DBW) for the corresponding chip select.

Figure 18-4 on page 310 shows how to connect a 512K x 8-bit memory on NCS2. Figure 18-5 on page 311 shows how to connect a 512K x 16-bit memory on NCS2.

18.6.3.2 Byte write or byte select access

Each chip select with a 16-bit data bus can operate with one of two different types of write access: byte write or byte select access. This is controlled by the Byte Access Type bit in the MODE register (MODE.BAT) for the corresponding chip select.

Figure 18-4. Memory Connection for an 8-bit Data Bus

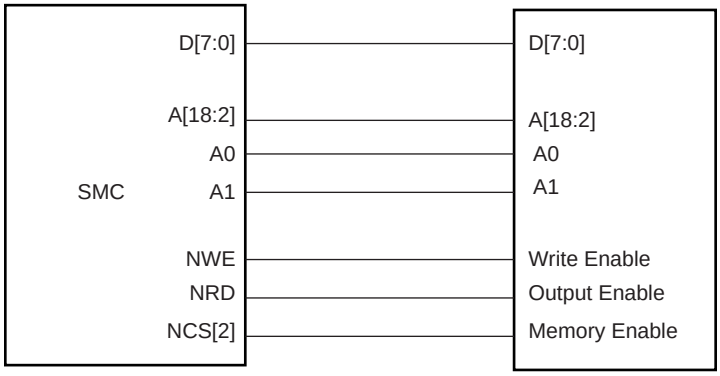
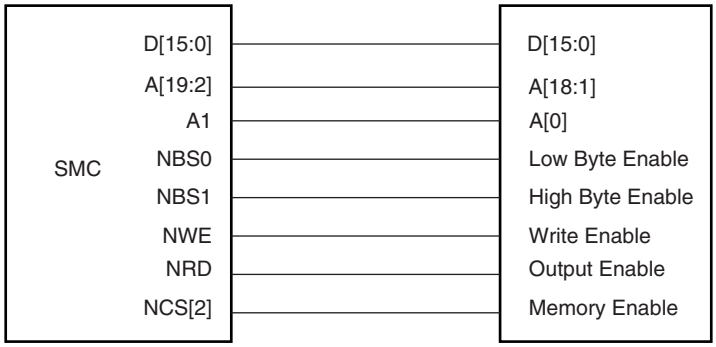


Figure 18-5. Memory Connection for a 16-bit Data Bus



•*Byte write access*

The byte write access mode supports one byte write signal per byte of the data bus and a single read signal.

Note that the SMC does not allow boot in byte write access mode.

- For 16-bit devices: the SMC provides NWR0 and NWR1 write signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus. One single read signal (NRD) is provided.

The byte write access mode is used to connect two 8-bit devices as a 16-bit memory.

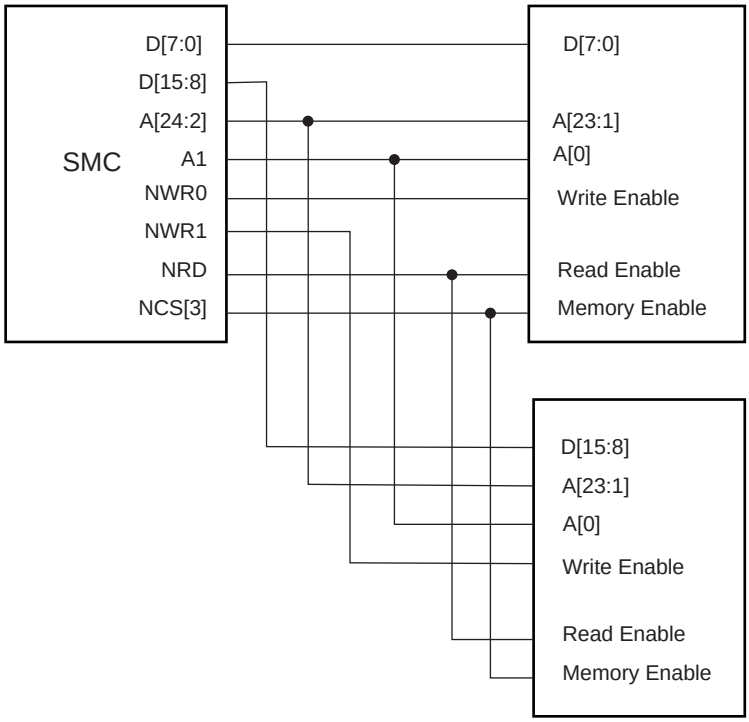
The byte write option is illustrated on Figure 18-6 on page 312.

•*Byte select access*

In this mode, read/write operations can be enabled/disabled at a byte level. One byte select line per byte of the data bus is provided. One NRD and one NWE signal control read and write.

- For 16-bit devices: the SMC provides NBS0 and NBS1 selection signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus. The byte select access is used to connect one 16-bit device.

Figure 18-6. Connection of two 8-bit Devices on a 16-bit Bus: Byte Write Option



•*Signal multiplexing*

Depending on the MODE.BAT bit, only the write signals or the byte select signals are used. To save I/Os at the external bus interface, control signals at the SMC interface are multiplexed.

For 16-bit devices, bit A0 of address is unused. When byte select option is selected, NWR1 is unused. When byte write option is selected, NBS0 to NBS1 are unused.

Table 18-3. SMC Multiplexed Signal Translation

Signal Name	**16-bit Bus**		**8-bit Bus**
Device Type	**1 x 16-bit**	**2 x 8-bit**	**1 x 8-bit**
Byte Access Type (BAT)	Byte Select	Byte Write	
NBS0_A0	NBS0		A0
NWE_NWR0	NWE	NWR0	NWE
NBS1_NWR1	NBS1	NWR1	
NBS2_NWR2_A1	A1	A1	A1

18.6.4 Standard Read and Write Protocols

In the following sections, the byte access type is not considered. Byte select lines (NBS0 to NBS1) always have the same timing as the address bus (A). NWE represents either the NWE signal in byte select access type or one of the byte write lines (NWR0 to NWR1) in byte write

access type. NWR0 to NWR1 have the same timings and protocol as NWE. In the same way, NCS represents one of the NCS[0..3] chip select lines.

18.6.4.1 *Read waveforms*

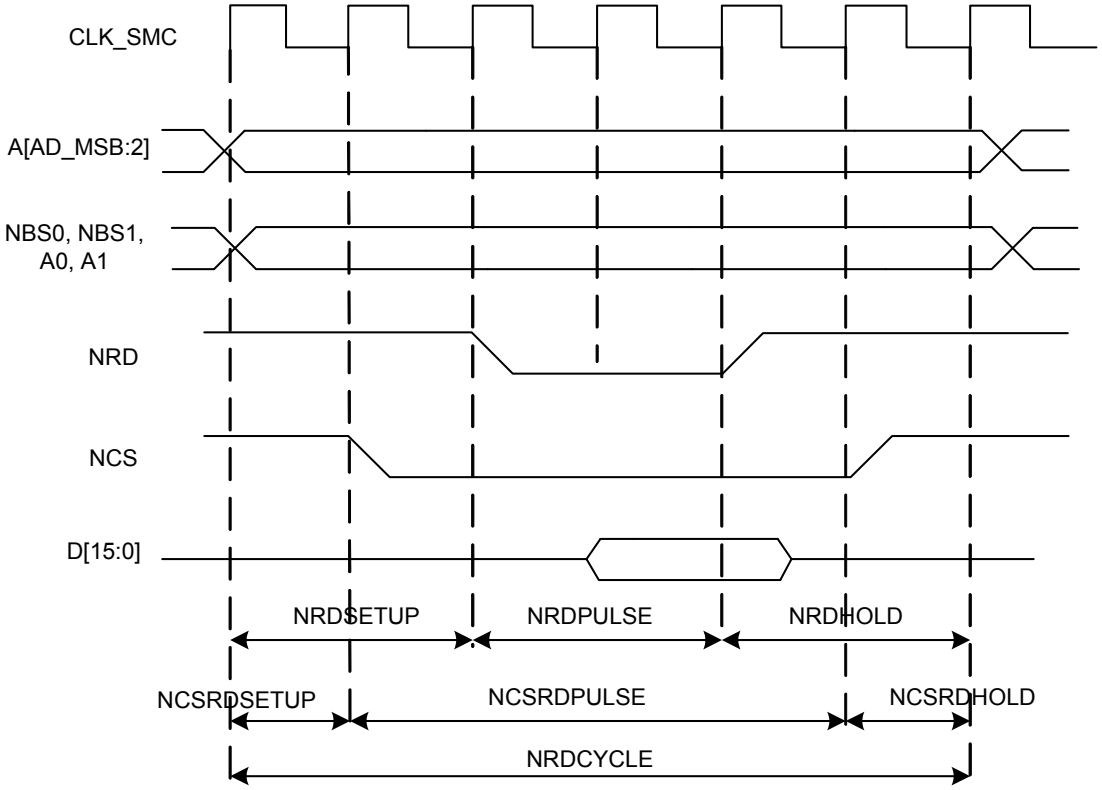
The read cycle is shown on Figure 18-7 on page 313.

The read cycle starts with the address setting on the memory address bus, i.e.:

{A[23:2], A1, A0} for 8-bit devices

{A[23:2], A1} for 16-bit devices

Figure 18-7. Standard Read Cycle



•*NRD waveform*

The NRD signal is characterized by a setup timing, a pulse width, and a hold timing.

1. NRDSETUP: the NRD setup time is defined as the setup of address before the NRD falling edge.
2. NRDPULSE: the NRD pulse length is the time between NRD falling edge and NRD rising edge.
3. NRDHOLD: the NRD hold time is defined as the hold time of address after the NRD rising edge.

•*NCS waveform*

Similarly, the NCS signal can be divided into a setup time, pulse length and hold time.

1. NCSRDSETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCSRDPULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge.
3. NCSRDHOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

•Read cycle

The NRDCYCLE time is defined as the total duration of the read cycle, i.e., from the time where address is set on the address bus to the point where address may change. The total read cycle time is equal to:

$$NRDCYCLE = NRDSETUP + NRDPULSE + NRDHOLD$$

Similarly,

$$NRDCYCLE = NCSRDSETUP + NCSRDPULSE + NCSRDHOLD$$

All NRD and NCS timings are defined separately for each chip select as an integer number of CLK_SMC cycles. To ensure that the NRD and NCS timings are coherent, the user must define the total read cycle instead of the hold timing. NRDCYCLE implicitly defines the NRD hold time and NCS hold time as:

$$NRDHOLD = NRDCYCLE - NRDSETUP - NRDPULSE$$

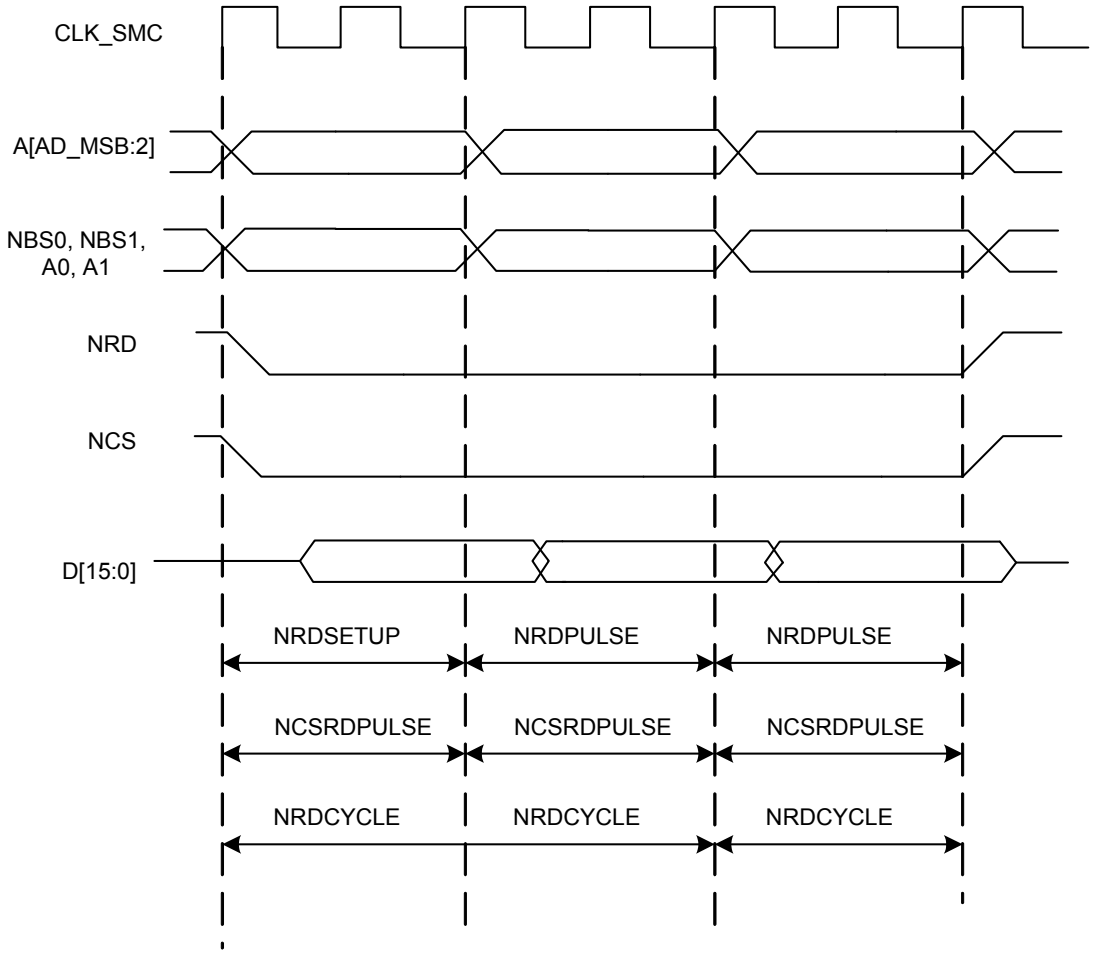
And,

$$NCSRDHOLD = NRDCYCLE - NCSRDSETUP - NCSRDPULSE$$

•Null delay setup and hold

If null setup and hold parameters are programmed for NRD and/or NCS, NRD and NCS remain active continuously in case of consecutive read cycles in the same memory (see Figure 18-8 on page 315).

Figure 18-8. No Setup, No Hold on NRD, and NCS Read Signals



- Null Pulse

Programming null pulse is not permitted. Pulse must be at least written to one. A null value leads to unpredictable behavior.

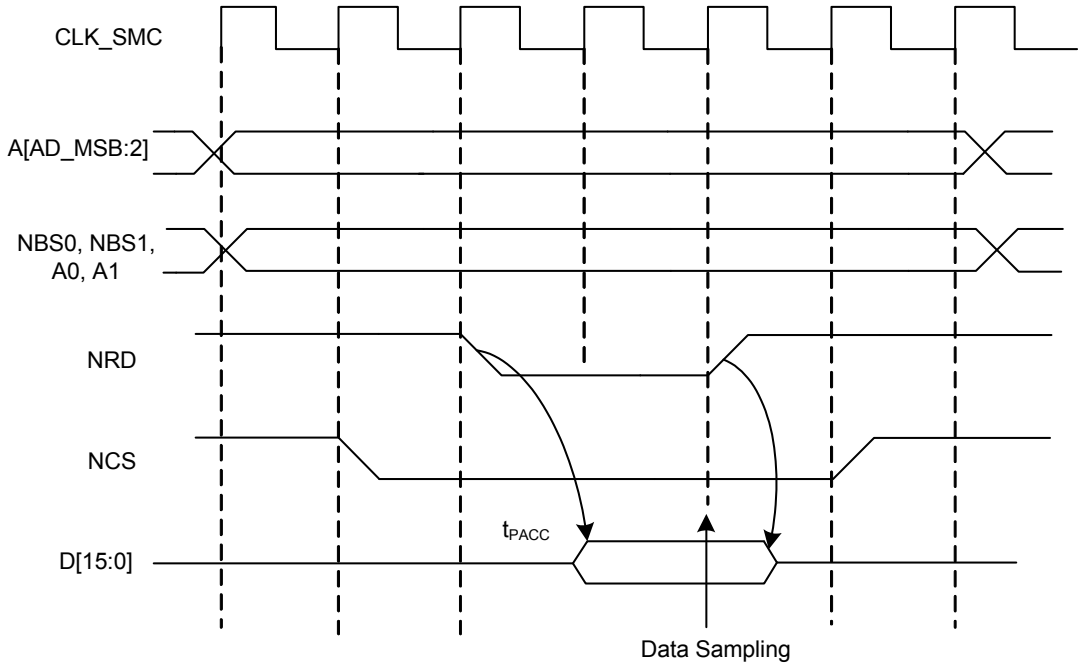
18.6.4.2 *Read mode*

As NCS and NRD waveforms are defined independently of one other, the SMC needs to know when the read data is available on the data bus. The SMC does not compare NCS and NRD timings to know which signal rises first. The Read Mode bit in the MODE register (MODE.READMODE) of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

•Read is controlled by NRD (MODE.READMODE = 1)

Figure 18-9 on page 316 shows the waveforms of a read operation of a typical asynchronous RAM. The read data is available t_{PACC} after the falling edge of NRD, and turns to 'Z' after the rising edge of NRD. In this case, the MODE.READMODE bit must be written to one (read is controlled by NRD), to indicate that data is available with the rising edge of NRD. The SMC samples the read data internally on the rising edge of CLK_SMC that generates the rising edge of NRD, whatever the programmed waveform of NCS may be.

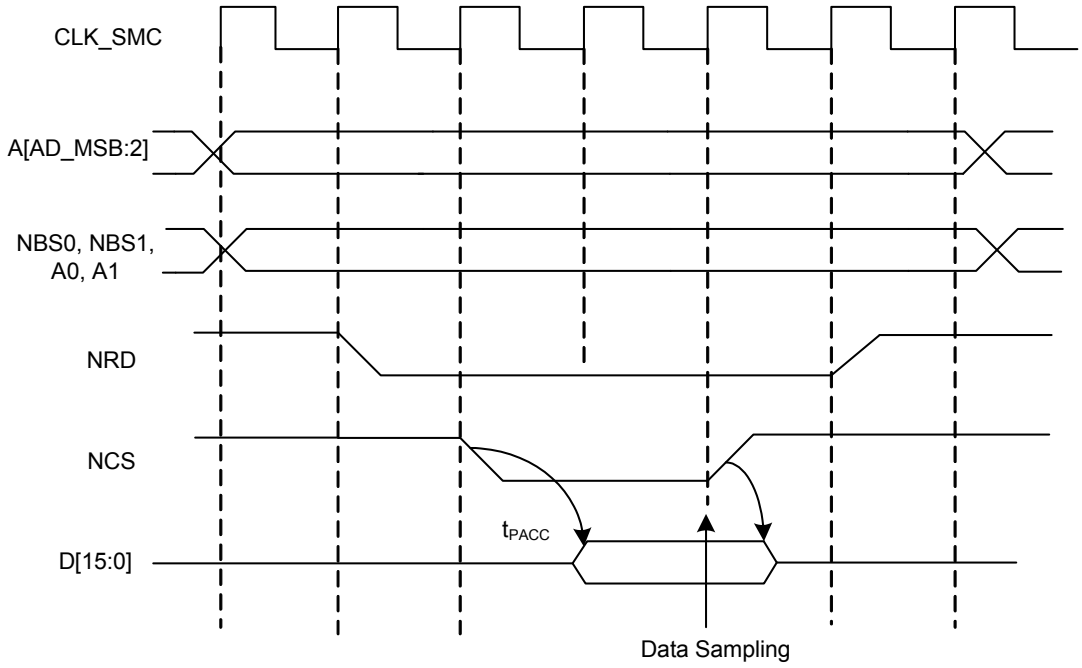
Figure 18-9. READMODE = 1: Data Is Sampled by SMC Before the Rising Edge of NRD



•*Read is controlled by NCS (MODE.READMODE = 0)*

Figure 18-10 on page 317 shows the typical read cycle of an LCD module. The read data is valid t_{PACC} after the falling edge of the NCS signal and remains valid until the rising edge of NCS. Data must be sampled when NCS is raised. In that case, the MODE.READMODE bit must be written to zero (read is controlled by NCS): the SMC internally samples the data on the rising edge of CML_SMC that generates the rising edge of NCS, whatever the programmed waveform of NRD may be.

Figure 18-10. READMODE = 0: Data Is Sampled by SMC Before the Rising Edge of NCS



18.6.4.3 *Write waveforms*

The write protocol is similar to the read protocol. It is depicted in Figure 18-11 on page 318. The write cycle starts with the address setting on the memory address bus.

•*NWE waveforms*

The NWE signal is characterized by a setup timing, a pulse width and a hold timing.

1. NWESETUP: the NWE setup time is defined as the setup of address and data before the NWE falling edge.
2. NWEPULSE: the NWE pulse length is the time between NWE falling edge and NWE rising edge.
3. NWEHOLD: the NWE hold time is defined as the hold time of address and data after the NWE rising edge.

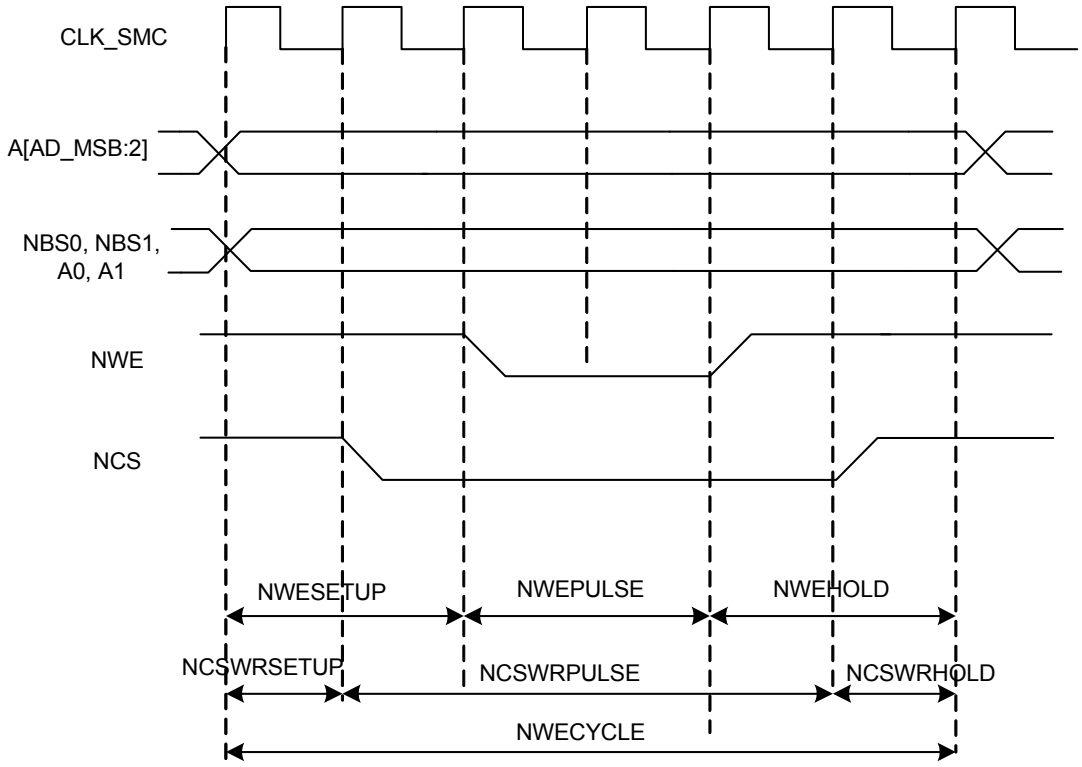
The NWE waveforms apply to all byte-write lines in byte write access mode: NWR0 to NWR3.

18.6.4.4 *NCS waveforms*

The NCS signal waveforms in write operation are not the same that those applied in read operations, but are separately defined.

1. NCSWRSETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCSWRPULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge;
3. NCSWRHOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

Figure 18-11. Write Cycle



•*Write cycle*

The write cycle time is defined as the total duration of the write cycle, that is, from the time where address is set on the address bus to the point where address may change. The total write cycle time is equal to:

$$NWECYCLE = NWESETUP + NWEPULSE + NWEHOLD$$

Similarly,

$$NWECYCLE = NCSWRSETUP + NCSWRPULSE + NCSWRHOLD$$

All NWE and NCS (write) timings are defined separately for each chip select as an integer number of CLK_SMC cycles. To ensure that the NWE and NCS timings are coherent, the user must define the total write cycle instead of the hold timing. This implicitly defines the NWE hold time and NCS (write) hold times as:

$$NWEHOLD = NWECYCLE - NWESETUP - NWEPULSE$$

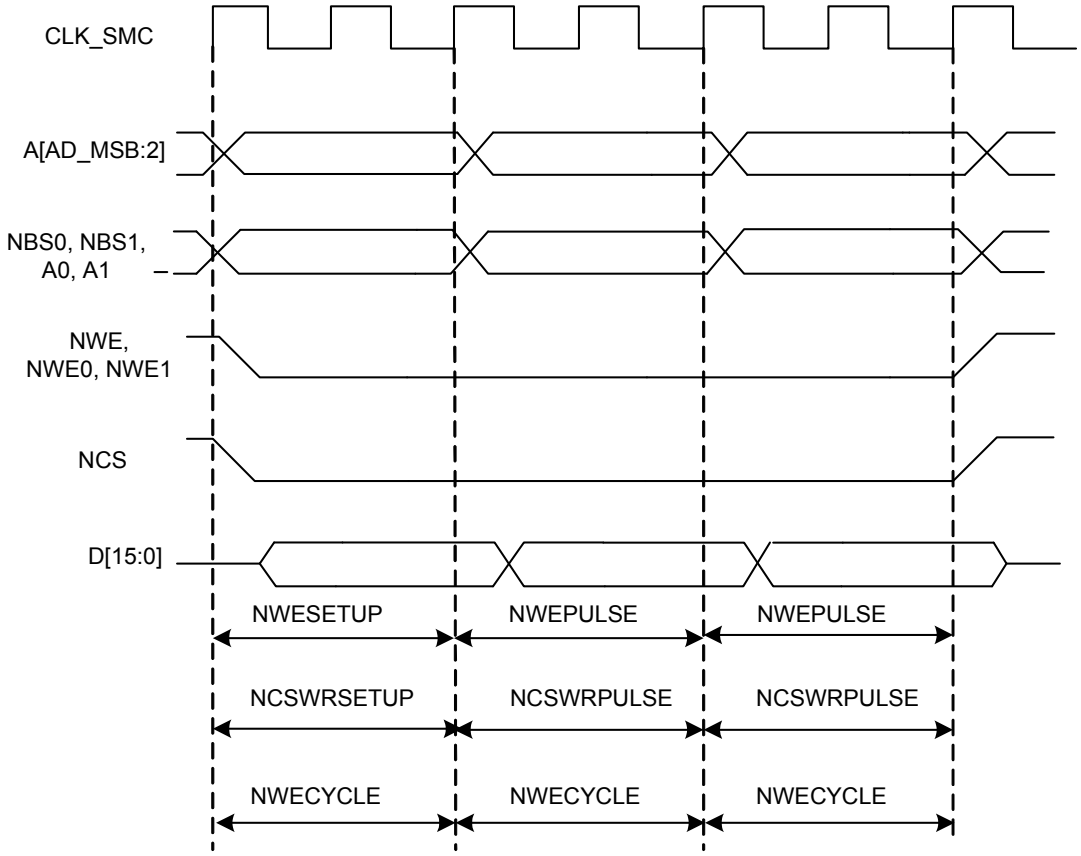
And,

$$NCSWRHOLD = NWECYCLE - NCSWRSETUP - NCSWRPULSE$$

•*Null delay setup and hold*

If null setup parameters are programmed for NWE and/or NCS, NWE and/or NCS remain active continuously in case of consecutive write cycles in the same memory (see Figure 18-12 on page 319). However, for devices that perform write operations on the rising edge of NWE or NCS, such as SRAM, either a setup or a hold must be programmed.

Figure 18-12. Null Setup and Hold Values of NCS and NWE in Write Cycle



•*Null pulse*

Programming null pulse is not permitted. Pulse must be at least written to one. A null value leads to unpredictable behavior.

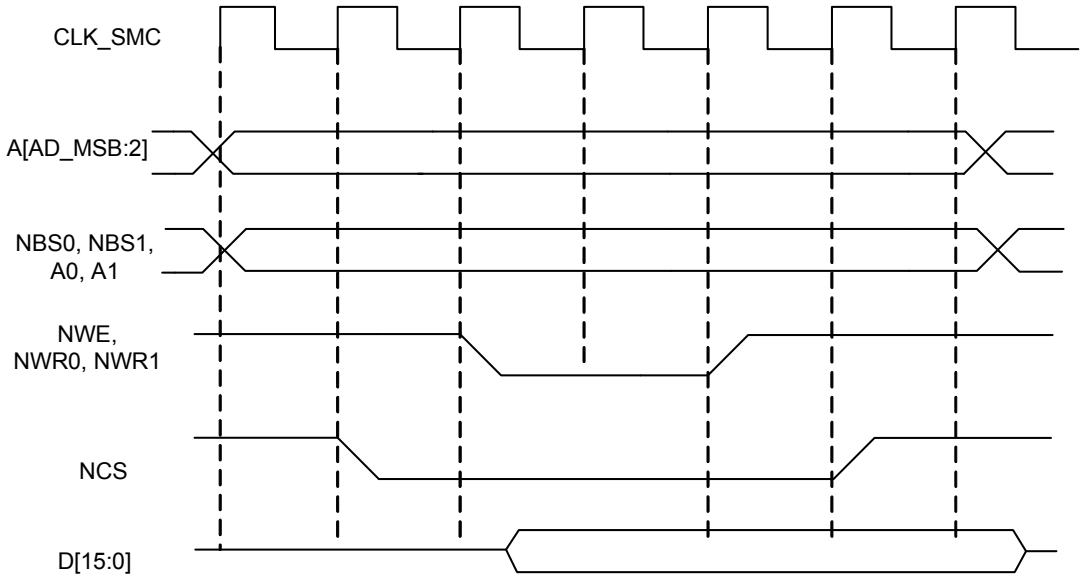
18.6.4.5 *Write mode*

The Write Mode bit in the MODE register (MODE.WRITEMODE) of the corresponding chip select indicates which signal controls the write operation.

•*Write is controlled by NWE (MODE.WRITEMODE = 1)*

Figure 18-13 on page 320 shows the waveforms of a write operation with MODE.WRITEMODE equal to one. The data is put on the bus during the pulse and hold steps of the NWE signal. The internal data buffers are turned out after the NWESETUP time, and until the end of the write cycle, regardless of the programmed waveform on NCS.

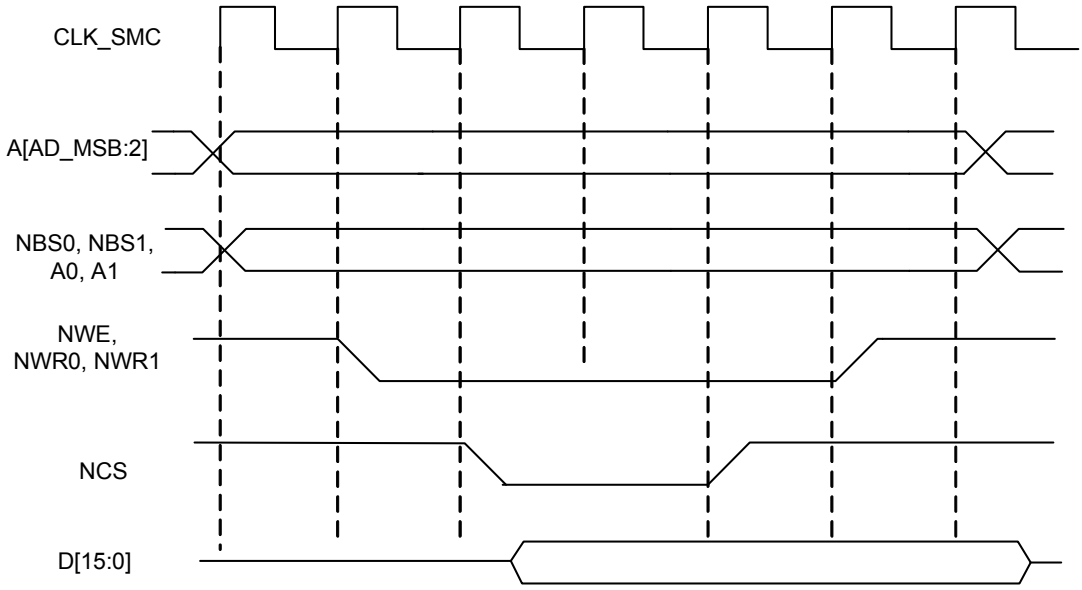
Figure 18-13. WRITEMODE = 1. The Write Operation Is Controlled by NWE



•*Write is controlled by NCS (MODE.WRITEMODE = 0)*

Figure 18-14 on page 320 shows the waveforms of a write operation with MODE.WRITEMODE written to zero. The data is put on the bus during the pulse and hold steps of the NCS signal. The internal data buffers are turned out after the NCSWRSETUP time, and until the end of the write cycle, regardless of the programmed waveform on NWE.

Figure 18-14. WRITEMODE = 0. The Write Operation Is Controlled by NCS



18.6.4.6 Coding timing parameters

All timing parameters are defined for one chip select and are grouped together in one register according to their type.

The Setup register (SETUP) groups the definition of all setup parameters:

- NRDSETUP, NCSRDSETUP, NWESETUP, and NCSWRSETUP.

The Pulse register (PULSE) groups the definition of all pulse parameters:

- NRDPULSE, NCSRDPULSE, NWEPULSE, and NCSWRPULSE.

The Cycle register (CYCLE) groups the definition of all cycle parameters:

- NRDCYCLE, NWECYCLE.

Table 18-4 on page 321 shows how the timing parameters are coded and their permitted range.

Table 18-4. Coding and Range of Timing Parameters

Coded Value	Number of Bits	Effective Value	Permitted Range	
			Coded Value	Effective Value
setup [5:0]	6	128 x setup[5] + setup[4:0]	0 ≤ value ≤ 31 32 ≤ value ≤ 63	0 ≤ value ≤ 31 128 ≤ value ≤ 128+31
pulse [6:0]	7	256 x pulse[6] + pulse[5:0]	0 ≤ value ≤ 63 64≤ value ≤ 127	0 ≤ value ≤ 63 256 ≤ value ≤ 256+63
cycle [8:0]	9	256 x cycle[8:7] + cycle[6:0]	0 ≤ value ≤ 127 128 ≤ value ≤ 255 256 ≤ value ≤ 383 384 ≤ value ≤ 511	0 ≤ value ≤ 127 256 ≤ value ≤ 256+127 512 ≤ value ≤ 512+127 768 ≤ value ≤ 768+127

18.6.4.7 Usage restriction

The SMC does not check the validity of the user-programmed parameters. If the sum of SETUP and PULSE parameters is larger than the corresponding CYCLE parameter, this leads to unpredictable behavior of the SMC.

For read operations:

Null but positive setup and hold of address and NRD and/or NCS can not be guaranteed at the memory interface because of the propagation delay of theses signals through external logic and pads. If positive setup and hold values must be verified, then it is strictly recommended to program non-null values so as to cover possible skews between address, NCS and NRD signals.

For write operations:

If a null hold value is programmed on NWE, the SMC can guarantee a positive hold of address, byte select lines, and NCS signal after the rising edge of NWE. This is true if the MODE.WRITEMODE bit is written to one. See Section 18.6.5.2.

For read and write operations: a null value for pulse parameters is forbidden and may lead to unpredictable behavior.

In read and write cycles, the setup and hold time parameters are defined in reference to the address bus. For external devices that require setup and hold time between NCS and NRD signals (read), or between NCS and NWE signals (write), these setup and hold times must be converted into setup and hold times in reference to the address bus.

18.6.5 Automatic Wait States

Under certain circumstances, the SMC automatically inserts idle cycles between accesses to avoid bus contention or operation conflict.

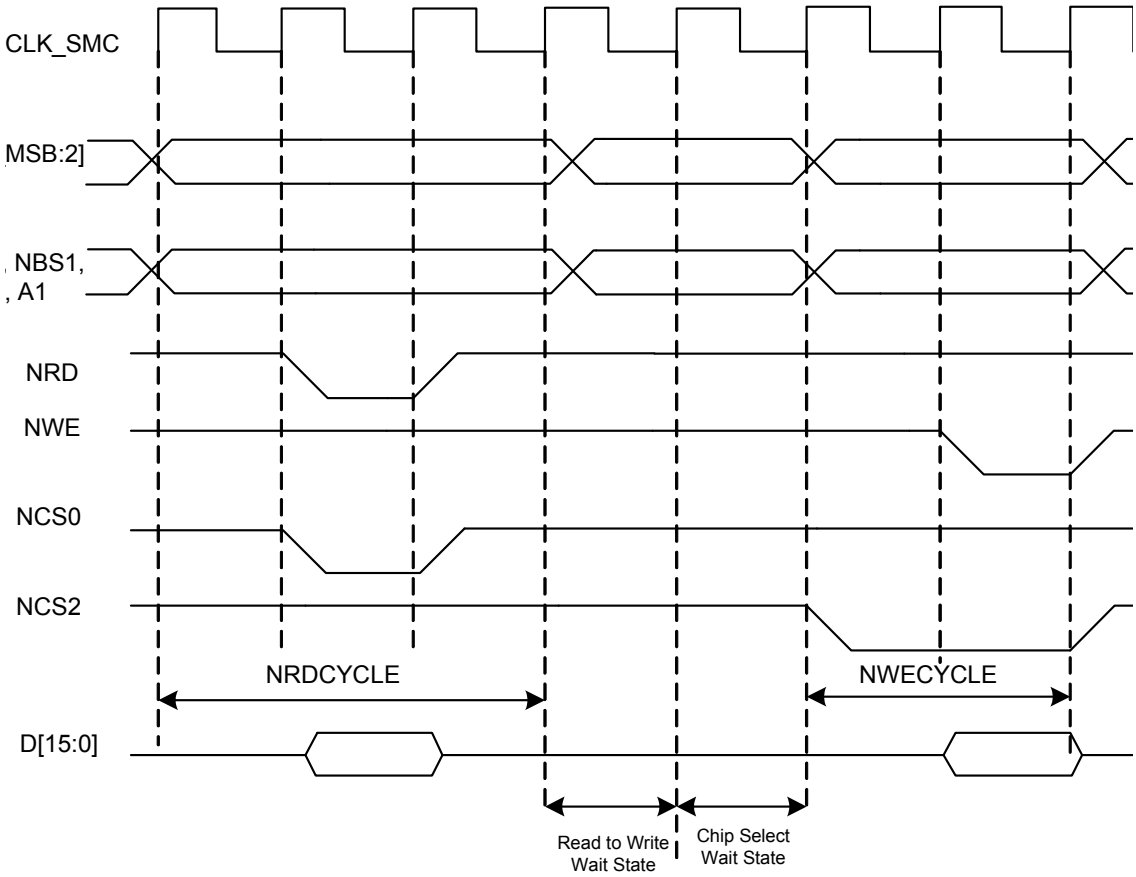
18.6.5.1 Chip select wait states

The SMC always inserts an idle cycle between two transfers on separate chip selects. This idle cycle ensures that there is no bus contention between the deactivation of one device and the activation of the next one.

During chip select wait state, all control lines are turned inactive: NBS0 to NBS3, NWR0 to NWR3, NCS[0..5], NRD lines are all set to high level.

Figure 18-15 on page 322 illustrates a chip select wait state between access on Chip Select 0 (NCS0) and Chip Select 2 (NCS2).

Figure 18-15. Chip Select Wait State Between a Read Access on NCS0 and a Write Access on NCS2



18.6.5.2 Early read wait state

In some cases, the SMC inserts a wait state cycle between a write access and a read access to allow time for the write cycle to end before the subsequent read cycle begins. This wait state is not generated in addition to a chip select wait state. The early read cycle thus only occurs between a write and read access to the same memory device (same chip select).

An early read wait state is automatically inserted if at least one of the following conditions is valid:

- if the write controlling signal has no hold time and the read controlling signal has no setup time (Figure 18-16 on page 323).
- in NCS write controlled mode (MODE.WRITEMODE = 0), if there is no hold timing on the NCS signal and the NCSRDSETUP parameter is set to zero, regardless of the read mode (Figure 18-17 on page 324). The write operation must end with a NCS rising edge. Without an early read wait state, the write operation could not complete properly.
- in NWE controlled mode (MODE.WRITEMODE = 1) and if there is no hold timing (NWEHOLD = 0), the feedback of the write control signal is used to control address, data, chip select, and byte select lines. If the external write control signal is not inactivated as expected due to load capacitances, an early read wait state is inserted and address, data and control signals are maintained one more cycle. See Figure 18-18 on page 325.

Figure 18-16. Early Read Wait State: Write with No Hold Followed by Read with No Setup.

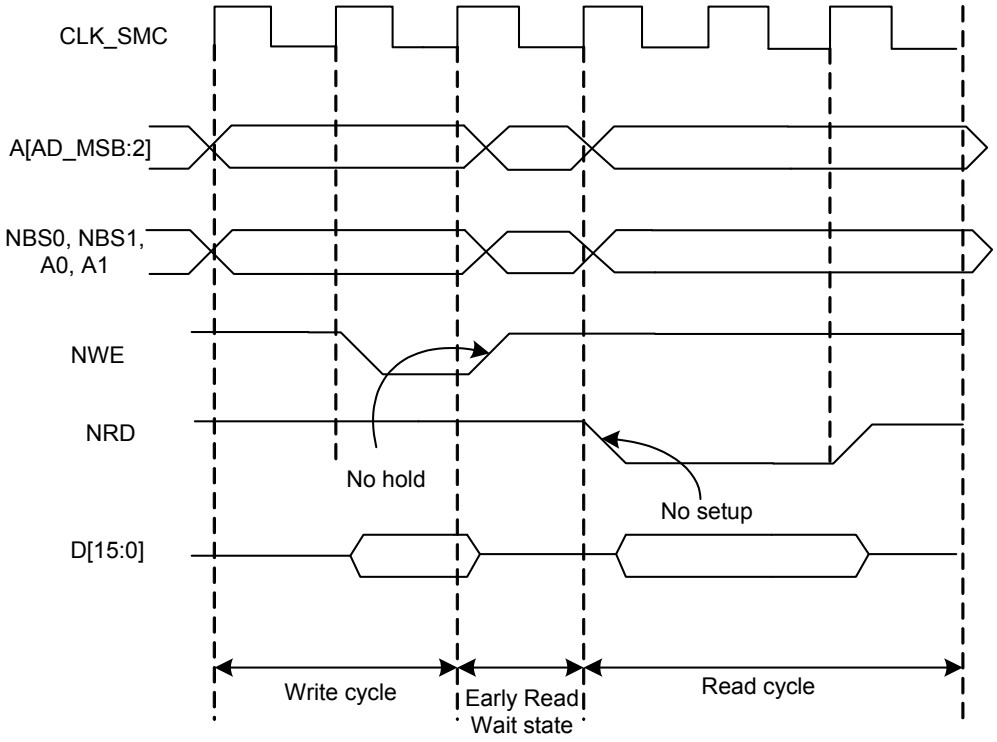


Figure 18-17. Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No Setup.

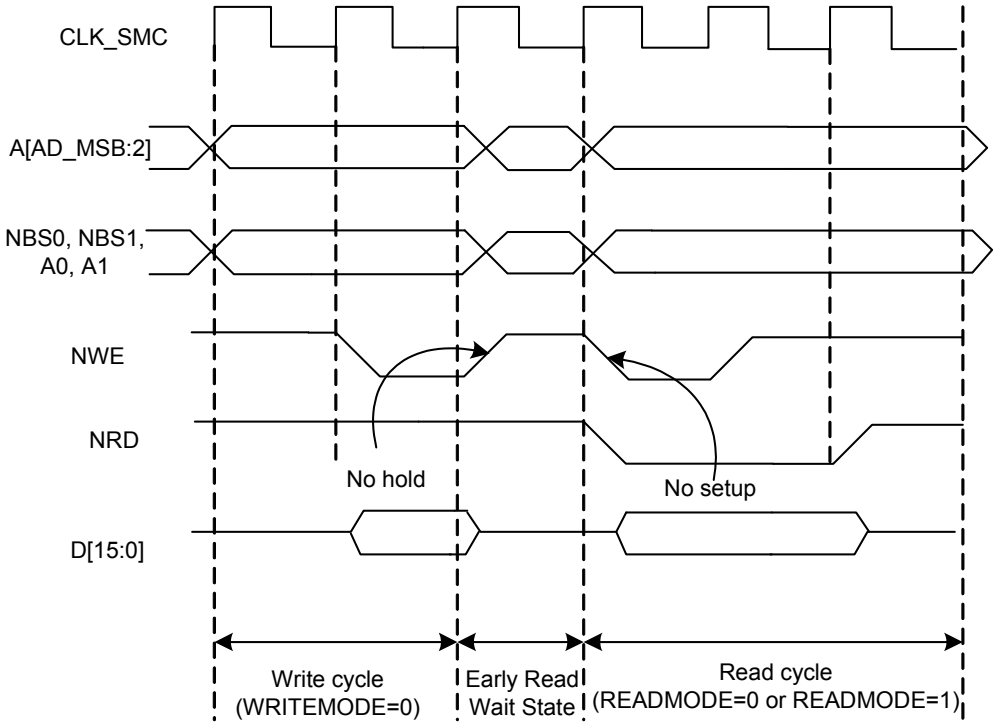
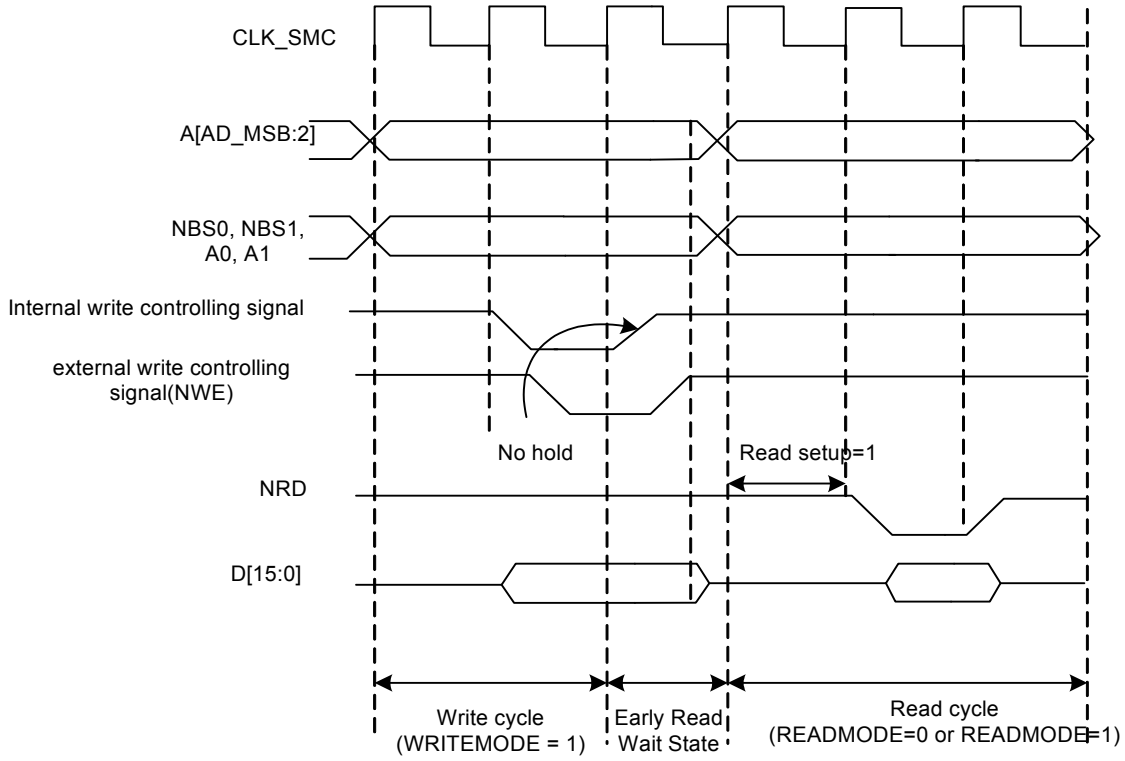


Figure 18-18. Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle.



18.6.5.3 *Reload user configuration wait state*

The user may change any of the configuration parameters by writing the SMC user interface.

When detecting that a new user configuration has been written in the user interface, the SMC inserts a wait state before starting the next access. The so called “reload user configuration wait state” is used by the SMC to load the new set of parameters to apply to next accesses.

The reload configuration wait state is not applied in addition to the chip select wait state. If accesses before and after reprogramming the user interface are made to different devices (different chip selects), then one single chip select wait state is applied.

On the other hand, if accesses before and after writing the user interface are made to the same device, a reload configuration wait state is inserted, even if the change does not concern the current chip select.

- *User procedure*

To insert a reload configuration wait state, the SMC detects a write access to any MODE register of the user interface. If the user only modifies timing registers (SETUP, PULSE, CYCLE registers) in the user interface, he must validate the modification by writing the MODE register, even if no change was made on the mode parameters.

•*Slow clock mode transition*

A reload configuration wait state is also inserted when the slow clock mode is entered or exited, after the end of the current transfer (see Section 18.6.8).

18.6.5.4 Read to write wait state

Due to an internal mechanism, a wait cycle is always inserted between consecutive read and write SMC accesses.

This wait cycle is referred to as a read to write wait state in this document.

This wait cycle is applied in addition to chip select and reload user configuration wait states when they are to be inserted. See Figure 18-15 on page 322.

18.6.6 Data Float Wait States

Some memory devices are slow to release the external bus. For such devices, it is necessary to add wait states (data float wait states) after a read access:

- before starting a read access to a different external memory.
- before starting a write access to the same device or to a different external one.

The Data Float Output Time (t_{DF}) for each external memory device is programmed in the Data Float Time field of the MODE register (MODE.TDFCYCLES) for the corresponding chip select. The value of MODE.TDFCYCLES indicates the number of data float wait cycles (between 0 and 15) before the external device releases the bus, and represents the time allowed for the data output to go to high impedance after the memory is disabled.

Data float wait states do not delay internal memory accesses. Hence, a single access to an external memory with long t_{DF} will not slow down the execution of a program from internal memory.

The data float wait states management depends on the MODE.READMODE bit and the TDF Optimization bit of the MODE register (MODE.TDFMODE) for the corresponding chip select.

18.6.6.1 Read mode

Writing a one to the MODE.READMODE bit indicates to the SMC that the NRD signal is responsible for turning off the tri-state buffers of the external memory device. The data float period then begins after the rising edge of the NRD signal and lasts MODE.TDFCYCLES cycles of the CLK_SMC clock.

When the read operation is controlled by the NCS signal (MODE.READMODE = 0), the MODE.TDFCYCLES field gives the number of CLK_SMC cycles during which the data bus remains busy after the rising edge of NCS.

Figure 18-19 on page 327 illustrates the data float period in NRD-controlled mode (MODE.READMODE =1), assuming a data float period of two cycles (MODE.TDFCYCLES = 2). Figure 18-20 on page 327 shows the read operation when controlled by NCS (MODE.READMODE = 0) and the MODE.TDFCYCLES field equals to three.

Figure 18-19. TDF Period in NRD Controlled Read Access (TDFCYCLES = 2)

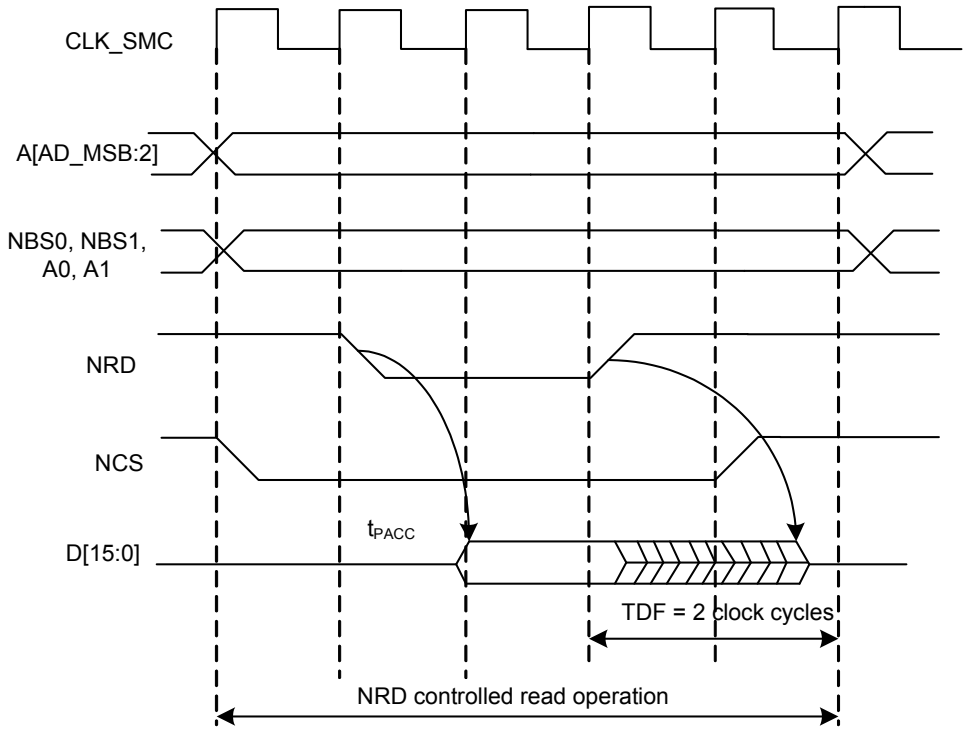
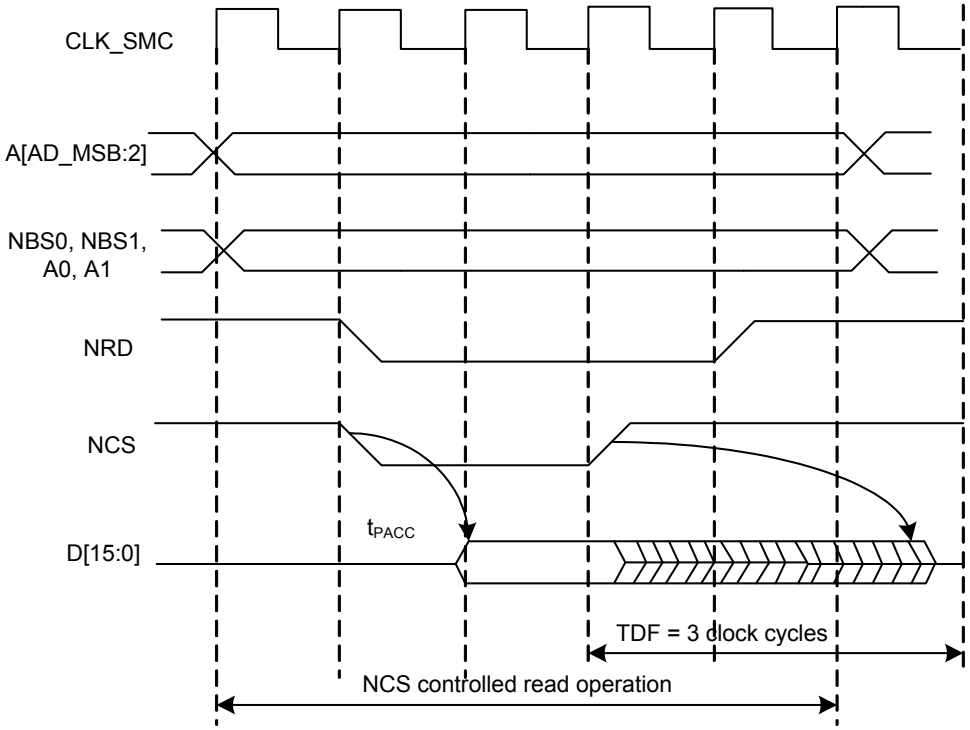


Figure 18-20. TDF Period in NCS Controlled Read Operation (TDFCYCLES = 3)



18.6.6.2 *TDF optimization enabled (MODE.TDFMODE = 1)*

When the MODE.TDFMODE bit is written to one (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

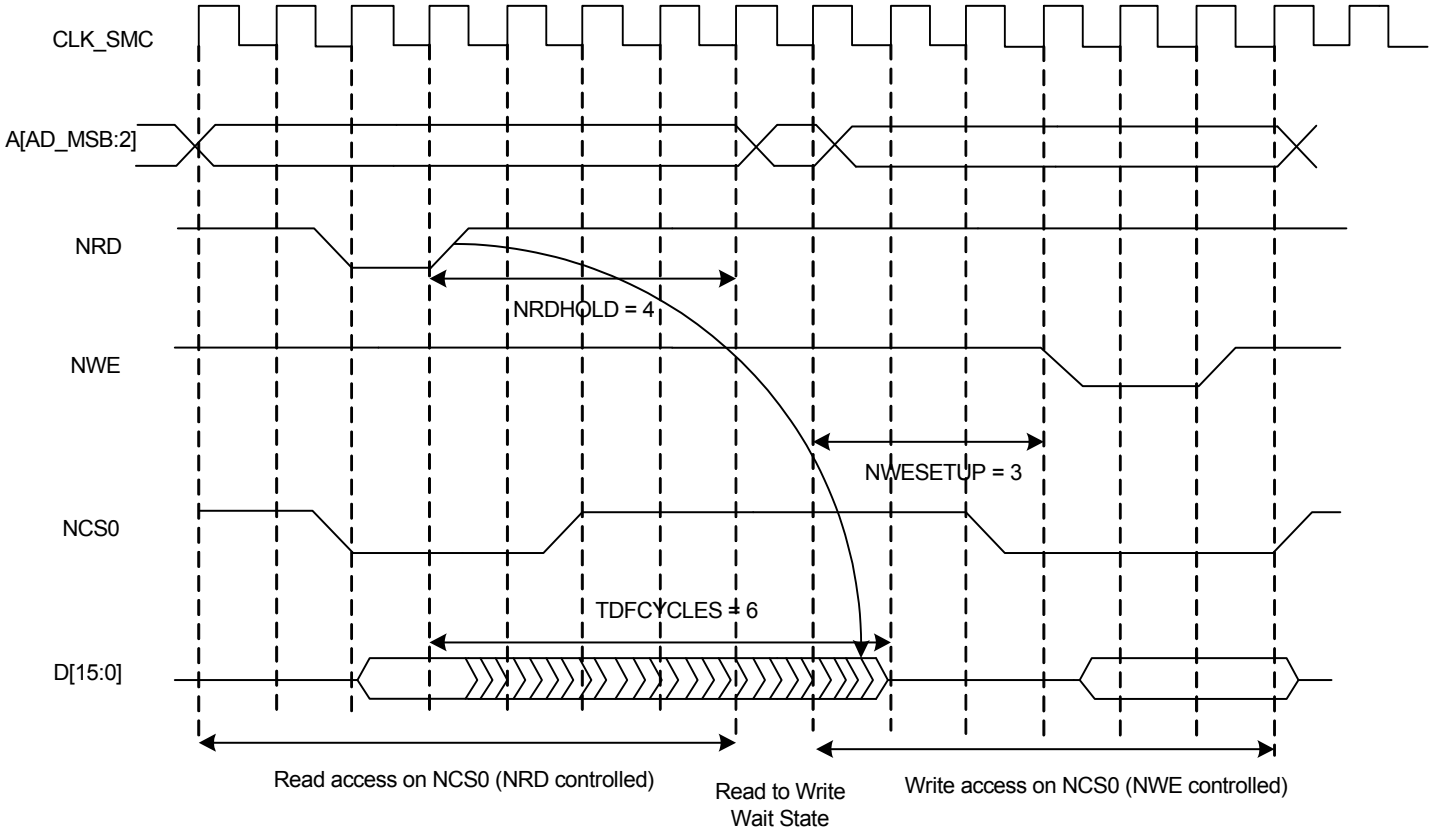
Figure 18-21 on page 328 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

NRDHOLD = 4; READMODE = 1 (NRD controlled)

NWESETUP = 3; WRITEMODE = 1 (NWE controlled)

TDFCYCLES = 6; TDFMODE = 1 (optimization enabled).

Figure 18-21. TDF Optimization: No TDF Wait States Are Inserted if the TDF Period Is over when the Next Access Begins



18.6.6.3 *TDF optimization disabled (MODE.TDFMODE = 0)*

When optimization is disabled, data float wait states are inserted at the end of the read transfer, so that the data float period is ended when the second access begins. If the hold period of the read1 controlling signal overlaps the data float period, no additional data float wait states will be inserted.

Figure 18-22 on page 329, Figure 18-23 on page 329 and Figure 18-24 on page 330 illustrate the cases:

- read access followed by a read access on another chip select.
- read access followed by a write access on another chip select.

- read access followed by a write access on the same chip select.

with no TDF optimization.

Figure 18-22. TDF Optimization Disabled (MODE.TDFMODE = 0). TDF Wait States between Two Read Accesses on Different Chip Selects.

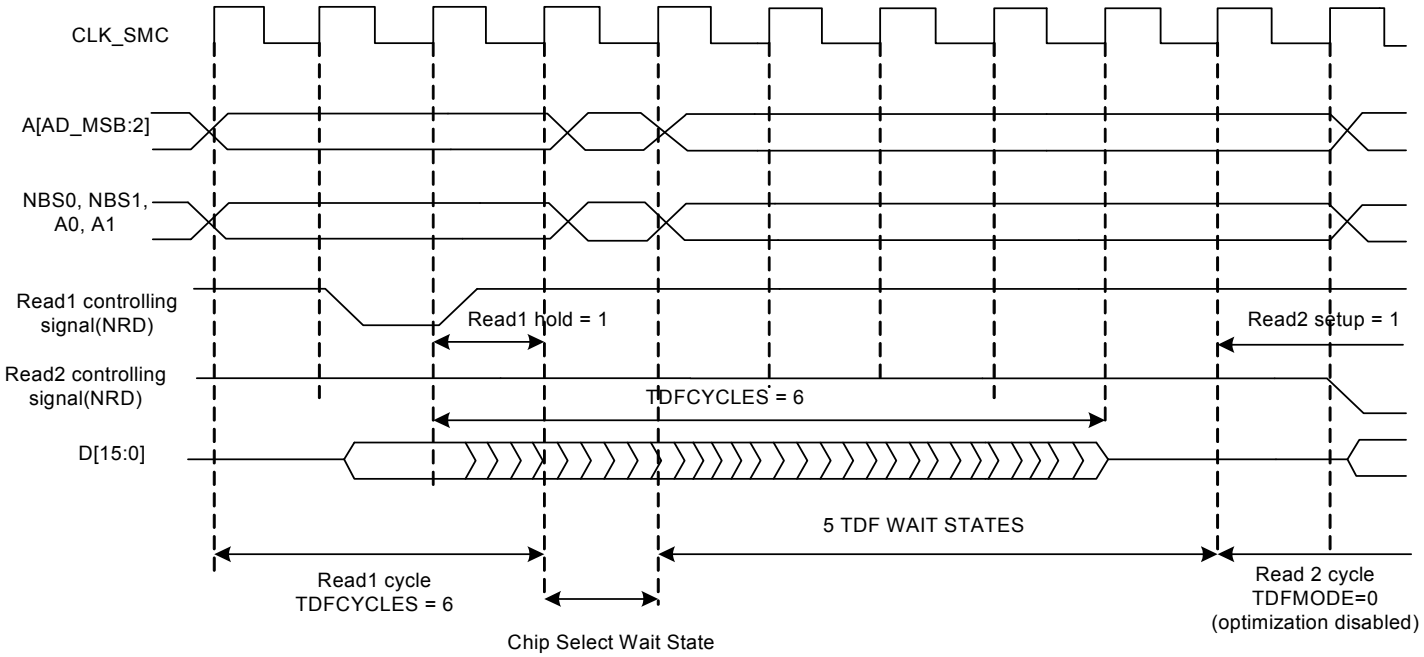


Figure 18-23. TDF Optimization Disabled (MODE.TDFMODE= 0). TDF Wait States between a Read and a Write Access on Different Chip Selects.

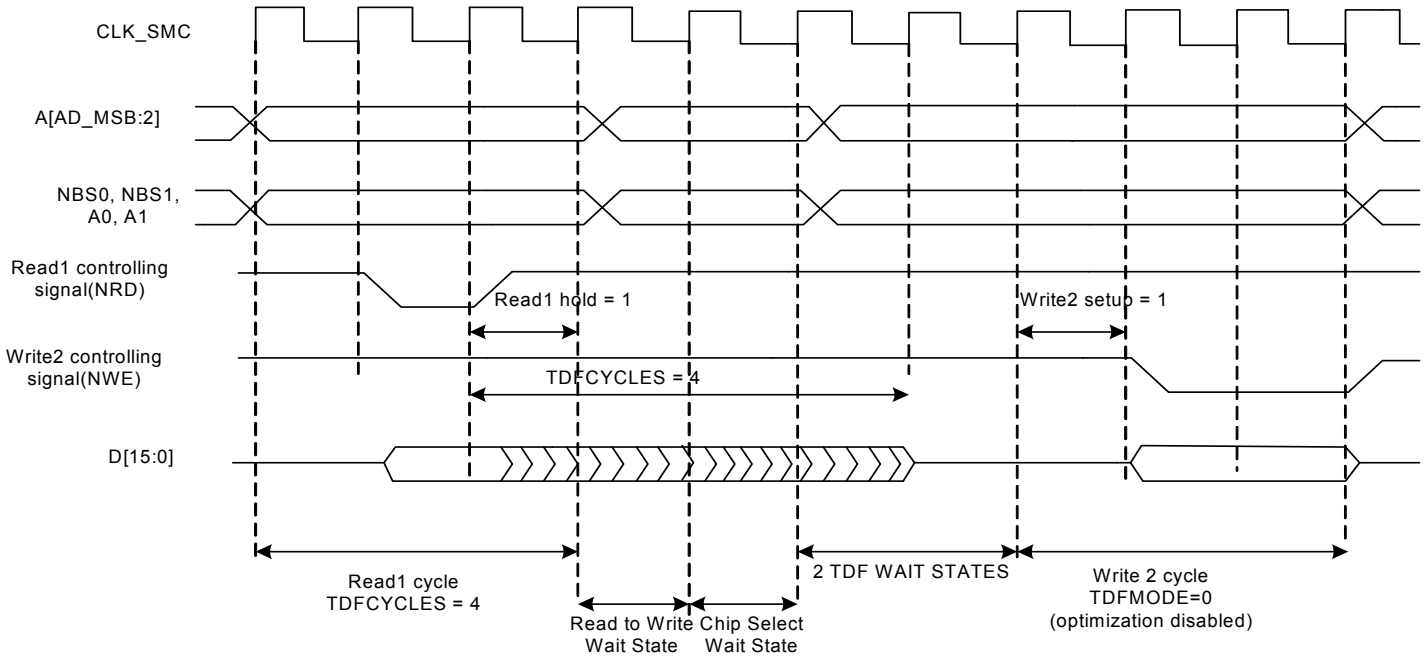
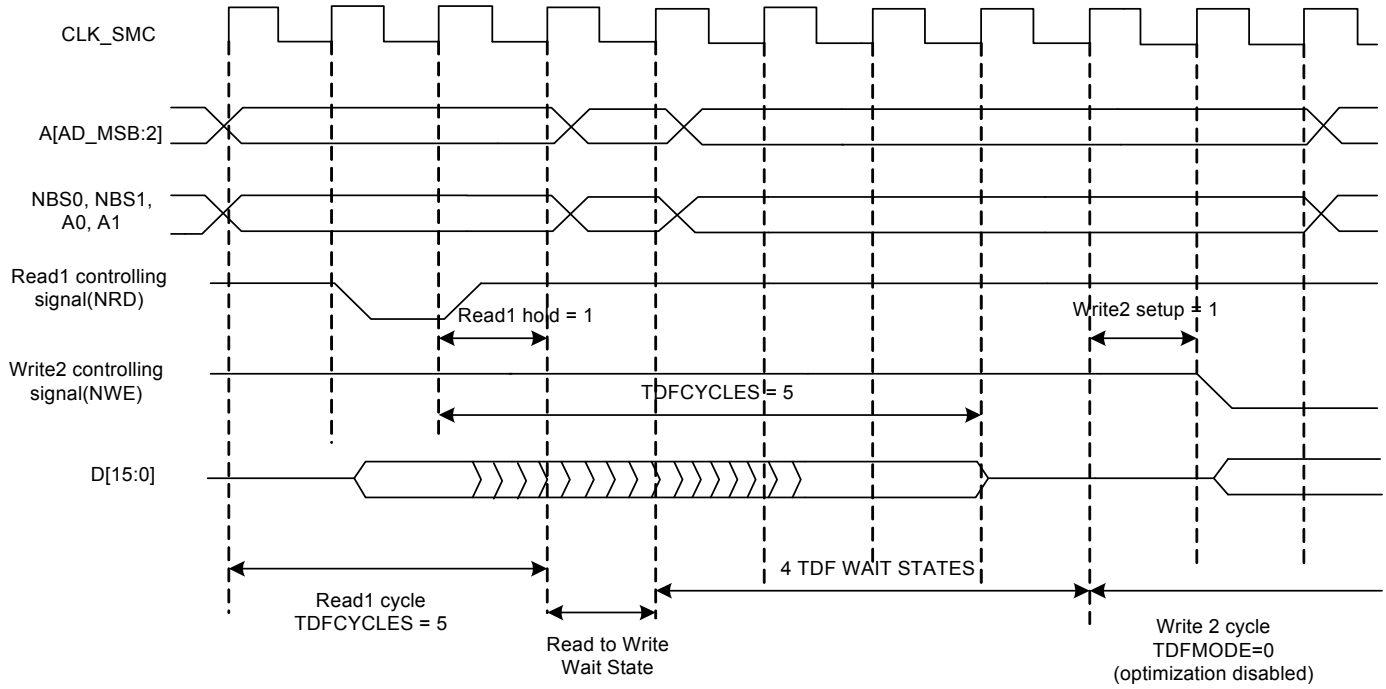


Figure 18-24. TDF Optimization Disabled (MODE.TDFMODE = 0). TDF Wait States between Read and Write accesses on the Same Chip Select.



18.6.7 External Wait

Any access can be extended by an external device using the NWAIT input signal of the SMC. The External Wait Mode field of the MODE register (MODE.EXNWMODE) on the corresponding chip select must be written to either two (frozen mode) or three (ready mode). When the MODE.EXNWMODE field is written to zero (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the read and write modes of the corresponding chip select.

18.6.7.1 Restriction

When one of the MODE.EXNWMODE is enabled, it is mandatory to program at least one hold cycle for the read/write controlling signal. For that reason, the NWAIT signal cannot be used in Page Mode (Section 18.6.9), or in Slow Clock Mode (Section 18.6.8).

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.

18.6.7.2 Frozen mode

When the external device asserts the NWAIT signal (active low), and after internal synchronization of this signal, the SMC state is frozen, i.e., SMC internal counters are frozen, and all control signals remain unchanged. When the synchronized NWAIT signal is deasserted, the SMC completes the access, resuming the access from the point where it was stopped. See Figure 18-25 on page 331. This mode must be selected when the external device uses the NWAIT signal to delay the access and to freeze the SMC.

The assertion of the NWAIT signal outside the expected period is ignored as illustrated in Figure 18-26 on page 332.

Figure 18-25. Write Access with NWAIT Assertion in Frozen Mode (MODE.EXNWMODE = 2).

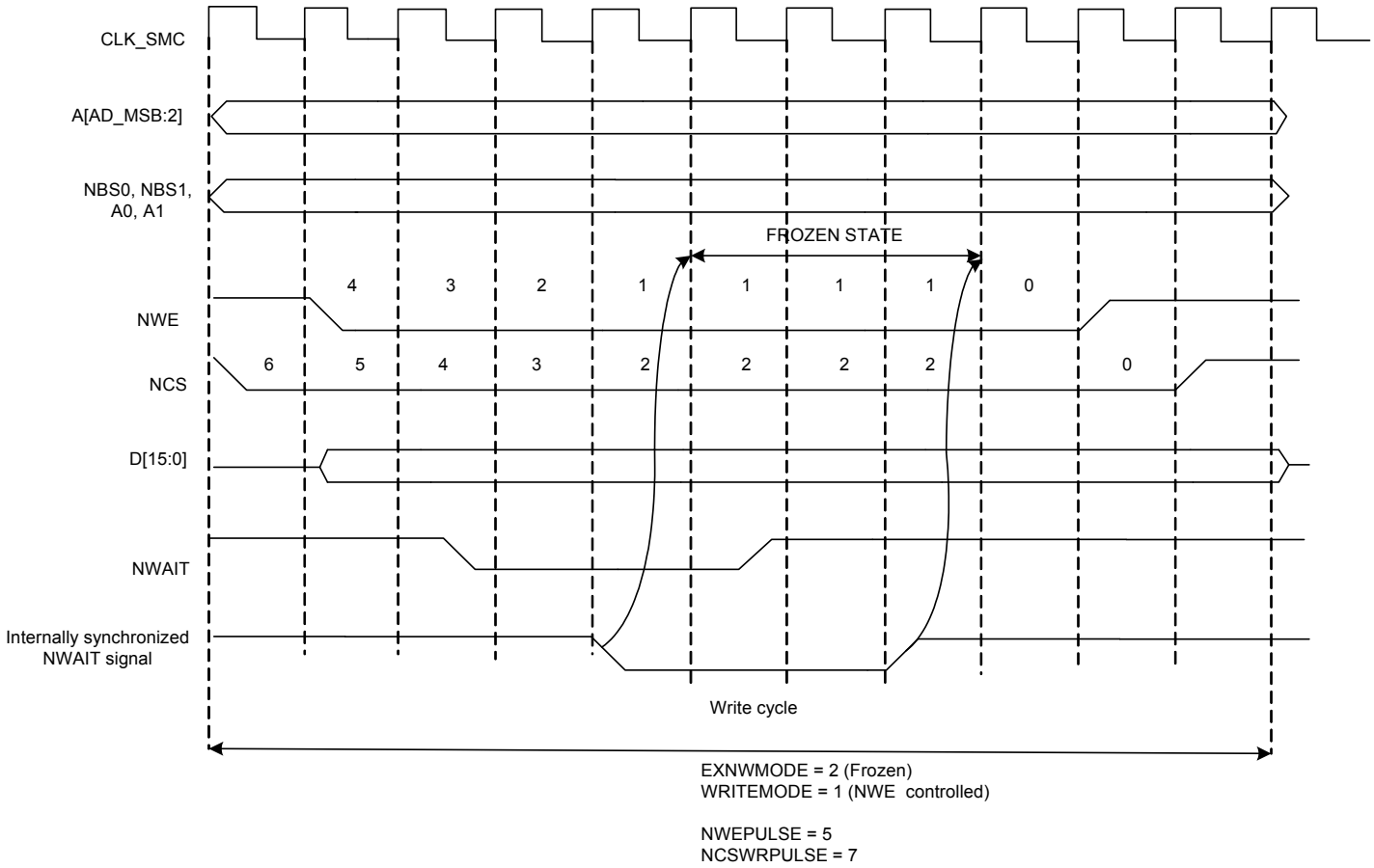
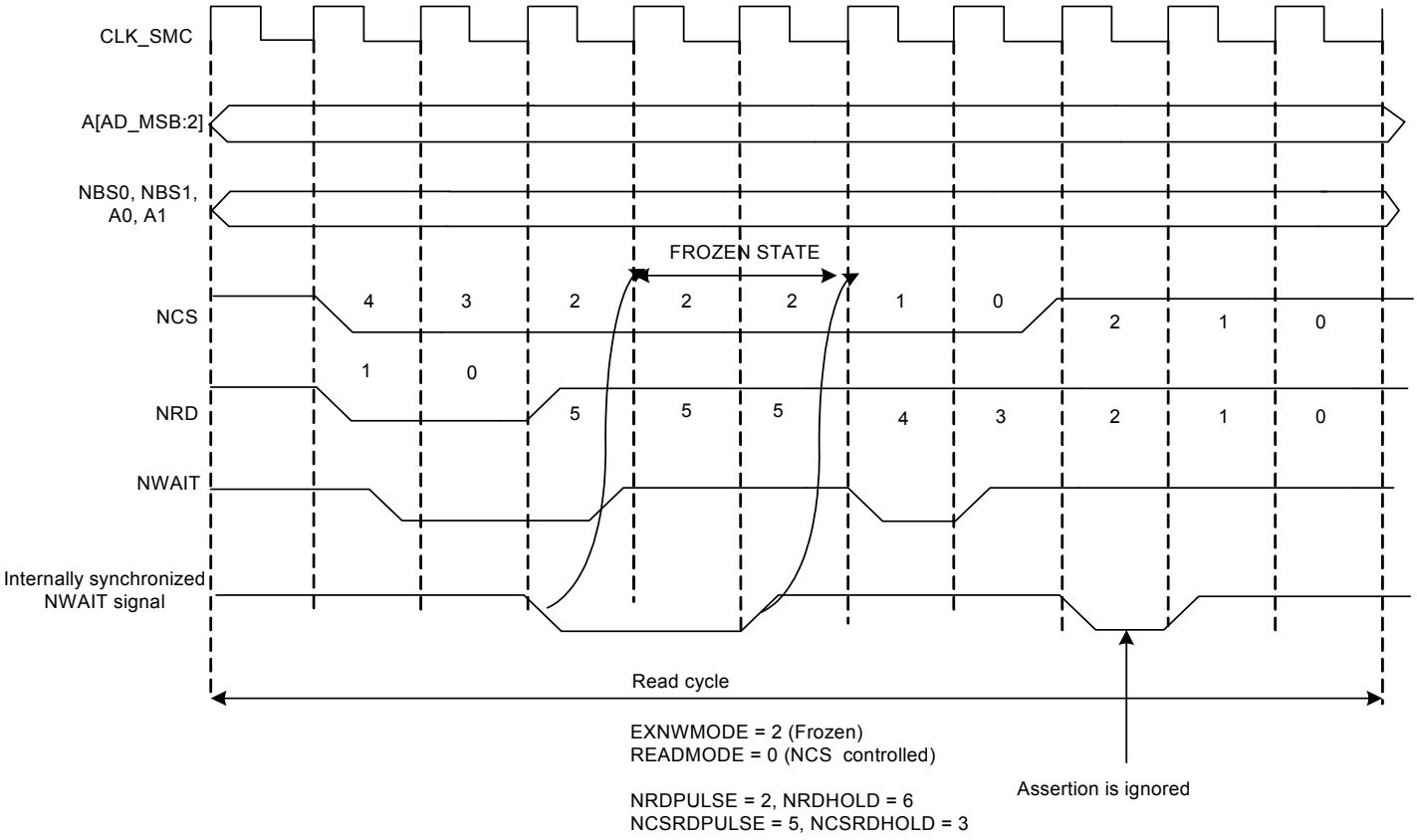


Figure 18-26. Read Access with NWAIT Assertion in Frozen Mode (MODE.EXNWMODE = 2).



18.6.7.3 Ready mode

In Ready mode (MODE.EXNWMODE = 3), the SMC behaves differently. Normally, the SMC begins the access by down counting the setup and pulse counters of the read/write controlling signal. In the last cycle of the pulse phase, the resynchronized NWAIT signal is examined.

If asserted, the SMC suspends the access as shown in Figure 18-27 on page 333 and Figure 18-28 on page 334. After deassertion, the access is completed: the hold step of the access is performed.

This mode must be selected when the external device uses deassertion of the NWAIT signal to indicate its ability to complete the read or write operation.

If the NWAIT signal is deasserted before the end of the pulse, or asserted after the end of the pulse of the controlling read/write signal, it has no impact on the access length as shown in Figure 18-28 on page 334.

Figure 18-27. NWAIT Assertion in Write Access: Ready Mode (MODE.EXNWMODE = 3).

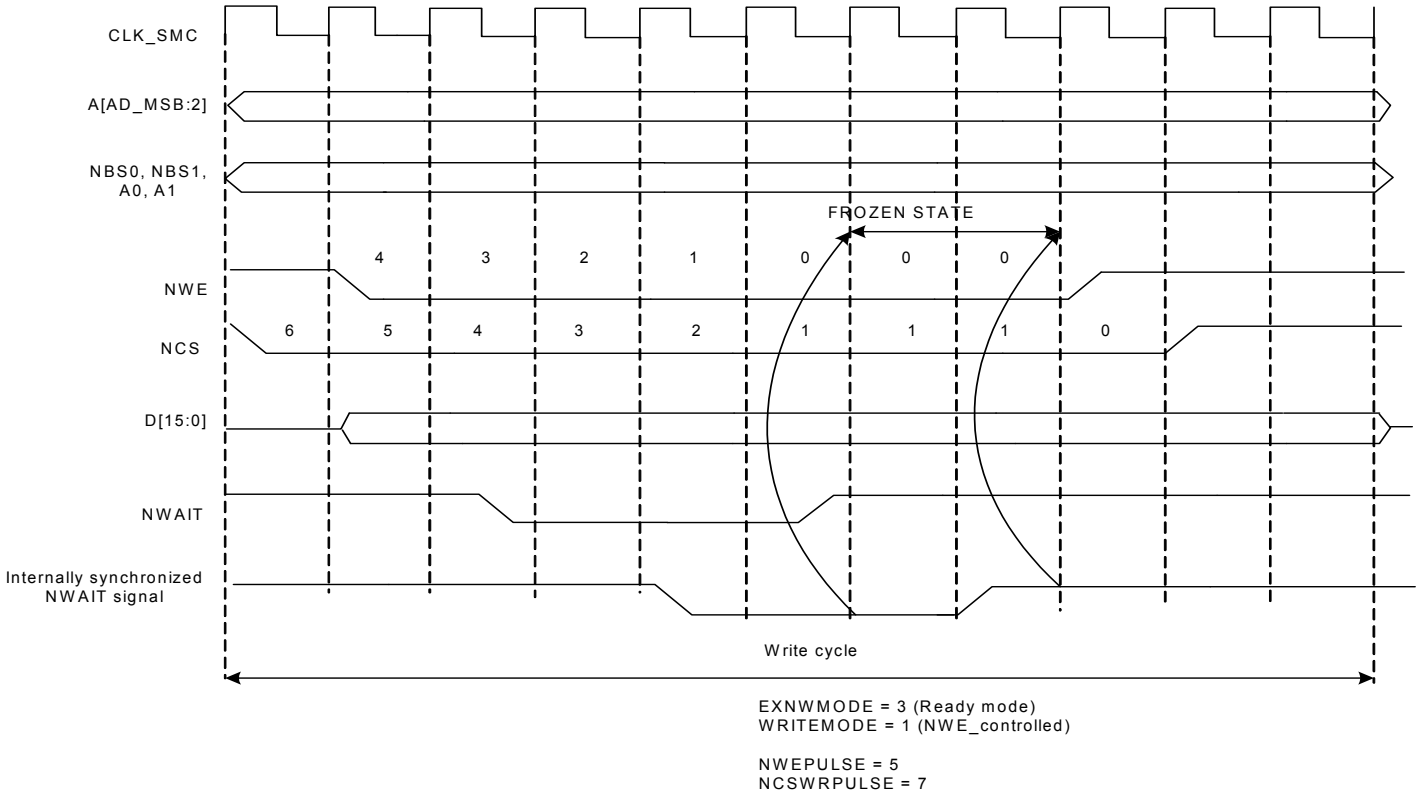
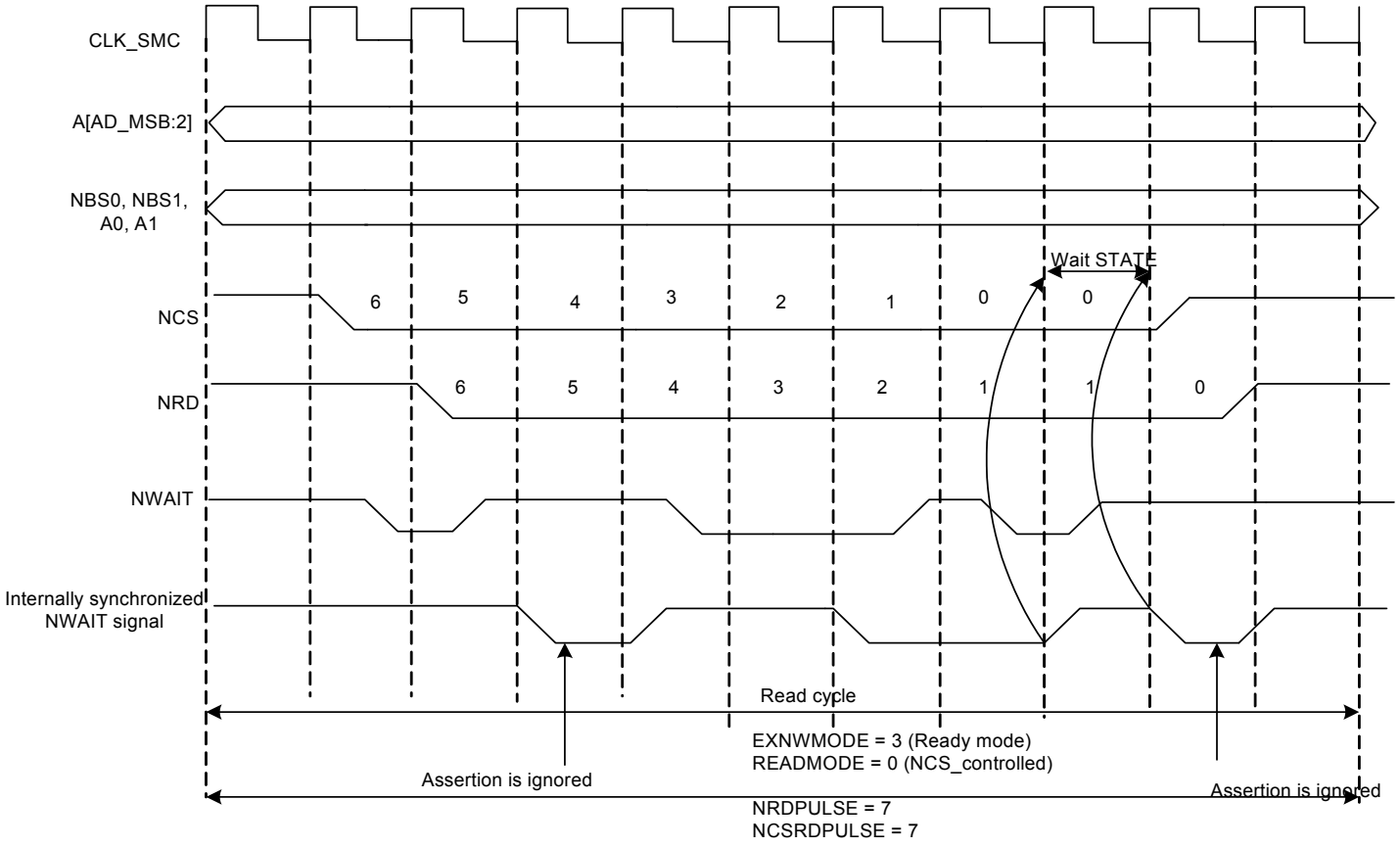


Figure 18-28. NWAIT Assertion in Read Access: Ready Mode (EXNWMODE = 3).



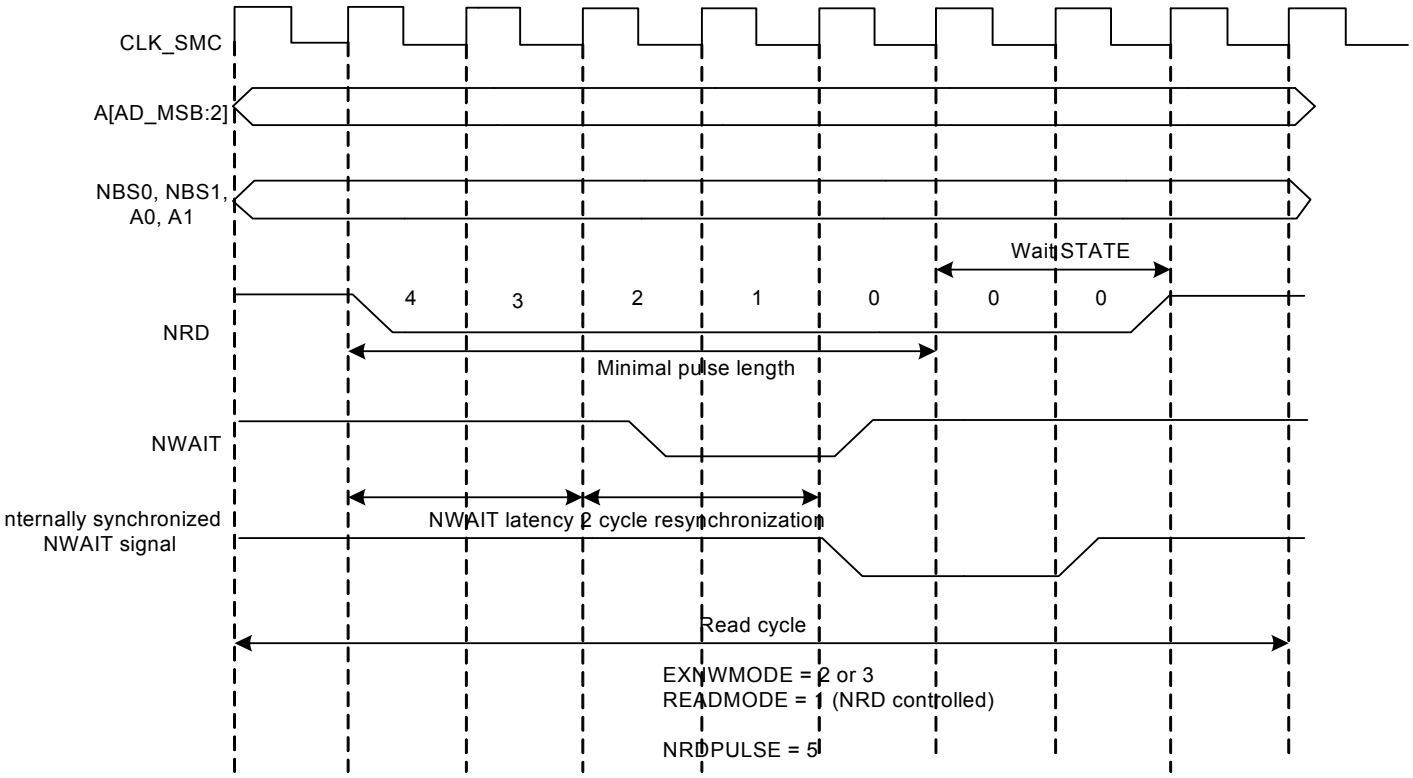
18.6.7.4 *NWAIT latency and read/write timings*

There may be a latency between the assertion of the read/write controlling signal and the assertion of the NWAIT signal by the device. The programmed pulse length of the read/write controlling signal must be at least equal to this latency plus the two cycles of resynchronization plus one cycle. Otherwise, the SMC may enter the hold state of the access without detecting the NWAIT signal assertion. This is true in frozen mode as well as in ready mode. This is illustrated on Figure 18-29 on page 335.

When the MODE.EXNWMODE field is enabled (ready or frozen), the user must program a pulse length of the read and write controlling signal of at least:

$$\text{minimal pulse length} = \text{NWAIT latency} + 2 \text{ synchronization cycles} + 1 \text{ cycle}$$

Figure 18-29. NWAIT Latency



18.6.8 Slow Clock Mode

The SMC is able to automatically apply a set of “slow clock mode” read/write waveforms when an internal signal driven by the SMC’s Power Management Controller is asserted because CLK_SMC has been turned to a very slow clock rate (typically 32 kHz clock rate). In this mode, the user-programmed waveforms are ignored and the slow clock mode waveforms are applied. This mode is provided so as to avoid reprogramming the User Interface with appropriate waveforms at very slow clock rate. When activated, the slow mode is active on all chip selects.

18.6.8.1 Slow clock mode waveforms

Figure 18-30 on page 336 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 18-5 on page 336 indicates the value of read and write parameters in slow clock mode.

Figure 18-30. Read and Write Cycles in Slow Clock Mode

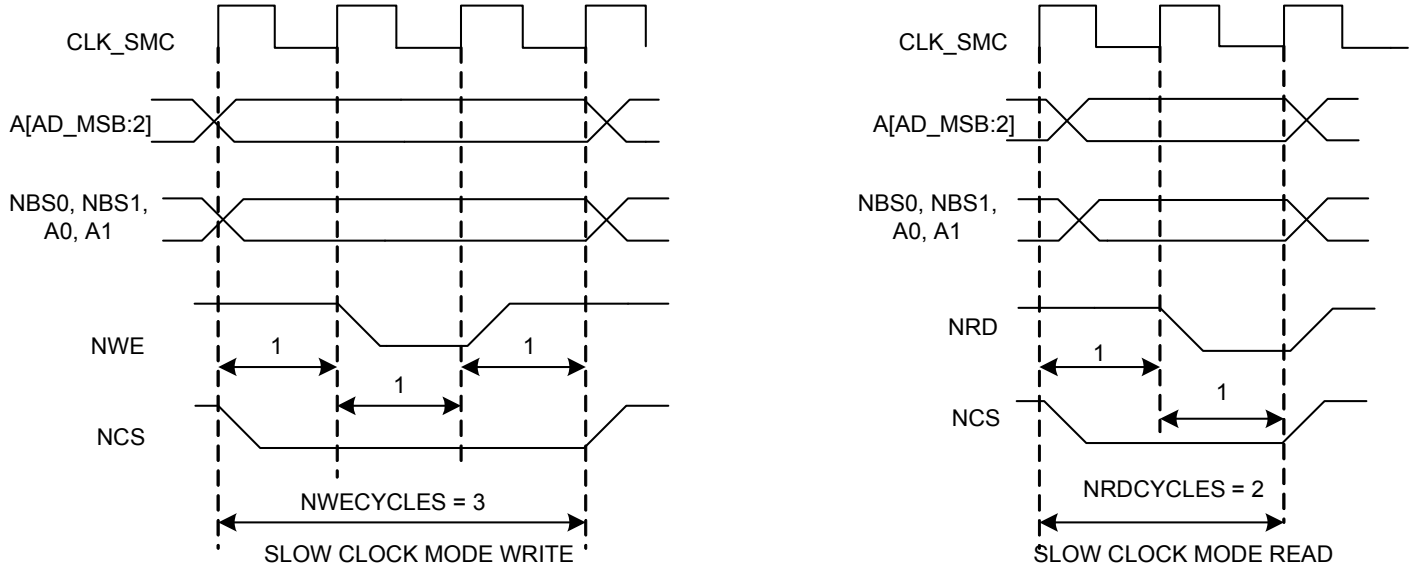


Table 18-5. Read and Write Timing Parameters in Slow Clock Mode

Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRDSETUP	1	NWESETUP	1
NRDPULSE	1	NWEPULSE	1
NCSRDSETUP	0	NCSWRSETUP	0
NCSRDPULSE	2	NCSWRPULSE	3
NRDCYCLE	2	NWECYCLE	3

18.6.8.2 *Switching from (to) slow clock mode to (from) normal mode*

When switching from slow clock mode to the normal mode, the current slow clock mode transfer is completed at high clock rate, with the set of slow clock mode parameters. See Figure 18-31 on page 337. The external device may not be fast enough to support such timings.

Figure 18-32 on page 338 illustrates the recommended procedure to properly switch from one mode to the other.

Figure 18-31. Clock Rate Transition Occurs while the SMC is Performing a Write Operation

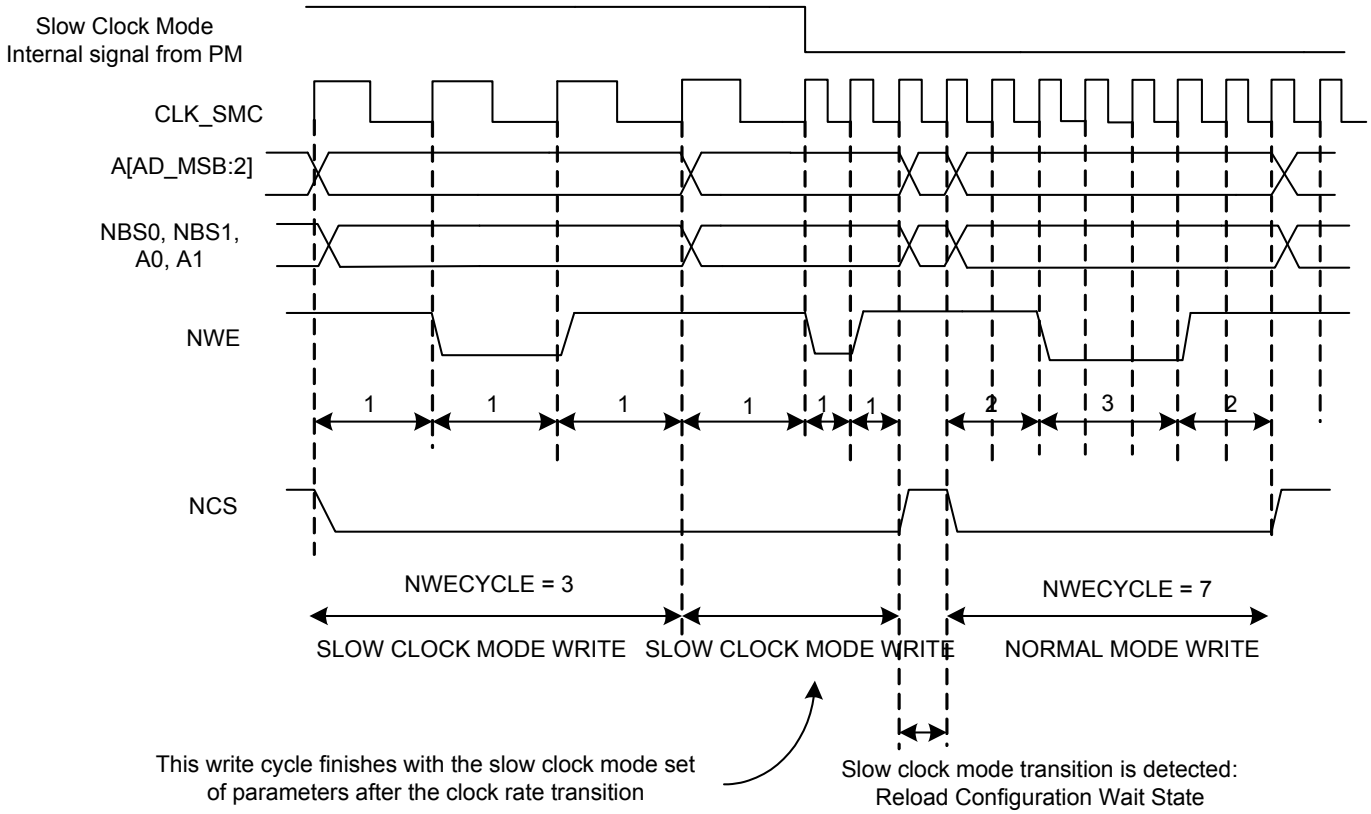
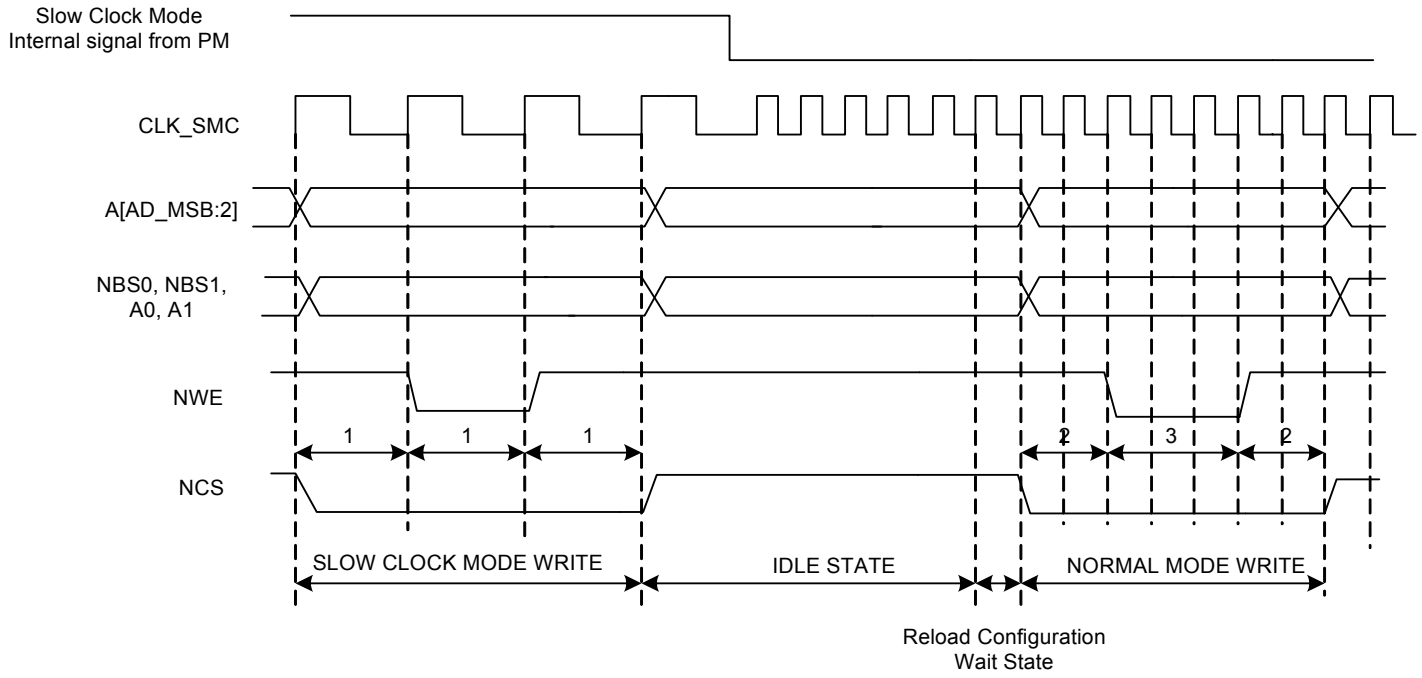


Figure 18-32. Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



18.6.9 Asynchronous Page Mode

The SMC supports asynchronous burst reads in page mode, providing that the Page Mode Enabled bit is written to one in the MODE register (MODE.PMEN). The page size must be configured in the Page Size field in the MODE register (MODE.PS) to 4, 8, 16, or 32 bytes.

The page defines a set of consecutive bytes into memory. A 4-byte page (resp. 8-, 16-, 32-byte page) is always aligned to 4-byte boundaries (resp. 8-, 16-, 32-byte boundaries) of memory. The MSB of data address defines the address of the page in memory, the LSB of address define the address of the data in the page as detailed in Table 18-6 on page 338.

With page mode memory devices, the first access to one page (t_{pa}) takes longer than the subsequent accesses to the page (t_{sa}) as shown in Figure 18-33 on page 339. When in page mode, the SMC enables the user to define different read timings for the first access within one page, and next accesses within the page.

Table 18-6. Page Address and Data Address within a Page

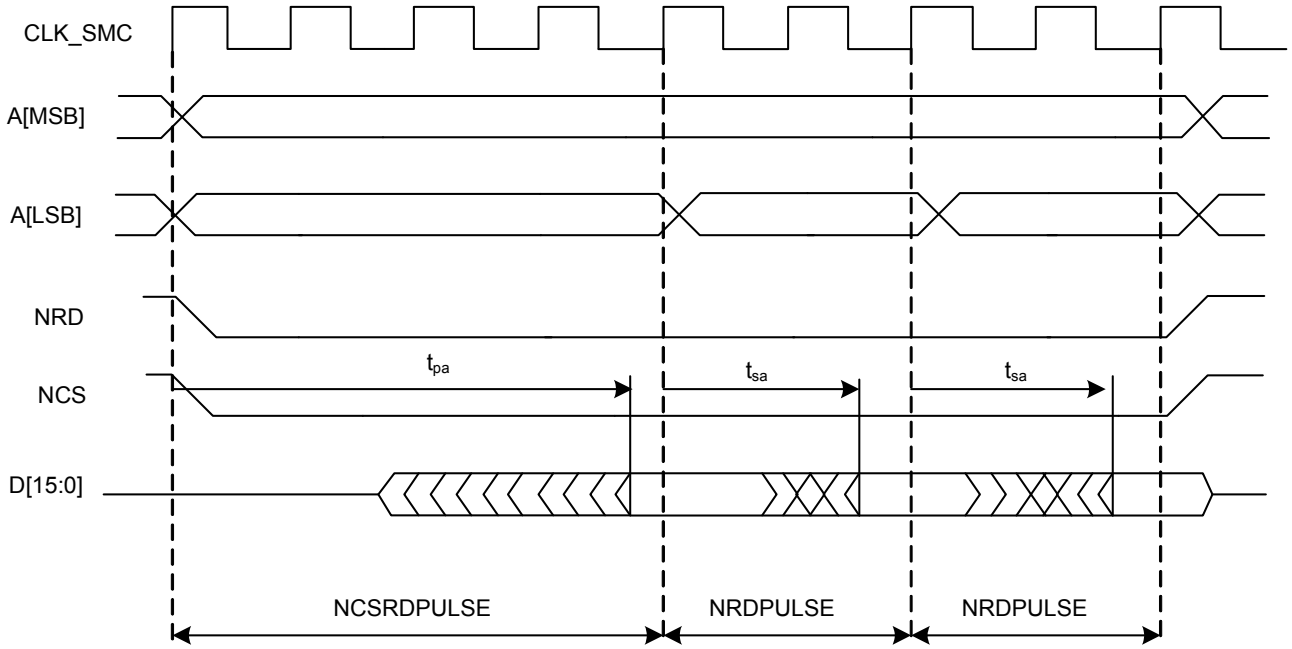
Page Size	Page Address(1)	Data Address in the Page(2)
4 bytes	A[23:2]	A[1:0]
8 bytes	A[23:3]	A[2:0]
16 bytes	A[23:4]	A[3:0]
32 bytes	A[23:5]	A[4:0]

Notes: 1. A denotes the address bus of the memory device
2. For 16-bit devices, the bit 0 of address is ignored.

18.6.9.1 Protocol and timings in page mode

Figure 18-33 on page 339 shows the NRD and NCS timings in page mode access.

Figure 18-33. Page Mode Read Protocol (Address MSB and LSB Are Defined in Table 18-6 on page 338)



The NRD and NCS signals are held low during all read transfers, whatever the programmed values of the setup and hold timings in the User Interface may be. Moreover, the NRD and NCS timings are identical. The pulse length of the first access to the page is defined with the PULSE.NCSRDPULSE field value. The pulse length of subsequent accesses within the page are defined using the PULSE.NRDPULSE field value.

In page mode, the programming of the read timings is described in Table 18-7 on page 339:

Table 18-7. Programming of Read Timings in Page Mode

Parameter	Value	Definition
READMODE	'x'	No impact
NCSRDSETUP	'x'	No impact
NCSRDPULSE	t_{pa}	Access time of first access to the page
NRDSETUP	'x'	No impact
NRDPULSE	t_{sa}	Access time of subsequent accesses in the page
NRDCYCLE	'x'	No impact

The SMC does not check the coherency of timings. It will always apply the NCSRDPULSE timings as page access timing (t_{pa}) and the NRDPULSE for accesses to the page (t_{sa}), even if the programmed value for t_{pa} is shorter than the programmed value for t_{sa}.

18.6.9.2 Byte access type in page mode

The byte access type configuration remains active in page mode. For 16-bit or 32-bit page mode devices that require byte selection signals, configure the MODE.BAT bit to zero (byte select access type).

18.6.9.3 Page mode restriction

The page mode is not compatible with the use of the NWAIT signal. Using the page mode and the NWAIT signal may lead to unpredictable behavior.

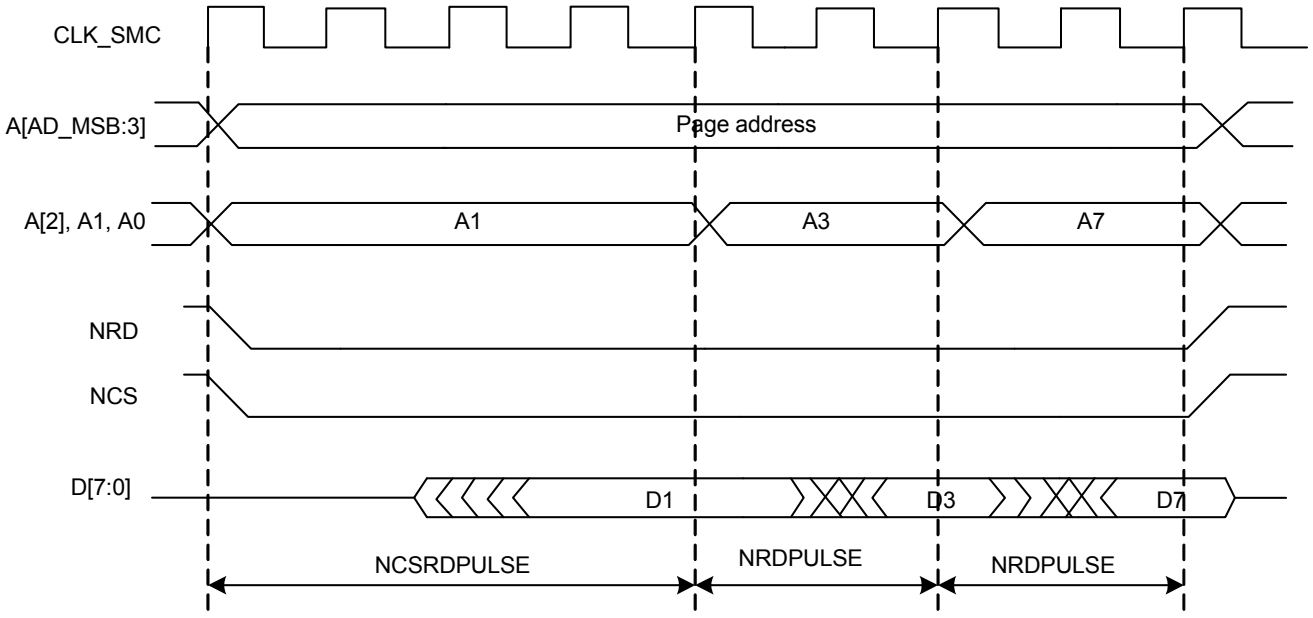
18.6.9.4 Sequential and non-sequential accesses

If the chip select and the MSB of addresses as defined in Table 18-6 on page 338 are identical, then the current access lies in the same page as the previous one, and no page break occurs.

Using this information, all data within the same page, sequential or not sequential, are accessed with a minimum access time (t_{sa}). Figure 18-34 on page 340 illustrates access to an 8-bit memory device in page mode, with 8-byte pages. Access to D1 causes a page access with a long access time (t_{pa}). Accesses to D3 and D7, though they are not sequential accesses, only require a short access time (t_{sa}).

If the MSB of addresses are different, the SMC performs the access of a new page. In the same way, if the chip select is different from the previous access, a page break occurs. If two sequential accesses are made to the page mode memory, but separated by an other internal or external peripheral access, a page break occurs on the second access because the chip select of the device was deasserted between both accesses.

Figure 18-34. Access to Non-sequential Data within the Same Page



18.7 User Interface

The SMC is programmed using the registers listed in Table 18-8 on page 341. For each chip select, a set of four registers is used to program the parameters of the external device connected on it. In Table 18-8 on page 341, “CS_number” denotes the chip select number. Sixteen bytes (0x10) are required per chip select.

The user must complete writing the configuration by writing anyone of the Mode Registers.

Table 18-8. SMC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00 + CS_number*0x10	Setup Register	SETUP	Read/Write	0x01010101
0x04 + CS_number*0x10	Pulse Register	PULSE	Read/Write	0x01010101
0x08 + CS_number*0x10	Cycle Register	CYCLE	Read/Write	0x00030003
0x0C + CS_number*0x10	Mode Register	MODE	Read/Write	0x10002103

18.7.1 Setup Register

Register Name: SETUP

Access Type: Read/Write

Offset: 0x00 + CS_number*0x10

Reset Value: 0x01010101

31	30	29	28	27	26	25	24
–	–	NCSRDSETUP					

23	22	21	20	19	18	17	16
–	–	NRDSETUP					

15	14	13	12	11	10	9	8
–	–	NCSWRSETUP					

7	6	5	4	3	2	1	0
–	–	NWESETUP					

- **NCSRDSETUP: NCS Setup Length in READ Access**
  In read access, the NCS signal setup length is defined as:

$$\text{NCS Setup Length in read access} = (128 \times NCSRDSETUP[5] + NCSRDSETUP[4:0]) \text{ clock cycles}$$

- **NRDSETUP: NRD Setup Length**
  The NRD signal setup length is defined in clock cycles as:

$$\text{NRD Setup Length} = (128 \times NRDSETUP[5] + NRDSETUP[4:0]) \text{ clock cycles}$$

- **NCSWRSETUP: NCS Setup Length in WRITE Access**
  In write access, the NCS signal setup length is defined as:

$$\text{NCS Setup Length in write access} = (128 \times NCSWRSETUP[5] + NCSWRSETUP[4:0]) \text{ clock cycles}$$

- **NWESETUP: NWE Setup Length**
  The NWE signal setup length is defined as:

$$\text{NWE Setup Length} = (128 \times NWESETUP[5] + NWESETUP[4:0]) \text{ clock cycles}$$

18.7.2 Pulse Register

Register Name: PULSE

Access Type: Read/Write

Offset: 0x04 + CS_number*0x10

Reset Value: 0x01010101

31	30	29	28	27	26	25	24
–	NCSRDPULSE						

23	22	21	20	19	18	17	16
–	NRDPULSE						

15	14	13	12	11	10	9	8
–	NCSWRPULSE						

7	6	5	4	3	2	1	0
–	NWEPULSE						

- **NCSRDPULSE: NCS Pulse Length in READ Access**

  In standard read access, the NCS signal pulse length is defined as:

  $$\text{NCS Pulse Length in read access} = (256 \times NCSRDPULSE[6] + NCSRDPULSE[5:0]) \text{ clock cycles}$$

  The NCS pulse length must be at least one clock cycle.

  In page mode read access, the NCSRDPULSE field defines the duration of the first access to one page.

- **NRDPULSE: NRD Pulse Length**

  In standard read access, the NRD signal pulse length is defined in clock cycles as:

  $$\text{NRD Pulse Length} = (256 \times NRDPULSE[6] + NRDPULSE[5:0]) \text{ clock cycles}$$

  The NRD pulse length must be at least one clock cycle.

  In page mode read access, the NRDPULSE field defines the duration of the subsequent accesses in the page.

- **NCSWRPULSE: NCS Pulse Length in WRITE Access**

  In write access, the NCS signal pulse length is defined as:

  $$\text{NCS Pulse Length in write access} = (256 \times NCSWRPULSE[6] + NCSWRPULSE[5:0]) \text{ clock cycles}$$

  The NCS pulse length must be at least one clock cycle.

- **NWEPULSE: NWE Pulse Length**

  The NWE signal pulse length is defined as:

  $$\text{NWE Pulse Length} = (256 \times NWEPULSE[6] + NWEPULSE[5:0]) \text{ clock cycles}$$

  The NWE pulse length must be at least one clock cycle.

18.7.3 Cycle Register

Register Name: CYCLE

Access Type: Read/Write

Offset: 0x08 + CS_number*0x10

Reset Value: 0x00030003

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	NRDCYCLE[8]

23	22	21	20	19	18	17	16
NRDCYCLE[7:0]							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NWECYCLE[8]

7	6	5	4	3	2	1	0
NWECYCLE[7:0]							

- **NRDCYCLE[8:0]: Total Read Cycle Length**

  The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as:

  $$\text{Read Cycle Length} = (256 \times NRDCYCLE[8:7] + NRDCYCLE[6:0]) \text{ clock cycles}$$

- **NWECYCLE[8:0]: Total Write Cycle Length**

  The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as:

  $$\text{Write Cycle Length} = (256 \times NWECYCLE[8:7] + NWECYCLE[6:0]) \text{ clock cycles}$$

18.7.4 Mode Register

Register Name: MODE

Access Type: Read/Write

Offset: 0x0C + CS_number*0x10

Reset Value: 0x10002103

31	30	29	28	27	26	25	24
–	–	PS		–	–	–	PMEN

23	22	21	20	19	18	17	16
–	–	–	TDFMODE	TDFCYCLES			

15	14	13	12	11	10	9	8
–	–	DBW		–	–	–	BAT

7	6	5	4	3	2	1	0
–	–	EXNWMODE		–	–	WRITEMODE	READMODE

- **PS: Page Size**

  If page mode is enabled, this field indicates the size of the page in bytes.

PS	Page Size
0	4-byte page
1	8-byte page
2	16-byte page
3	32-byte page

- **PMEN: Page Mode Enabled**

  1: Asynchronous burst read in page mode is applied on the corresponding chip select.

  0: Standard read is applied.

- **TDFMODE: TDF Optimization**

  1: TDF optimization is enabled. The number of TDF wait states is optimized using the setup period of the next read/write access.

  0: TDF optimization is disabled.The number of TDF wait states is inserted before the next access begins.

- **TDFCYCLES: Data Float Time**

  This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDFCYCLES period. The external bus cannot be used by another chip select during TDFCYCLES plus one cycles. From 0 up to 15 TDFCYCLES can be set.

- **DBW: Data Bus Width**

DBW	Data Bus Width
0	8-bit bus
1	16-bit bus
2	Reserved
3	Reserved

- **BAT: Byte Access Type**

  This field is used only if DBW defines a 16-bit data bus.

BAT	Byte Access Type
0	Byte select access type: Write operation is controlled using NCS, NWE, NBS0, NBS1 Read operation is controlled using NCS, NRD, NBS0, NBS1
1	Byte write access type: Write operation is controlled using NCS, NWR0, NWR1 Read operation is controlled using NCS and NRD

- **EXNWMODE: External WAIT Mode**

  The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

EXNWMODE	External NWAIT Mode
0	Disabled: the NWAIT input signal is ignored on the corresponding chip select.
1	Reserved
2	Frozen Mode: if asserted, the NWAIT signal freezes the current read or write cycle. after deassertion, the read or write cycle is resumed from the point where it was stopped.
3	Ready Mode: the NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

- **WRITEMODE: Write Mode**

  1: The write operation is controlled by the NWE signal. If TDF optimization is enabled (TDFMODE =1), TDF wait states will be inserted after the setup of NWE.

  0: The write operation is controlled by the NCS signal. If TDF optimization is enabled (TDFMODE =1), TDF wait states will be inserted after the setup of NCS.

- **READMODE: Read Mode**

READMODE	Read Access Mode
0	The read operation is controlled by the NCS signal. If TDF are programmed, the external bus is marked busy after the rising edge of NCS. If TDF optimization is enabled (TDFMODE = 1), TDF wait states are inserted after the setup of NCS.
1	The read operation is controlled by the NRD signal. If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD. If TDF optimization is enabled (TDFMODE =1), TDF wait states are inserted after the setup of NRD.

19. SDRAM Controller (SDRAMC)

Rev: 2.2.0.4

19.1 Features

- 128-Mbytes address space
- Numerous configurations supported
  - 2K, 4K, 8K row address memory parts
  - SDRAM with two or four internal banks
  - SDRAM with 16-bit data path
- Programming facilities
  - Word, halfword, byte access
  - Automatic page break when memory boundary has been reached
  - Multibank ping-pong access
  - Timing parameters specified by software
  - Automatic refresh operation, refresh rate is programmable
  - Automatic update of DS, TCR and PASR parameters (mobile SDRAM devices)
- Energy-saving capabilities
  - Self-refresh, power-down, and deep power-down modes supported
  - Supports mobile SDRAM devices
- Error detection
  - Refresh error interrupt
- SDRAM power-up initialization by software
- CAS latency of one, two, and three supported
- Auto Precharge command not used

19.2 Overview

The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to an external 16-bit SDRAM device. The page size supports ranges from 2048 to 8192 and the number of columns from 256 to 2048. It supports byte (8-bit) and halfword (16-bit) accesses.

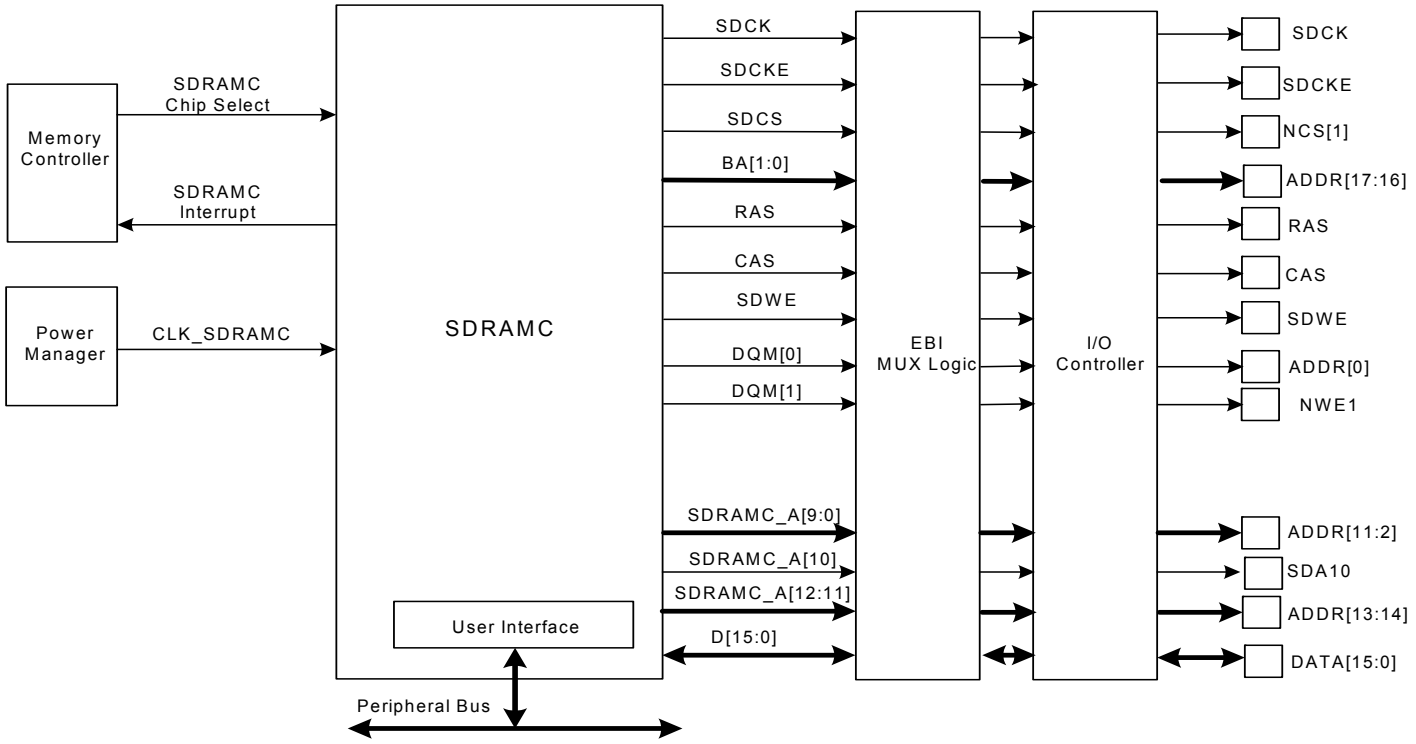
The SDRAMC supports a read or write burst length of one location. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.

The SDRAMC supports a CAS latency of one, two, or three and optimizes the read access depending on the frequency.

The different modes available (self refresh, power-down, and deep power-down modes) minimize power consumption on the SDRAM device.

19.3 Block Diagram

Figure 19-1. SDRAM Controller Block Diagram



19.4 I/O Lines Description

Table 19-1. I/O Lines Description

Name	Description	Type	Active Level
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Chip Select	Output	Low
BA[1:0]	Bank Select Signals	Output	
RAS	Row Signal	Output	Low
CAS	Column Signal	Output	Low
SDWE	SDRAM Write Enable	Output	Low

Table 19-1. I/O Lines Description

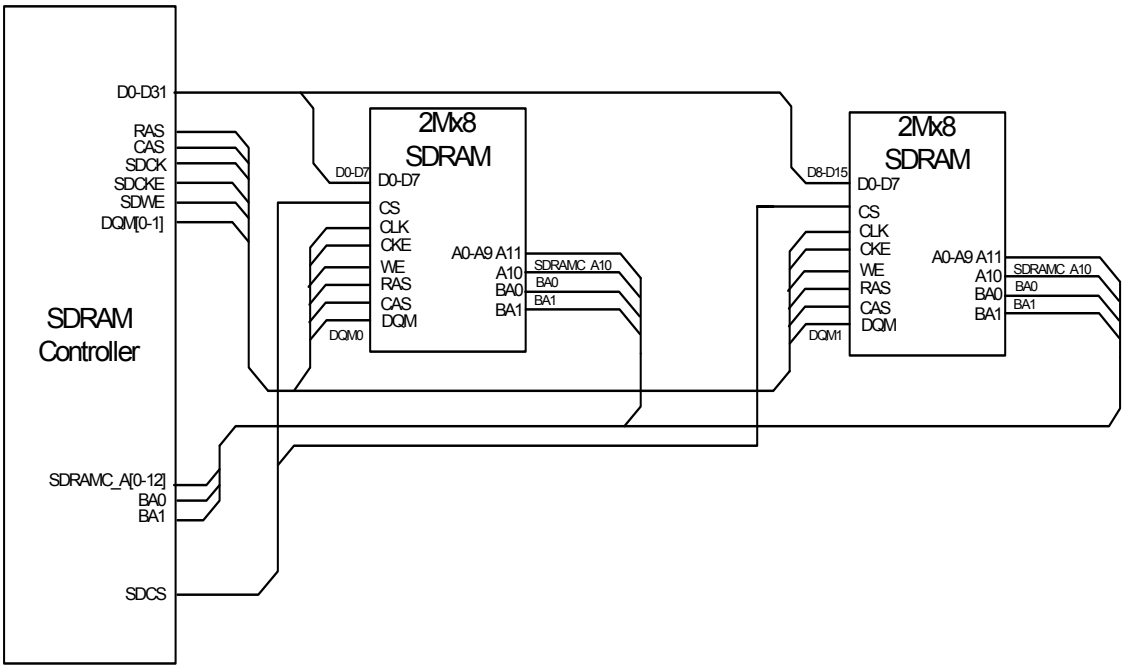
Name	Description	Type	Active Level
DQM[1:0]	Data Mask Enable Signals	Output	High
SDRAMC_A[12:0]	Address Bus	Output	
D[15:0]	Data Bus	Input/Output	

19.5 Application Example

19.5.1 Hardware Interface

Figure 19-2 on page 350 shows an example of SDRAM device connection using a 16-bit data bus width. It is important to note that this example is given for a direct connection of the devices to the SDRAMC, without External Bus Interface or I/O Controller multiplexing.

Figure 19-2. SDRAM Controller Connections to SDRAM Devices: 16-bit Data Bus Width



19.5.2 Software Interface

The SDRAM address space is organized into banks, rows, and columns. The SDRAMC allows mapping different memory types according to the values set in the SDRAMC Configuration Register (CR).

The SDRAMC's function is to make the SDRAM device access protocol transparent to the user. Table 19-2 on page 351 to Table 19-4 on page 351 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

19.5.2.1 16-bit memory data bus width

Table 19-2. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						BA[1:0]	Row[10:0]												Column[7:0]								M0
					BA[1:0]	Row[10:0]													Column[8:0]								M0
				BA[1:0]	Row[10:0]														Column[9:0]								M0
			BA[1:0]	Row[10:0]															Column[10:0]								M0

Table 19-3. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					BA[1:0]	Row[11:0]													Column[7:0]								M0
				BA[1:0]	Row[11:0]														Column[8:0]								M0
			BA[1:0]	Row[11:0]															Column[9:0]								M0
		BA[1:0]	Row[11:0]																Column[10:0]								M0

Table 19-4. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				BA[1:0]	Row[12:0]														Column[7:0]								M0
			BA[1:0]	Row[12:0]															Column[8:0]								M0
		BA[1:0]	Row[12:0]																Column[9:0]								M0
	BA[1:0]	Row[12:0]																	Column[10:0]								M0

Notes: 1. M0 is the byte address inside a 16-bit halfword.

19.6 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

19.6.1 I/O Lines

The SDRAMC module signals pass through the External Bus Interface (EBI) module where they are multiplexed. The user must first configure the I/O controller to assign the EBI pins corresponding to SDRAMC signals to their peripheral function. If I/O lines of the EBI corresponding to SDRAMC signals are not used by the application, they can be used for other purposes by the I/O Controller.

19.6.2 Power Management

The SDRAMC must be properly stopped before entering in reset mode, i.e., the user must issue a Deep power mode command in the Mode (MD) register and wait for the command to be completed.

19.6.3 Clocks

The clock for the SDRAMC bus interface (CLK_SDRAMC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the SDRAMC before disabling the clock, to avoid freezing the SDRAMC in an undefined state.

19.6.4 Interrupts

The SDRAMC interrupt request line is connected to the interrupt controller. Using the SDRAMC interrupt requires the interrupt controller to be programmed first.

19.7 Functional Description

19.7.1 SDRAM Device Initialization

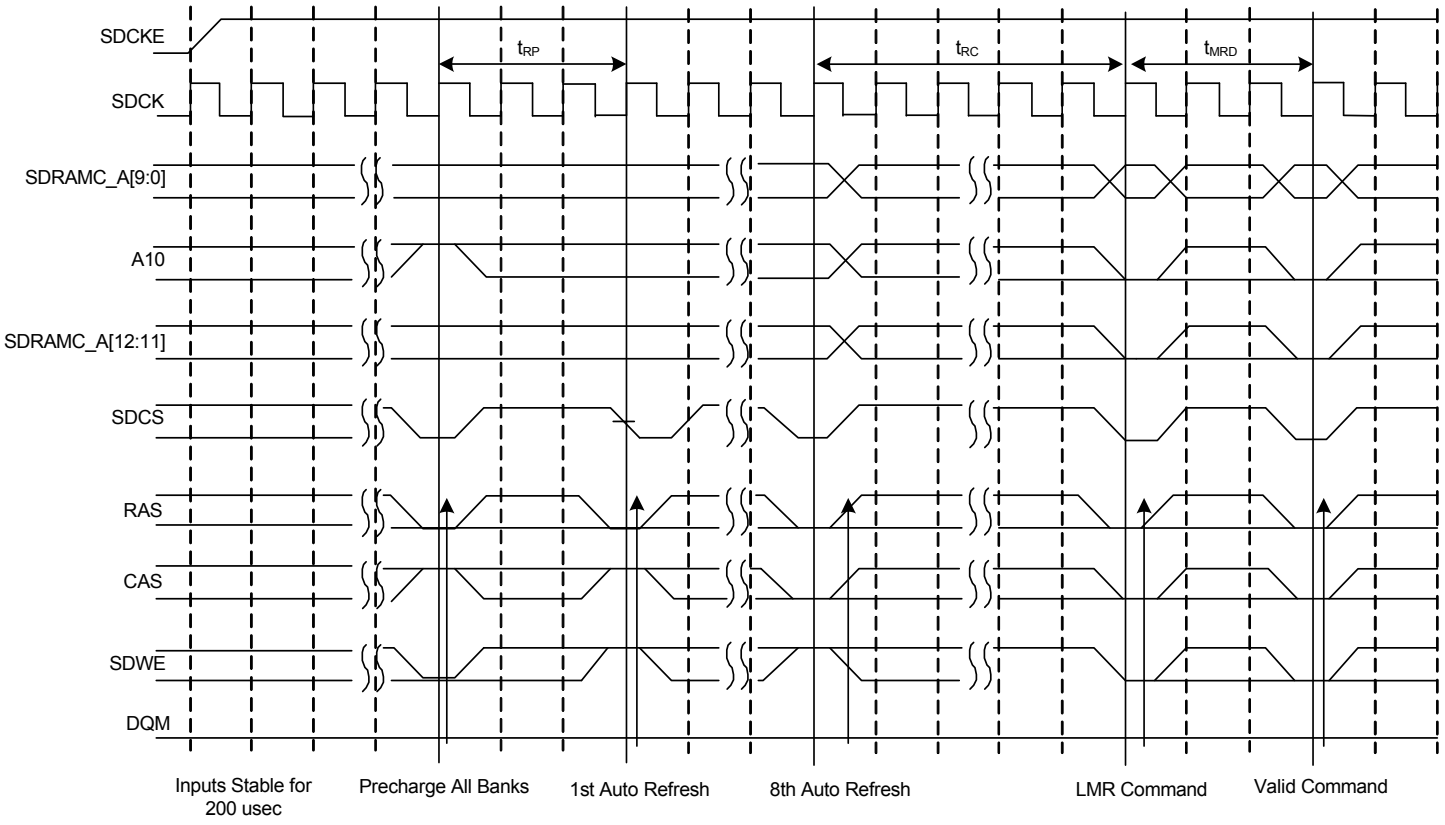
The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

1. SDRAM features must be defined in the CR register by writing the following fields with the desired value: asynchronous timings (TXSR, TRAS, TRCD, TRP, TRC, and TWR), Number of Columns (NC), Number of Rows (NR), Number of Banks (NB), CAS Latency (CAS), and the Data Bus Width (DBW).
2. For mobile SDRAM devices, Temperature Compensated Self Refresh (TCSR), Drive Strength (DS) and Partial Array Self Refresh (PASR) fields must be defined in the Low Power Register (LPR).
3. The Memory Device Type field must be defined in the Memory Device Register (MDR.MD).
4. A No Operation (NOP) command must be issued to the SDRAM devices to start the SDRAM clock. The user must write the value one to the Command Mode field in the SDRAMC Mode Register (MR.MODE) and perform a write access to any SDRAM address.
5. A minimum pause of 200µs is provided to precede any signal toggle.
6. An All Banks Precharge command must be issued to the SDRAM devices. The user must write the value two to the MR.MODE field and perform a write access to any SDRAM address.
7. Eight Auto Refresh commands are provided. The user must write the value four to the MR.MODE field and performs a write access to any SDRAM location eight times.
8. A Load Mode Register command must be issued to program the parameters of the SDRAM devices in its Mode Register, in particular CAS latency, burst type, and burst length. The user must write the value three to the MR.MODE field and perform a write access to the SDRAM. The write address must be chosen so that BA[1:0] are set to zero. See Section 19.8.1 for details about Load Mode Register command.
9. For mobile SDRAM initialization, an Extended Load Mode Register command must be issued to program the SDRAM devices parameters (TCSR, PASR, DS). The user must write the value five to the MR.MODE field and perform a write access to the SDRAM. The write address must be chosen so that BA[1] or BA[0] are equal to one. See Section 19.8.1 for details about Extended Load Mode Register command.
10. The user must go into Normal Mode, writing the value 0 to the MR.MODE field and performing a write access at any location in the SDRAM.
11. Write the refresh rate into the Refresh Timer Count field in the Refresh Timer Register (TR.COUNT). The refresh rate is the delay between two successive refresh cycles. The SDRAM device requires a refresh every 15.625µs or 7.81µs. With a 100MHz fre-

quency, the TR register must be written with the value 1562 (15.625 µs x 100 MHz) or 781 (7.81 µs x 100 MHz).

After initialization, the SDRAM devices are fully functional.

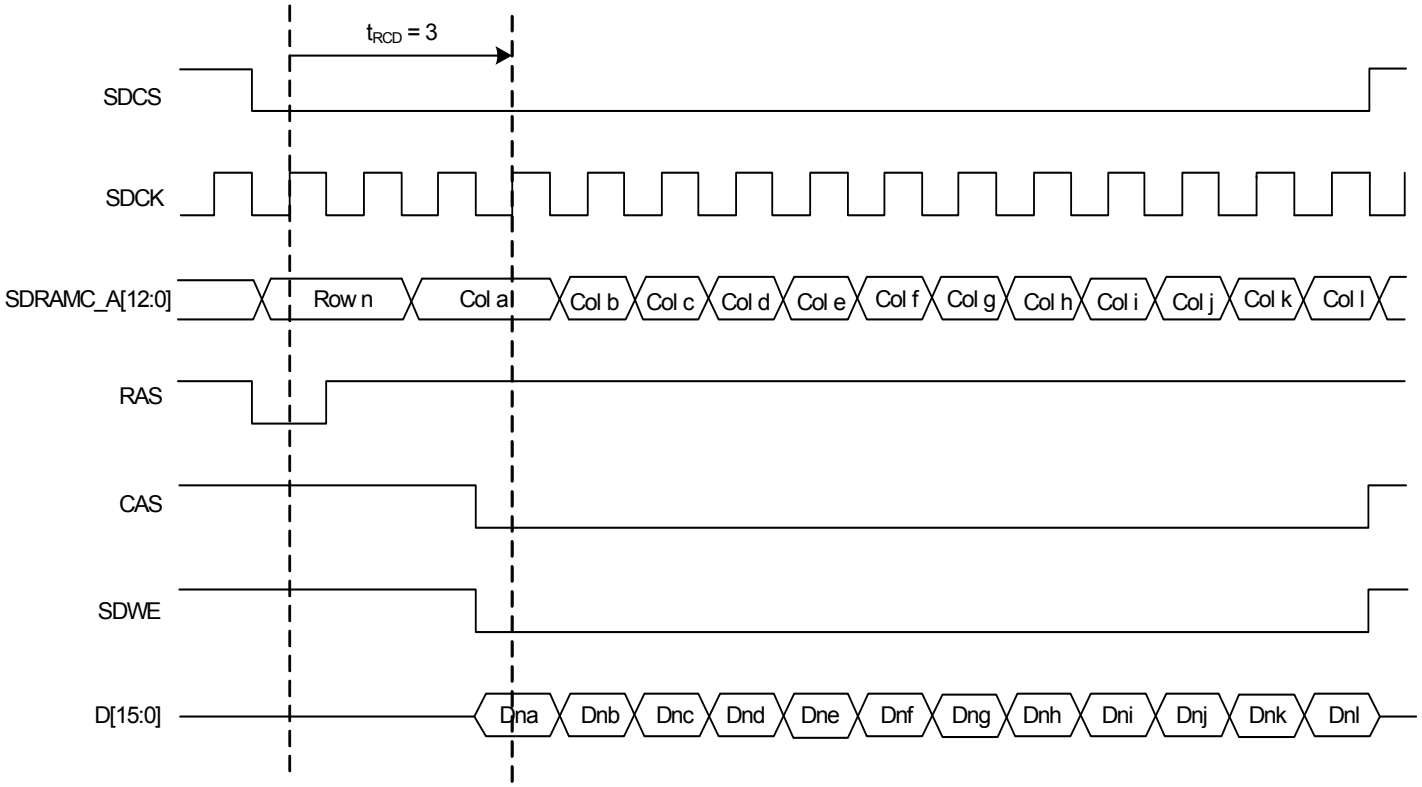
Figure 19-3. SDRAM Device Initialization Sequence



19.7.2 SDRAM Controller Write Cycle

The SDRAMC allows burst access or single access. In both cases, the SDRAMC keeps track of the active row in each bank, thus maximizing performance. To initiate a burst access, the SDRAMC uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write-sequential access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAMC generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge and active (t_{RP}) commands and between active and write (t_{RCD}) commands. For definition of these timing parameters, refer to the Section 19.8.3. This is described in Figure 19-4 on page 354.

Figure 19-4. Write Burst, 16-bit SDRAM Access



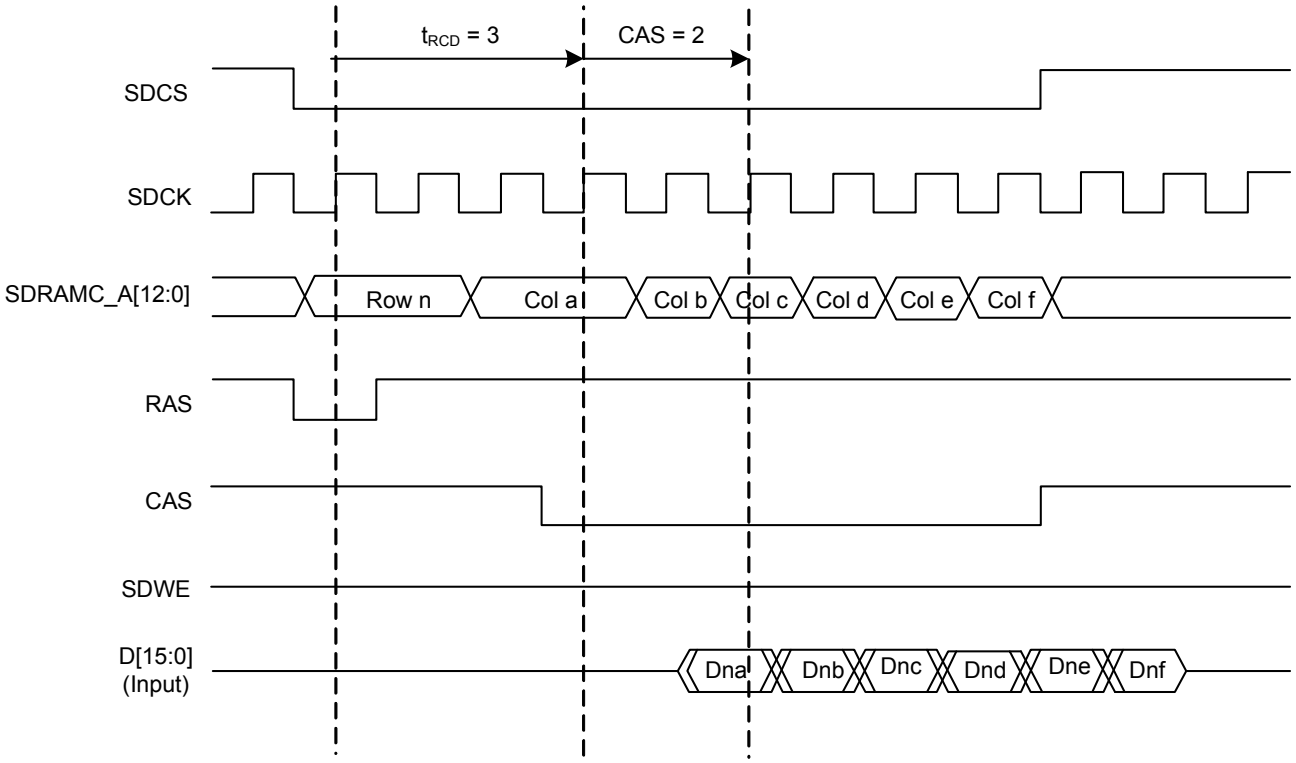
19.7.3 SDRAM Controller Read Cycle

The SDRAMC allows burst access, incremental burst of unspecified length or single access. In all cases, the SDRAMC keeps track of the active row in each bank, thus maximizing performance of the SDRAM. If row and bank addresses do not match the previous row/bank address, then the SDRAMC automatically generates a precharge command, activates the new row and starts the read command. To comply with the SDRAM timing parameters, additional clock cycles on SDCK are inserted between precharge and active (t_{RP}) commands and between active and read (t_{RCD}) commands. These two parameters are set in the CR register of the SDRAMC. After a read command, additional wait states are generated to comply with the CAS latency (one, two, or three clock delays specified in the CR register).

For a single access or an incremented burst of unspecified length, the SDRAMC anticipates the next access. While the last value of the column is returned by the SDRAMC on the bus, the SDRAMC anticipates the read to the next column and thus anticipates the CAS latency. This reduces the effect of the CAS latency on the internal bus.

For burst access of specified length (4, 8, 16 words), access is not anticipated. This case leads to the best performance. If the burst is broken (border, busy mode, etc.), the next access is handled as an incrementing burst of unspecified length.

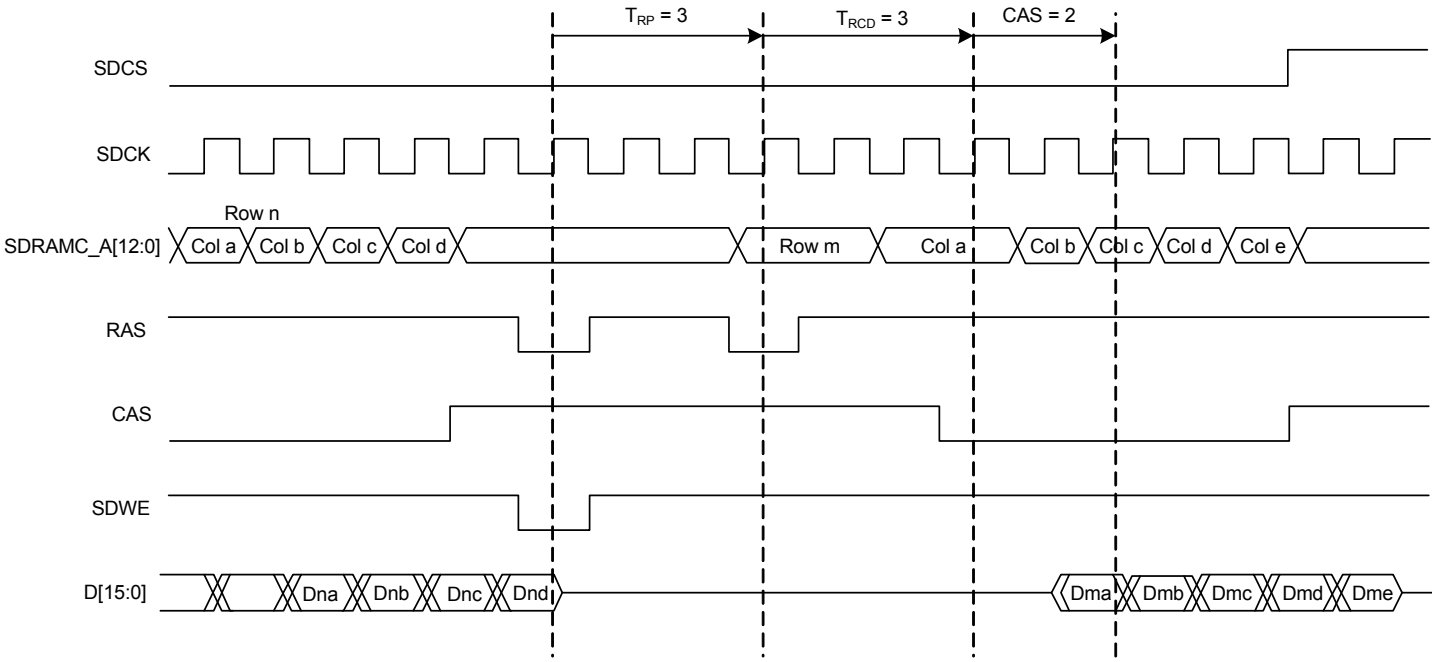
Figure 19-5. Read Burst, 16-bit SDRAM Access



19.7.4 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAMC generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge and active (t_{RP}) commands and between the active and read (t_{RCD}) commands. This is described in Figure 19-6 on page 356.

Figure 19-6. Read Burst with Boundary Row Access



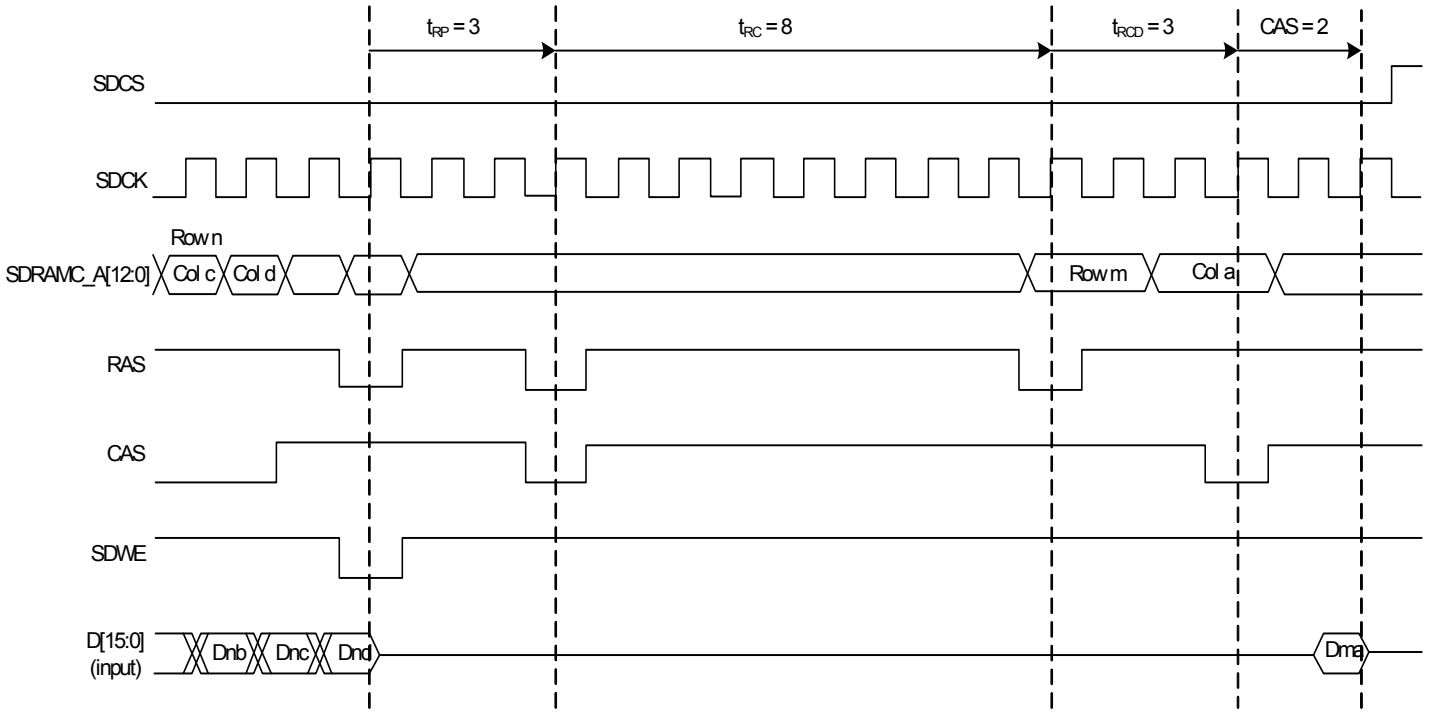
19.7.5 SDRAM Controller Refresh Cycles

An auto refresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each auto refresh automatically. The SDRAMC generates these auto refresh commands periodically. An internal timer is loaded with the value in the Refresh Timer Register (TR) that indicates the number of clock cycles between successive refresh cycles.

A refresh error interrupt is generated when the previous auto refresh command did not perform. In this case a Refresh Error Status bit is set in the Interrupt Status Register (ISR.RES). It is cleared by reading the ISR register.

When the SDRAMC initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM, the slave indicates that the device is busy and the master is held by a wait signal. See Figure 19-7 on page 357.

Figure 19-7. Refresh Cycle Followed by a Read Access



19.7.6 Power Management

Three low power modes are available:

- Self refresh mode: the SDRAM executes its own auto refresh cycles without control of the SDRAMC. Current drained by the SDRAM is very low.
- Power-down mode: auto refresh cycles are controlled by the SDRAMC. Between auto refresh cycles, the SDRAM is in power-down. Current drained in power-down mode is higher than in self refresh mode.
- Deep power-down mode (only available with mobile SDRAM): the SDRAM contents are lost, but the SDRAM does not drain any current.

The SDRAMC activates one low power mode as soon as the SDRAM device is not selected. It is possible to delay the entry in self refresh and power-down mode after the last access by configuring the Timeout field in the Low Power Register (LPR.TIMEOUT).

19.7.6.1 Self refresh mode

This mode is selected by writing the value one to the Low Power Configuration Bits field in the SDRAMC Low Power Register (LPR.LPCB). In self refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto refresh cycles. All the inputs to the SDRAM device become “don’t care” except SDCKE, which remains low. As soon as the SDRAM device is selected, the SDRAMC provides a sequence of commands and exits self refresh mode.

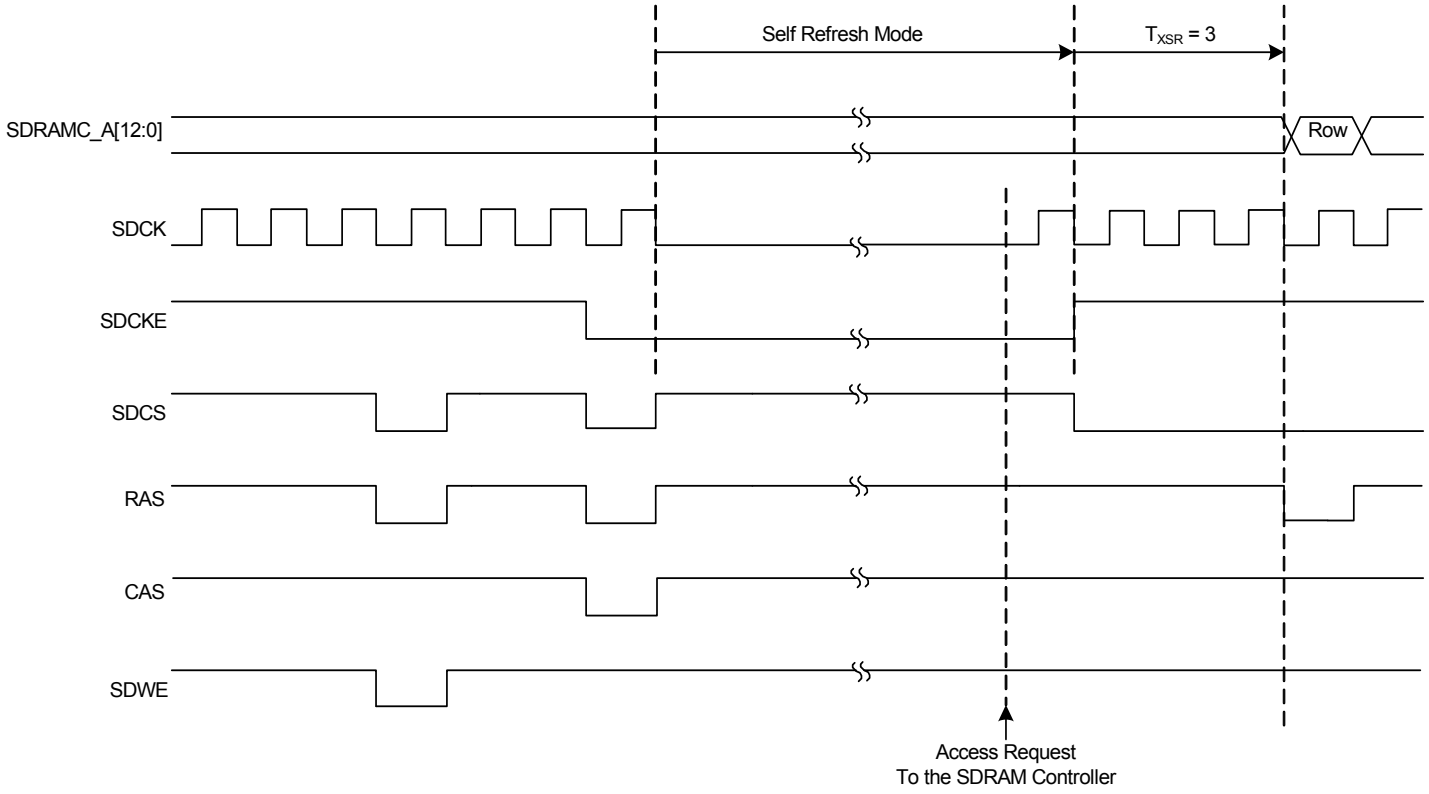
Some low power SDRAMs (e.g., mobile SDRAM) can refresh only one quarter or a half quarter or all banks of the SDRAM array. This feature reduces the self refresh current. To configure this feature, Temperature Compensated Self Refresh (TCSR), Partial Array Self Refresh (PASR)

and Drive Strength (DS) parameters must be set by writing the corresponding fields in the LPR register, and transmitted to the low power SDRAM device during initialization.

After initialization, as soon as the LPR.PASR, LPR.DS, or LPR.TCSR fields are modified and self refresh mode is activated, the SDRAMC issues an Extended Load Mode Register command to the SDRAM and the Extended Mode Register of the SDRAM device is accessed automatically. The PASR/DS/TCSR parameters values are therefore updated before entry into self refresh mode.

The SDRAM device must remain in self refresh mode for a minimum period of t_{RAS} and may remain in self refresh mode for an indefinite period. This is described in Figure 19-8 on page 358.

Figure 19-8. Self Refresh Mode Behavior

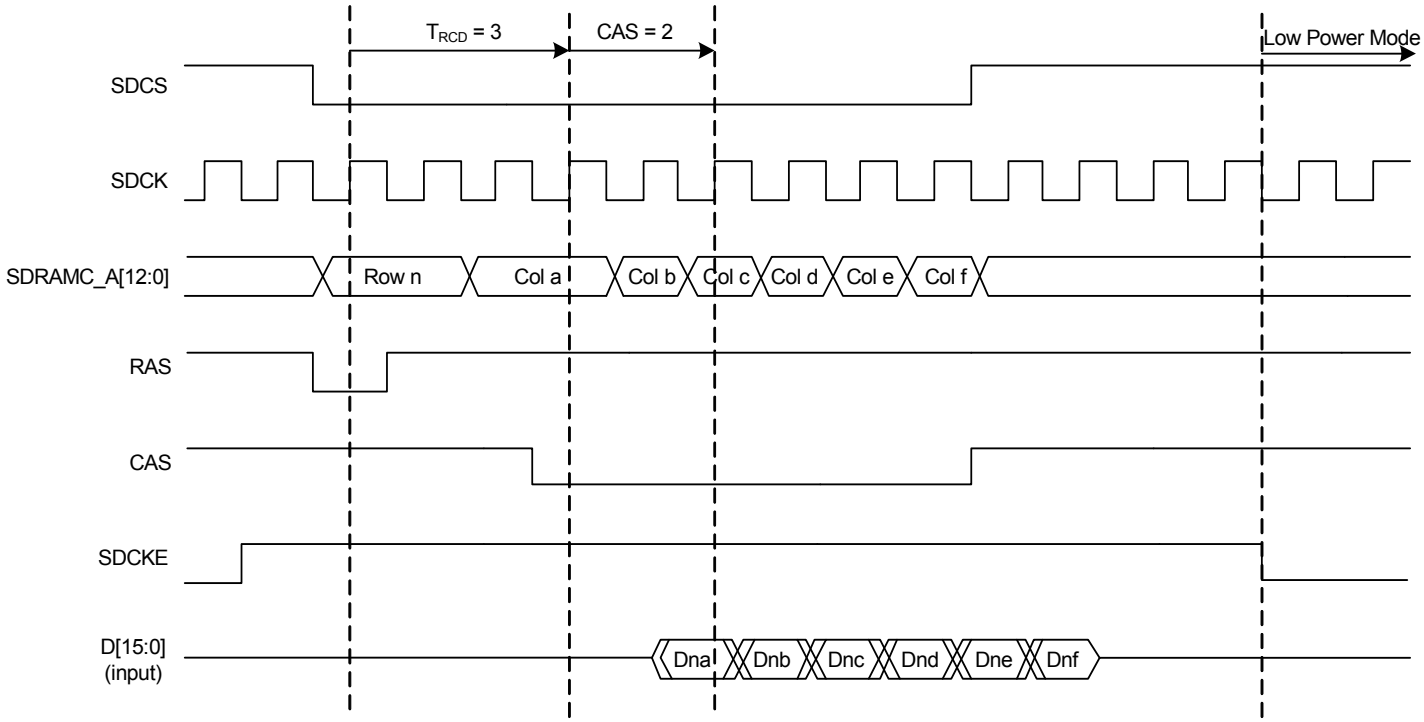


19.7.6.2 Low power mode

This mode is selected by writing the value two to the LPR.LPCB field. Power consumption is greater than in self refresh mode. All the input and output buffers of the SDRAM device are deactivated except SDCKE, which remains low. In contrast to self refresh mode, the SDRAM device cannot remain in low power mode longer than the refresh period (64ms for a whole device refresh operation). As no auto refresh operations are performed by the SDRAM itself, the SDRAMC carries out the refresh operation. The exit procedure is faster than in self refresh mode.

This is described in Figure 19-9 on page 359.

Figure 19-9. Low Power Mode Behavior



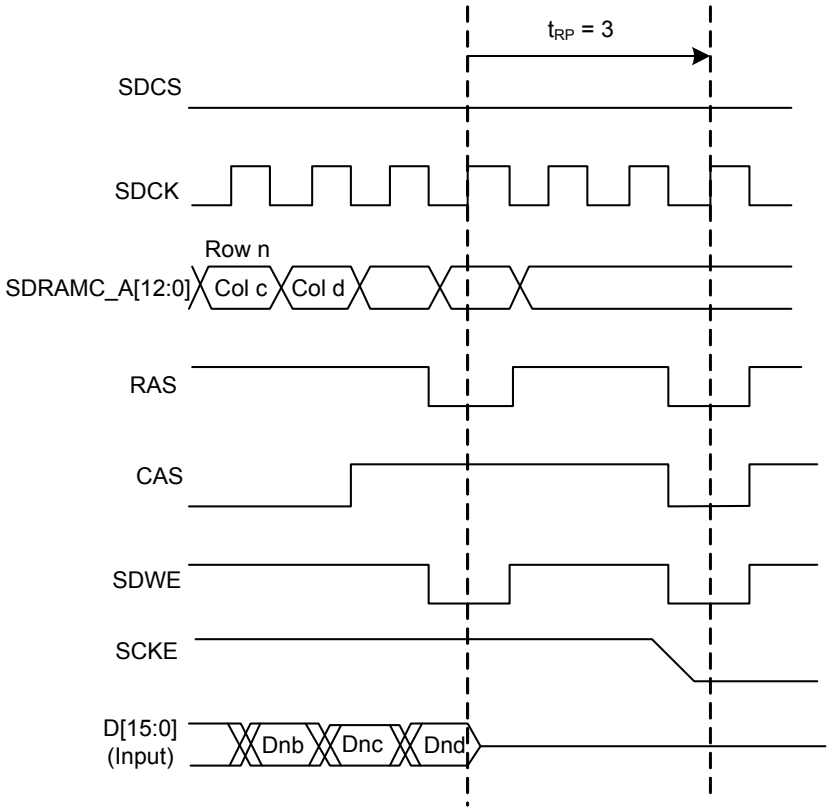
19.7.6.3 Deep power-down mode

This mode is selected by writing the value three to the LPR.LPCB field. When this mode is activated, all internal voltage generators inside the SDRAM are stopped and all data is lost.

When this mode is enabled, the user must not access to the SDRAM until a new initialization sequence is done (See Section 19.7.1).

This is described in Figure 19-10 on page 360.

Figure 19-10. Deep Power-down Mode Behavior



19.8 User Interface

Table 19-5. SDRAMC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Mode Register	MR	Read/Write	0x00000000
0x04	Refresh Timer Register	TR	Read/Write	0x00000000
0x08	Configuration Register	CR	Read/Write	0x852372C0
0x0C	High Speed Register	HSR	Read/Write	0x00000000
0x10	Low Power Register	LPR	Read/Write	0x00000000
0x14	Interrupt Enable Register	IER	Write-only	0x00000000
0x18	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x20	Interrupt Status Register	ISR	Read-only	0x00000000
0x24	Memory Device Register	MDR	Read/Write	0x00000000
0xFC	Version Register	VERSION	Read-only	- (1)

1. The reset values for these fields are device specific. Please refer to the Module Configuration section at the end of this chapter.

19.8.1 Mode Register

Register Name: MR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	MODE		

- **MODE: Command Mode**
  This field defines the command issued by the SDRAMC when the SDRAM device is accessed.

MODE	Description
0	Normal mode. Any access to the SDRAM is decoded normally.
1	The SDRAMC issues a “NOP” command when the SDRAM device is accessed regardless of the cycle.
2	The SDRAMC issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle.
3	The SDRAMC issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. This command will load the CR.CAS field into the SDRAM device Mode Register. All the other parameters of the SDRAM device Mode Register will be set to zero (burst length, burst type, operating mode, write burst mode...).
4	The SDRAMC issues an “Auto Refresh” command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued.
5	The SDRAMC issues an “Extended Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. This command will load the LPR.PASR, LPR.DS, and LPR.TCR fields into the SDRAM device Extended Mode Register. All the other bits of the SDRAM device Extended Mode Register will be set to zero.
6	Deep power-down mode. Enters deep power-down mode.

19.8.2 Refresh Timer Register

Register Name: TR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	COUNT[11:8]			

7	6	5	4	3	2	1	0
COUNT[7:0]							

- **COUNT[11:0]: Refresh Timer Count**

  This 12-bit field is loaded into a timer that generates the refresh pulse. Each time the refresh pulse is generated, a refresh burst is initiated.

  The value to be loaded depends on the SDRAMC clock frequency (CLK_SDRAMC), the refresh rate of the SDRAM device and the refresh burst length where 15.6µs per row is a typical value for a burst of length one.

  To refresh the SDRAM device, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.

19.8.3 Configuration Register

Register Name: CR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x852372C0

31	30	29	28	27	26	25	24
TXSR				TRAS			

23	22	21	20	19	18	17	16
TRCD				TRP			

15	14	13	12	11	10	9	8
TRC				TWR			

7	6	5	4	3	2	1	0
DBW	CAS		NB	NR		NC	

- **TXSR: Exit Self Refresh to Active Delay**
  Reset value is eight cycles.
  This field defines the delay between SCKE set high and an Activate command in number of cycles. Number of cycles is between 0 and 15.
- **TRAS: Active to Precharge Delay**
  Reset value is five cycles.
  This field defines the delay between an Activate command and a Precharge command in number of cycles. Number of cycles is between 0 and 15.
- **TRCD: Row to Column Delay**
  Reset value is two cycles.
  This field defines the delay between an Activate command and a Read/Write command in number of cycles. Number of cycles is between 0 and 15.
- **TRP: Row Precharge Delay**
  Reset value is three cycles.
  This field defines the delay between a Precharge command and another command in number of cycles. Number of cycles is between 0 and 15.
- **TRC: Row Cycle Delay**
  Reset value is seven cycles.
  This field defines the delay between a Refresh and an Activate Command in number of cycles. Number of cycles is between 0 and 15.
- **TWR: Write Recovery Delay**
  Reset value is two cycles.
  This field defines the Write Recovery Time in number of cycles. Number of cycles is between 0 and 15.
- **DBW: Data Bus Width**
  Reset value is 16 bits.
  0: Reserved.
  1: Data bus width is 16 bits.

- **CAS: CAS Latency**
  Reset value is two cycles.
  In the SDRAMC, only a CAS latency of one, two and three cycles is managed.

CAS	CAS Latency (Cycles)
0	Reserved
1	1
2	2
3	3

- **NB: Number of Banks**
  Reset value is two banks.

NB	Number of Banks
0	2
1	4

- **NR: Number of Row Bits**
  Reset value is 11 row bits.

NR	Row Bits
0	11
1	12
2	13
3	Reserved

- **NC: Number of Column Bits**
  Reset value is 8 column bits.

NC	Column Bits
0	8
1	9
2	10
3	11

19.8.4 High Speed Register

Register Name: HSR

Access Type: Read/Write

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DA

- **DA: Decode Cycle Enable**

  A decode cycle can be added on the addresses as soon as a non-sequential access is performed on the HSB bus. The addition of the decode cycle allows the SDRAMC to gain time to access the SDRAM memory.

  1: Decode cycle is enabled.

  0: Decode cycle is disabled.

19.8.5 Low Power Register

Register Name: LPR

Access Type: Read/Write

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	TIMEOUT		DS		TCSR	

7	6	5	4	3	2	1	0
-	PASR			-	-	LPCB	

- **TIMEOUT: Time to Define when Low Power Mode Is Enabled**

TIMEOUT	Time to Define when Low Power Mode Is Enabled
0	The SDRAMC activates the SDRAM low power mode immediately after the end of the last transfer.
1	The SDRAMC activates the SDRAM low power mode 64 clock cycles after the end of the last transfer.
2	The SDRAMC activates the SDRAM low power mode 128 clock cycles after the end of the last transfer.
3	Reserved.

- **DS: Drive Strength (only for low power SDRAM)**
  This field is transmitted to the SDRAM during initialization to select the SDRAM strength of data output. This parameter must be set according to the SDRAM device specification.
  After initialization, as soon as this field is modified and self refresh mode is activated, the Extended Mode Register of the SDRAM device is accessed automatically and its DS parameter value is updated before entry in self refresh mode.
- **TCSR: Temperature Compensated Self Refresh (only for low power SDRAM)**
  This field is transmitted to the SDRAM during initialization to set the refresh interval during self refresh mode depending on the temperature of the low power SDRAM. This parameter must be set according to the SDRAM device specification.
  After initialization, as soon as this field is modified and self refresh mode is activated, the Extended Mode Register of the SDRAM device is accessed automatically and its TCSR parameter value is updated before entry in self refresh mode.
- **PASR: Partial Array Self Refresh (only for low power SDRAM)**
  This field is transmitted to the SDRAM during initialization to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in self refresh mode. This parameter must be set according to the SDRAM device specification.
  After initialization, as soon as this field is modified and self refresh mode is activated, the Extended Mode Register of the SDRAM device is accessed automatically and its PASR parameter value is updated before entry in self refresh mode.

- **LPCB: Low Power Configuration Bits**

LPCB	Low Power Configuration
0	Low power feature is inhibited: no power-down, self refresh or deep power-down command is issued to the SDRAM device.
1	The SDRAMC issues a self refresh command to the SDRAM device, the SDCLK clock is deactivated and the SDCKE signal is set low. The SDRAM device leaves the self refresh mode when accessed and enters it after the access.
2	The SDRAMC issues a power-down command to the SDRAM device after each access, the SDCKE signal is set to low. The SDRAM device leaves the power-down mode when accessed and enters it after the access.
3	The SDRAMC issues a deep power-down command to the SDRAM device. This mode is unique to low-power SDRAM.

19.8.6 Interrupt Enable Register

Register Name: IER

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	RES

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

19.8.7 Interrupt Disable Register

Register Name: IDR

Access Type: Write-only

Offset: 0x18

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

19.8.8 Interrupt Mask Register

Register Name: IMR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	RES

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

19.8.9 Interrupt Status Register

Register Name: ISR

Access Type: Read-only

Offset: 0x20

Reset Value: 0x00000000

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

- **RES: Refresh Error Status**
  This bit is set when a refresh error is detected.
  This bit is cleared when the register is read.

19.8.10 Memory Device Register

Register Name: MDR

Access Type: Read/Write

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	MD	

- **MD: Memory Device Type**

MD	Device Type
0	SDRAM
1	Low power SDRAM
Other	Reserved

19.8.11 Version Register

Register Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION			

7	6	5	4	3	2	1	0
VERSION							

- **Variant: Variant Number**
  Reserved. No functionality associated.

- **Version: Version Number**
  Version number of the module.No functionality associated.

20. Peripheral DMA Controller (PDCA)

Rev: 1.2.3.1

20.1 Features

- **Multiple channels**
- **Generates transfers between memories and peripherals such as USART and SPI**
- **Two address pointers/counters per channel allowing double buffering**
- **Performance monitors to measure average and maximum transfer latency**
- **Optional synchronizing of data transfers with extenal peripheral events**
- **Ring buffer functionality**

20.2 Overview

The Peripheral DMA Controller (PDCA) transfers data between on-chip peripheral modules such as USART, SPI and memories (those memories may be on- and off-chip memories). Using the PDCA avoids CPU intervention for data transfers, improving the performance of the microcontroller. The PDCA can transfer data from memory to a peripheral or from a peripheral to memory.

The PDCA consists of multiple DMA channels. Each channel has:

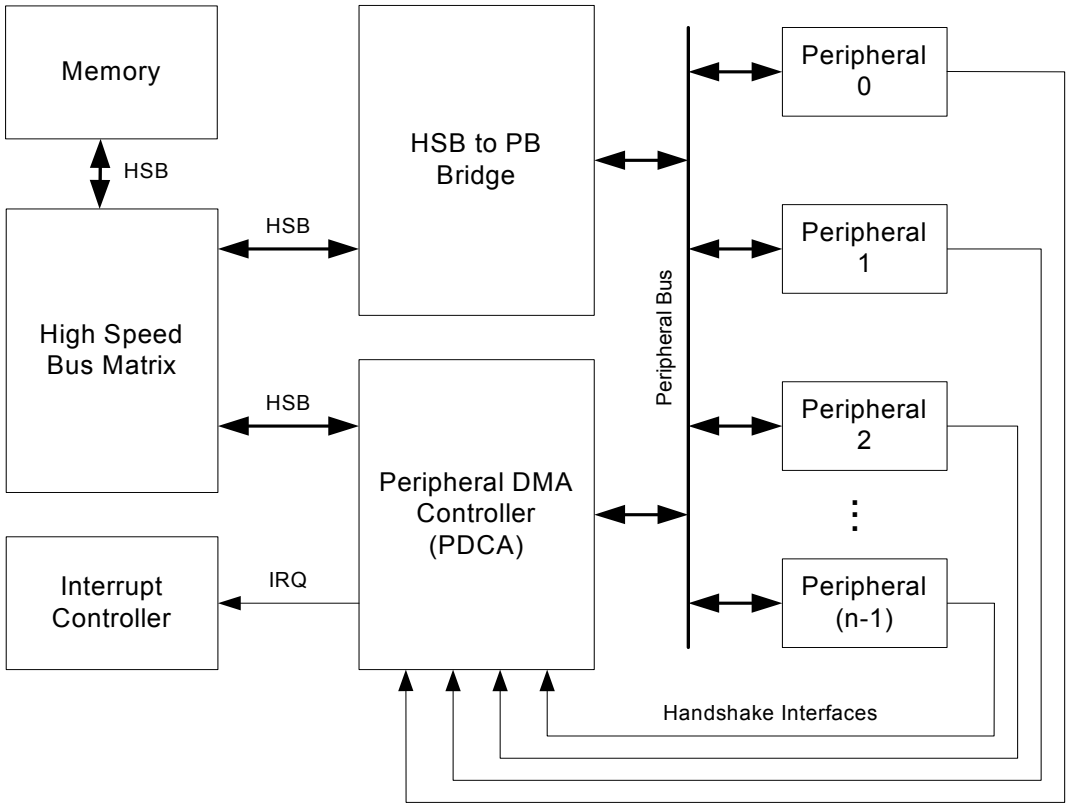
- A Peripheral Select Register
- A 32-bit memory pointer
- A 16-bit transfer counter
- A 32-bit memory pointer reload value
- A 16-bit transfer counter reload value

The PDCA communicates with the peripheral modules over a set of handshake interfaces. The peripheral signals the PDCA when it is ready to receive or transmit data. The PDCA acknowledges the request when the transmission has started.

When a transmit buffer is empty or a receive buffer is full, an optional interrupt request can be generated.

20.3 Block Diagram

Figure 20-1. PDCA Block Diagram



20.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

20.4.1 Power Management

If the CPU enters a sleep mode that disables the PDCA clocks, the PDCA will stop functioning and resume operation after the system wakes up from sleep mode.

20.4.2 Clocks

The PDCA has two bus clocks connected: One High Speed Bus clock (CLK_PDCA_HSB) and one Peripheral Bus clock (CLK_PDCA_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by writing to the Power Manager. It is recommended to disable the PDCA before disabling the clocks, to avoid freezing the PDCA in an undefined state.

20.4.3 Interrupts

The PDCA interrupt request lines are connected to the interrupt controller. Using the PDCA interrupts requires the interrupt controller to be programmed first.

20.4.4 Peripheral Events

The PDCA peripheral events are connected via the Peripheral Event System. Refer to the Peripheral Event System chapter for details.

20.5 Functional Description

20.5.1 Basic Operation

The PDCA consists of multiple independent PDCA channels, each capable of handling DMA requests in parallel. Each PDCA channels contains a set of configuration registers which must be configured to start a DMA transfer.

In this section the steps necessary to configure one PDCA channel is outlined.

The peripheral to transfer data to or from must be configured correctly in the Peripheral Select Register (PSR). This is performed by writing the Peripheral Identity (PID) value for the corresponding peripheral to the PID field in the PSR register. The PID also encodes the transfer direction, i.e. memory to peripheral or peripheral to memory. See Section 20.5.6.

The transfer size must be written to the Transfer Size field in the Mode Register (MR.SIZE). The size must match the data size produced or consumed by the selected peripheral. See Section 20.5.7.

The memory address to transfer to or from, depending on the PSR, must be written to the Memory Address Register (MAR). For each transfer the memory address is increased by either a one, two or four, depending on the size set in MR. See Section 20.5.2.

The number of data items to transfer is written to the TCR register. If the PDCA channel is enabled, a transfer will start immediately after writing a non-zero value to TCR or the reload version of TCR, TCRR. After each transfer the TCR value is decreased by one. Both MAR and TCR can be read while the PDCA channel is active to monitor the DMA progress. See Section 20.5.3.

The channel must be enabled for a transfer to start. A channel is enable by writing a one to the EN bit in the Control Register (CR).

20.5.2 Memory Pointer

Each channel has a 32-bit Memory Address Register (MAR). This register holds the memory address for the next transfer to be performed. The register is automatically updated after each transfer. The address will be increased by either one, two or four depending on the size of the DMA transfer (byte, halfword or word). The MAR can be read at any time during transfer.

20.5.3 Transfer Counter

Each channel has a 16-bit Transfer Counter Register (TCR). This register must be programmed with the number of transfers to be performed. The TCR register should contain the number of data items to be transferred independently of the transfer size. The TCR can be read at any time during transfer to see the number of remaining transfers.

20.5.4 Reload Registers

Both the MAR and the TCR have a reload register, respectively Memory Address Reload Register (MARR) and Transfer Counter Reload Register (TCRR). These registers provide the possibility for the PDCA to work on two memory buffers for each channel. When one buffer has completed, MAR and TCR will be reloaded with the values in MARR and TCRR. The reload logic is always enabled and will trigger if the TCR reaches zero while TCRR holds a non-zero value. After reload, the MARR and TCRR registers are cleared.

If TCR is zero when writing to TCRR, the TCR and MAR are automatically updated with the value written in TCRR and MARR.

20.5.5 Ring Buffer

When Ring Buffer mode is enabled the TCRR and MARR registers will not be cleared when TCR and MAR registers reload. This allows the PDCA to read or write to the same memory region over and over again until the transfer is actively stopped by the user. Ring Buffer mode is enabled by writing a one to the Ring Buffer bit in the Mode Register (MR.RING).

20.5.6 Peripheral Selection

The Peripheral Select Register (PSR) decides which peripheral should be connected to the PDCA channel. A peripheral is selected by writing the corresponding Peripheral Identity (PID) to the PID field in the PSR register. Writing the PID will both select the direction of the transfer (memory to peripheral or peripheral to memory), which handshake interface to use, and the address of the peripheral holding register. Refer to the Peripheral Identity (PID) table in the Module Configuration section for the peripheral PID values.

20.5.7 Transfer Size

The transfer size can be set individually for each channel to be either byte, halfword or word (8-bit, 16-bit or 32-bit respectively). Transfer size is set by writing the desired value to the Transfer Size field in the Mode Register (MR.SIZE).

When the PDCA moves data between peripherals and memory, data is automatically sized and aligned. When memory is accessed, the size specified in MR.SIZE and system alignment is used. When a peripheral register is accessed the data to be transferred is converted to a word where bit n in the data corresponds to bit n in the peripheral register. If the transfer size is byte or halfword, bits greater than 8 and16 respectively are set to zero.

Refer to the Module Configuration section for information regarding what peripheral registers are used for the different peripherals and then to the peripheral specific chapter for information about the size option available for the different registers.

20.5.8 Enabling and Disabling

Each DMA channel is enabled by writing a one to the Transfer Enable bit in the Control Register (CR.TEN) and disabled by writing a one to the Transfer Disable bit (CR.TDIS). The current status can be read from the Status Register (SR).

While the PDCA channel is enabled all DMA request will be handled as long the TCR and TCRR is not zero.

20.5.9 Interrupts

Interrupts can be enabled by writing a one to the corresponding bit in the Interrupt Enable Register (IER) and disabled by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The Interrupt Mask Register (IMR) can be read to see whether an interrupt is enabled or not. The current status of an interrupt source can be read through the Interrupt Status Register (ISR).

The PDCA has three interrupt sources:

- Reload Counter Zero - The TCRR register is zero.
- Transfer Finished - Both the TCR and TCRR registers are zero.
- Transfer Error - An error has occurred in accessing memory.

20.5.10 Priority

If more than one PDCA channel is requesting transfer at a given time, the PDCA channels are prioritized by their channel number. Channels with lower numbers have priority over channels with higher numbers, giving channel zero the highest priority.

20.5.11 Error Handling

If the Memory Address Register (MAR) is set to point to an invalid location in memory, an error will occur when the PDCA tries to perform a transfer. When an error occurs, the Transfer Error bit in the Interrupt Status Register (ISR.TERR) will be set and the DMA channel that caused the error will be stopped. In order to restart the channel, the user must program the Memory Address Register to a valid address and then write a one to the Error Clear bit in the Control Register (CR.ECLR). If the Transfer Error interrupt is enabled, an interrupt request will be generated when an transfer error occurs.

20.5.12 Peripheral Event Trigger

Peripheral events can be used to trigger PDCA channel transfers. Peripheral Event synchronizations are enabled by writing a one to the Event Trigger bit in the Mode Register (MR.ETRIG). When set, all DMA requests will be blocked until an peripheral event is received. For each peripheral event received, only one data item is transferred. If no DMA requests are pending when a peripheral event is received, the PDCA will start a transfer as soon as a peripheral event is detected. If multiple events arrive while the PDCA channel is busy transferring data, an overflow condition will be signaled in the Peripheral Event System. Refer to the Peripheral Event System chapter for more information.

20.6 Performance Monitors

Up tp two performance monitors allow the user to measure the activity and stall cycles for PDCA transfers. To monitor a PDCA channel, the corresponding channel number must be written to one of the MONnCH fields in the Performance Control Register (PCONTROL) and a one must be written to the corresponding CHnEN bit in the same register.

Due to performance monitor hardware resource sharing, the two monitor channels should NOT be programmed to monitor the same PDCA channel. This may result in UNDEFINED performance monitor behavior.

20.6.1 Measuring mechanisms

Three different parameters can be measured by each channel:

- The number of data transfer cycles since last channel reset, both for read and write
- The number of stall cycles since last channel reset, both for read and write
- The maximum latency since last channel reset, both for read and write

These measurements can be extracted by software and used to generate indicators for bus latency, bus load, and maximum bus latency.

Each of the counters has a fixed width, and may therefore overflow. When an overflow is encountered in either the Performance Channel Data Read/Write Cycle registers (PRDATAn and PWDATAn) or the Performance Channel Read/Write Stall Cycles registers (PRSTALLn and PWSTALLn) of a channel, all registers in the channel are reset. This behavior is altered if the Channel Overflow Freeze bit is one in the Performance Control register (PCONTROL.CHnOVF). If this bit is one, the channel registers are frozen when either DATA or STALL reaches its maximum value. This simplifies one-shot readout of the counter values.

The registers can also be manually reset by writing a one to the Channel Reset bit in the PCONTROL register (PCONTROL.CHnRES). The Performance Channel Read/Write Latency registers (PRLATn and PWLATn) are saturating when their maximum count value is reached. The PRLATn and PWLATn registers are reset only by writing a one to the CHnRES in PCONTROL.

A counter must manually be enabled by writing a one to the Channel Enable bit in the Performance Control Register (PCONTROL.CHnEN).

20.7 User Interface

20.7.1 Memory Map Overview

Table 20-1. PDCA Register Memory Map

Address Range	Contents
0x000 - 0x03F	DMA channel 0 configuration registers
0x040 - 0x07F	DMA channel 1 configuration registers
...	...
(0x000 - 0x03F)+m*0x040	DMA channel m configuration registers
0x800-0x830	Performance Monitor registers
0x834	Version register

The channels are mapped as shown in Table 20-1. Each channel has a set of configuration registers, shown in Table 20-2, where *n* is the channel number.

20.7.2 Channel Memory Map

Table 20-2. PDCA Channel Configuration Registers

Offset	Register	Register Name	Access	Reset
0x000 + n*0x040	Memory Address Register	MAR	Read/Write	0x00000000
0x004 + n*0x040	Peripheral Select Register	PSR	Read/Write	- (1)
0x008 + n*0x040	Transfer Counter Register	TCR	Read/Write	0x00000000
0x00C + n*0x040	Memory Address Reload Register	MARR	Read/Write	0x00000000
0x010 + n*0x040	Transfer Counter Reload Register	TCRR	Read/Write	0x00000000
0x014 + n*0x040	Control Register	CR	Write-only	0x00000000
0x018 + n*0x040	Mode Register	MR	Read/Write	0x00000000
0x01C + n*0x040	Status Register	SR	Read-only	0x00000000
0x020 + n*0x040	Interrupt Enable Register	IER	Write-only	0x00000000
0x024 + n*0x040	Interrupt Disable Register	IDR	Write-only	0x00000000
0x028 + n*0x040	Interrupt Mask Register	IMR	Read-only	0x00000000
0x02C + n*0x040	Interrupt Status Register	ISR	Read-only	0x00000000

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

20.7.3 Performance Monitor Memory Map

Table 20-3. PDCA Performance Monitor Registers[1]

Offset	Register	Register Name	Access	Reset
0x800	Performance Control Register	PCONTROL	Read/Write	0x00000000
0x804	Channel0 Read Data Cycles	PRDATA0	Read-only	0x00000000
0x808	Channel0 Read Stall Cycles	PRSTALL0	Read-only	0x00000000
0x80C	Channel0 Read Max Latency	PRLAT0	Read-only	0x00000000
0x810	Channel0 Write Data Cycles	PWDATA0	Read-only	0x00000000
0x814	Channel0 Write Stall Cycles	PWSTALL0	Read-only	0x00000000
0x818	Channel0 Write Max Latency	PWLAT0	Read-only	0x00000000
0x81C	Channel1 Read Data Cycles	PRDATA1	Read-only	0x00000000
0x820	Channel1 Read Stall Cycles	PRSTALL1	Read-only	0x00000000
0x824	Channel1 Read Max Latency	PRLAT1	Read-only	0x00000000
0x828	Channel1 Write Data Cycles	PWDATA1	Read-only	0x00000000
0x82C	Channel1 Write Stall Cycles	PWSTALL1	Read-only	0x00000000
0x830	Channel1 Write Max Latency	PWLAT1	Read-only	0x00000000

Note: 1. The number of performance monitors is device specific. If the device has only one performance monitor, the Channel1 registers are not available. Please refer to the Module Configuration section at the end of this chapter for the number of performance monitors on this device.

20.7.4 Version Register Memory Map

Table 20-4. PDCA Version Register Memory Map

Offset	Register	Register Name	Access	Reset
0x834	Version Register	VERSION	Read-only	-[1]

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

20.7.5 Memory Address Register

Name: MAR

Access Type: Read/Write

Offset: 0x000 + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MADDR[31:24]							

23	22	21	20	19	18	17	16
MADDR[23:16]							

15	14	13	12	11	10	9	8
MADDR[15:8]							

7	6	5	4	3	2	1	0
MADDR[7:0]							

- **MADDR: Memory Address**
  Address of memory buffer. MADDR should be programmed to point to the start of the memory buffer when configuring the PDCA. During transfer, MADDR will point to the next memory location to be read/written.

20.7.6 Peripheral Select Register

Name: PSR

Access Type: Read/Write

Offset: 0x004 + n*0x040

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
PID							

- **PID: Peripheral Identifier**

  The Peripheral Identifier selects which peripheral should be connected to the DMA channel. Writing a PID will select both which handshake interface to use, the direction of the transfer and also the address of the Receive/Transfer Holding Register for the peripheral. See the Module Configuration section of PDCA for details. The width of the PID field is device specific and dependent on the number of peripheral modules in the device.

20.7.7 Transfer Counter Register

Name: TCR

Access Type: Read/Write

Offset: 0x008 + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
TCV[15:8]							

7	6	5	4	3	2	1	0
TCV[7:0]							

- **TCV: Transfer Counter Value**
  Number of data items to be transferred by the PDCA. TCV must be programmed with the total number of transfers to be made. During transfer, TCV contains the number of remaining transfers to be done.

20.7.8 Memory Address Reload Register

Name: MARR

Access Type: Read/Write

Offset: 0x00C + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MARV[31:24]							

23	22	21	20	19	18	17	16
MARV[23:16]							

15	14	13	12	11	10	9	8
MARV[15:8]							

7	6	5	4	3	2	1	0
MARV[7:0]							

- **MARV: Memory Address Reload Value**
  Reload Value for the MAR register. This value will be loaded into MAR when TCR reaches zero if the TCRR register has a non-zero value.

20.7.9 Transfer Counter Reload Register

Name: TCRR

Access Type: Read/Write

Offset: 0x010 + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
TCRV[15:8]							

7	6	5	4	3	2	1	0
TCRV[7:0]							

- **TCRV: Transfer Counter Reload Value**
  Reload value for the TCR register. When TCR reaches zero, it will be reloaded with TCRV if TCRV has a positive value. If TCRV is zero, no more transfers will be performed for the channel. When TCR is reloaded, the TCRR register is cleared.

20.7.10 Control Register

Name: CR

Access Type: Write-only

Offset: 0x014 + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	ECLR

7	6	5	4	3	2	1	0
-	-	-	-	-	-	TDIS	TEN

- **ECLR: Transfer Error Clear**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Transfer Error bit in the Status Register (SR.TERR). Clearing the SR.TERR bit will allow the channel to transmit data. The memory address must first be set to point to a valid location.
- **TDIS: Transfer Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will disable transfer for the DMA channel.
- **TEN: Transfer Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will enable transfer for the DMA channel.

20.7.11 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x018 + n*0x040

Reset Value: 0x00000000

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

- **RING: Ring Buffer**
  0:The Ring buffer functionality is disabled.
  1:The Ring buffer functionality is enabled. When enabled, the reload registers, MARR and TCRR will not be cleared after reload.
- **ETRIG: Event Trigger**
  0:Start transfer when the peripheral selected in Peripheral Select Register (PSR) requests a transfer.
  1:Start transfer only when or after a peripheral event is received.
- **SIZE: Size of Transfer**

Table 20-5. Size of Transfer

SIZE	Size of Transfer
0	Byte
1	Halfword
2	Word
3	Reserved

20.7.12 Status Register

Name: SR

Access Type: Read-only

Offset: 0x01C + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	TEN

- **TEN: Transfer Enabled**

  This bit is cleared when the TDIS bit in CR is written to one.

  This bit is set when the TEN bit in CR is written to one.

  0: Transfer is disabled for the DMA channel.

  1: Transfer is enabled for the DMA channel.

20.7.13 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x020 + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	TERR	TRC	RCZ

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

20.7.14 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x024 + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	TERR	TRC	RCZ

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

20.7.15 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x028 + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	TERR	TRC	RCZ

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

20.7.16 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x02C + n*0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	TERR	TRC	RCZ

- **TERR: Transfer Error**
  This bit is cleared when no transfer errors have occurred since the last write to CR.ECLR.
  This bit is set when one or more transfer errors has occurred since reset or the last write to CR.ECLR.
- **TRC: Transfer Complete**
  This bit is cleared when the TCR and/or the TCRR holds a non-zero value.
  This bit is set when both the TCR and the TCRR are zero.
- **RCZ: Reload Counter Zero**
  This bit is cleared when the TCRR holds a non-zero value.
  This bit is set when TCRR is zero.

20.7.17 Performance Control Register

Name: PCONTROL

Access Type: Read/Write

Offset: 0x800

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	MON1CH					

23	22	21	20	19	18	17	16
-	-	MON0CH					

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CH1RES	CH0RES

7	6	5	4	3	2	1	0
-	-	CH1OF	CH0OF	-	-	CH1EN	CH0EN

- **MON1CH: Performance Monitor Channel 1**
- **MON0CH: Performance Monitor Channel 0**

  The PDCA channel number to monitor with counter n

  Due to performance monitor hardware resource sharing, the two performance monitor channels should NOT be programmed to monitor the same PDCA channel. This may result in UNDEFINED monitor behavior.
- **CH1RES: Performance Channel 1 Counter Reset**

  Writing a zero to this bit has no effect.

  Writing a one to this bit will reset the counter in performance channel 1.

  This bit always reads as zero.
- **CH0RES: Performance Channel 0 Counter Reset**

  Writing a zero to this bit has no effect.

  Writing a one to this bit will reset the counter in performance channel 0.

  This bit always reads as zero.
- **CH1OF: Performance Channel 1 Overflow Freeze**

  0: The performance channel registers are reset if DATA or STALL overflows.

  1: All performance channel registers are frozen just before DATA or STALL overflows.
- **CH1OF: Performance Channel 0 Overflow Freeze**

  0: The performance channel registers are reset if DATA or STALL overflows.

  1: All performance channel registers are frozen just before DATA or STALL overflows.
- **CH1EN: Performance Channel 1 Enable**

  0: Performance channel 1 is disabled.

  1: Performance channel 1 is enabled.
- **CH0EN: Performance Channel 0 Enable**

  0: Performance channel 0 is disabled.

  1: Performance channel 0 is enabled.

20.7.18 Performance Channel 0 Read Data Cycles

Name: PRDATA0

Access Type: Read-only

Offset: 0x804

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

20.7.19 Performance Channel 0 Read Stall Cycles

Name: PRSTALL0

Access Type: Read-only

Offset: 0x808

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL: Stall Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

20.7.20 Performance Channel 0 Read Max Latency

Name: PRLAT0

Access Type: Read/Write

Offset: 0x80C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock
  This counter is saturating. The register is reset only when PCONTROL.CH0RES is written to one.

20.7.21 Performance Channel 0 Write Data Cycles

Name: PWDATA0

Access Type: Read-only

Offset: 0x810

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

20.7.22 Performance Channel 0 Write Stall Cycles

Name: PWSTALL0

Access Type: Read-only

Offset: 0x814

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL: Stall Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

20.7.23 Performance Channel 0 Write Max Latency

Name: PWLAT0

Access Type: Read/Write

Offset: 0x818

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock
  This counter is saturating. The register is reset only when PCONTROL.CH0RES is written to one.

20.7.24 Performance Channel 1 Read Data Cycles

Name: PRDATA1

Access Type: Read-only

Offset: 0x81C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

20.7.25 Performance Channel 1 Read Stall Cycles

Name: PRSTALL1

Access Type: Read-only

Offset: 0x820

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL: Stall Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

20.7.26 Performance Channel 1 Read Max Latency

Name: PLATR1

Access Type: Read/Write

Offset: 0x824

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock
  This counter is saturating. The register is reset only when PCONTROL.CH1RES is written to one.

20.7.27 Performance Channel 1 Write Data Cycles

Name: PWDATA1

Access Type: Read-only

Offset: 0x828

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DATA[31:24]							

23	22	21	20	19	18	17	16
DATA[23:16]							

15	14	13	12	11	10	9	8
DATA[15:8]							

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

20.7.28 Performance Channel 1 Write Stall Cycles

Name: PWSTALL1

Access Type: Read-only

Offset: 0x82C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

7	6	5	4	3	2	1	0
STALL[7:0]							

- **STALL: Stall Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

20.7.29 Performance Channel 1 Write Max Latency

Name: PWLAT1

Access Type: Read/Write

Offset: 0x830

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
LAT[15:8]							

7	6	5	4	3	2	1	0
LAT[7:0]							

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock
  This counter is saturating. The register is reset only when PCONTROL.CH1RES is written to one.

20.7.30 PDCA Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x834

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

20.8 Module Configuration

The specific configuration for the PDCA instance is listed in the following tables.

Table 20-6. PDCA Configuration

Features	PDCA
Number of channels	16
Number of performance monitors	1

Table 20-7. Module Clock Name

Module name	Clock name	Description
PDCA	CLK_PDCA_HSB	HSB clock
	CLK_PDCA_PB	Peripheral Bus clock from the PBC clock domain

Table 20-8. Register Reset Values

Register	Reset Value
PSRn	n
VERSION	0x123

The table below defines the valid Peripheral Identifiers (PIDs). The direction is specified as observed from the memory, so RX means transfers from peripheral to memory and TX means from memory to peripheral.

Table 20-9. Peripheral Identity Values

PID	Direction	Peripheral Instance	Peripheral Register
0	RX	ADCIFA	LCV0
1	RX	ADCIFA	LCV1
2	RX	USART0	RHR
3	RX	USART1	RHR
4	RX	USART2	RHR
5	RX	USART3	RHR
6	RX	TWIM0	RHR
7	RX	TWIM1	RHR
8	RX	TWIS0	RHR
9	RX	TWIS1	RHR
10	RX	SPI0	RDR
11	RX	SPI1	RDR
12	RX	AW	RHR
13	TX	USART0	THR
14	TX	USART1	THR
15	TX	USART2	THR
16	TX	USART3	THR

Table 20-9. Peripheral Identity Values

PID	Direction	Peripheral Instance	Peripheral Register
17	TX	TWIM0	THR
18	TX	TWIM1	THR
19	TX	TWIS0	THR
20	TX	TWIS1	THR
21	TX	SPI0	TDR
22	TX	SPI1	TDR
23	TX	DACIFB0	DR0
24	TX	DACIFB0	DR1
25	TX	DACIFB1	DR0
26	TX	DACIFB1	DR1
27	TX	PWM	PWM PDCA register
28	TX	AW	THR
31	RX	USART4	RHR
32	RX	TWIM2	RHR
33	RX	TWIS2	RHR
34	TX	USART4	THR
35	TX	TWIM2	THR
36	TX	TWIS2	THR
[44:37]	RX	IISC	RHR
[52:45]	TX	IISC	THR

21. Memory DMA Controller (MDMA)

Rev 1.0.1.1

21.1 Features

- **1-4 DMA channels, depending on implementation**
- **Chained (descriptor-list controlled) or unchained (single) transfers**
- **Descriptor read and writeback support**
- **Descriptors are placed in circular buffers of programmable length**
- **Programmable fixed or round-robin priority between channels**
- **Programmable burst length (1, 4, 8, or 16-beat)**
- **Byte/halfword/word transfers**
- **Optional endianess-conversion on transferred data**
- **Interrupt on transfer complete**

21.2 Overview

The purpose of the MDMA is to perform memory-to-memory transfers. For peripheral-to-memory transfers, the Peripheral DMA Controller should be used instead.

The MDMA has two HSB master interfaces. One interface is dedicated to reading data while the other is dedicated to writing. The MDMA is configured through a Peripheral Bus (PB) interface.

A DMA transfer on a channel can be started manually by writing the MDMA configuration registers for that channel. This transfer mode is referred to as Single Transfer Mode.

MDMA channels can also be controlled by a descriptor list in memory. The descriptor list contains all information needed to control a transfer. Once a transfer has been completed, the MDMA automatically reads the next descriptor, and if this descriptor is valid, starts the next DMA transfer. This transfer mode is referred to Descriptor Mode.

21.3 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

21.3.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the MDMA, the MDMA will stop functioning and resume operation after the system wakes up from sleep mode.

21.3.2 Clocks

The MDMA has two bus clocks connected: One High Speed Bus clock (CLK_MDMA_HSB) and one Peripheral Bus clock (CLK_MDMA_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled in the Power Manager. The user has to ensure that CLK_MDMA_HSB is not turned off while performing MDMA transfers. Failing to do so may deadlock the HSB.

21.3.3 Interrupts

The MDMA interrupt request lines are connected to the interrupt controller. Using the MDMA interrupts requires the interrupt controller to be programmed first.

21.3.4 Debug Operation

When an external debugger forces the CPU into debug mode, the MDMA continues normal operation. If the MDMA is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

21.4 Functional Description

21.4.1 Bus Interfaces

The MDMA has three bus interfaces, two High-Speed Bus interfaces for data and descriptor transfer, and one Peripheral Bus interface for writing control information to and reading status information from the controller.

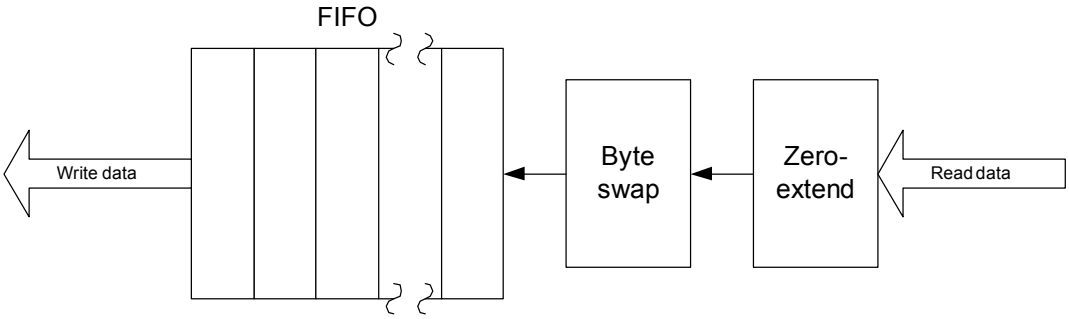
21.4.2 Transferring Data

Once a channel (x) is selected by the arbiter, data of the size given by the SIZE field in the Channel Control Register (CCRx.SIZE) will be transferred from consecutive addresses starting as specified in the Read Address Register (RARx) to consecutive addresses starting as specified in the Write Address Register (WARx). The number of data to be transferred is given by the Transfer Count field (CCRx.TCNT). The MDMA will try to transfer data in bursts with burst length given by CCRx.BURST. The MDMA is free to use bursts of shorter length if this is required by the bus semantics or if TCNT is not perfectly divisible by BURST.

During transfers, TCNT is continuously decremented until it reaches zero, indicating that the transfer has completed. RARx and WARx are not changed by hardware during transfers.

Data read from the bus is put into a FIFO before being written to the bus. The FIFO has word-sized entries, so any halfwords or bytes transferred from the bus will be zero-extended before being put in the FIFO. Words are not extended in any way. The Byte Swap (BSWP) field in CCRx determines if any modifications are to be performed on the read data from the zero-extension unit. This allows data reformatting such as endianness-conversion.

Figure 21-1. Byte Swapping the FIFO Inputs



21.4.3 Arbitration

Arbitration between the channels is performed at the end of each burst. If no other channels have pending transfers, the current channel continues uninterrupted.

In Fixed Priority Mode, if a channel of higher priority is enabled when another channel is transferring data, the channel of higher priority will preempt the other channel. When the preempting channel has completed, the arbiter will grant control to the original channel so it can complete its transfer.

In Round-Robin Mode, other channels with transfers pending will preempt the current channel in a round-robin fashion. This eliminates the possibility of starvation.

21.4.4 Aborting Transfers

Transfers on any channel can be gracefully aborted by writing a one to the corresponding Channel Disable bit in the Control Register (CR.CHxDIS). Note that in order to successfully write to CHxDIS, the same write operation must also write a one to the corresponding Channel Enable bit (CR.CHxEN). Successfully writing to CHxDIS will result in both CHxEN and CHxDIS being cleared. CHxEN can not be cleared by simply writing a zero to it.

The hardware will disable the transfer as soon as possible. Since the transfer must terminate gracefully, the CHxEN bits may not be cleared immediately after writing CHxDIS. The user could poll CHxDIS to check when the channel has been disabled.

21.4.5 Interrupts

The MDMA can generate an interrupts when a channel has completed a transfer or when a DMA transfer causes a bus error. Interrupts are only generated if enabled in the Interrupt Mask Register (IMR). IMR is read-only, but can be modified through the write-only Interrupt Enable Register (IER) and Interrupt Disable Register (IDR). An interrupt is enabled by writing a one to the corresponding bit in the IER. An interrupt is disabled by writing a one to the corresponding bit in the IDR. If an interrupt occurs, the corresponding bit in the Interrupt Status Register (ISR) is set and an interrupt request is generated. Bits in ISR and their corresponding interrupt request can be cleared by writing to the appropriate bits in the Interrupt Clear Register (ICR).

21.4.6 Bus Errors

Any bus errors from transfers on a channel will automatically disable the channel. Other channels will not be affected by this. An interrupt is generated if not masked by the Interrupt Mask Register (IMR).

21.5 Single Transfer Mode

The Single Transfer Mode (STM) is the simplest transfer mode. The software programs the registers controlling the channel, writes the correct enable bit (CR.CHxEN), and the transfer starts. The transfer can be preempted if a channel with higher priority is enabled before the transfer completes, or if we use Round-Robin Mode.

When the transfer completes, the CR.CHxEN bit is automatically cleared. If the Channel Complete interrupt is enabled in IMR (IMR.CHxC is one), an interrupt request is generated.

In order to perform a transfer in STM, the following steps must be performed:

1. Make sure that the channel is free by checking the CR.CHxEN bit, or by waiting for a Transfer Complete interrupt from the channel.
2. Set up the Read Address Register (RARx), Write Address Register (WARx) and Channel Control Register (CCRx) associated with the channel.
3. Enable the desired interrupts by writing a one to the corresponding bits in the Interrupt Enable Register (IER).
4. Write a one to CR.CHxEN to start the transfer. Make sure the Channel Mode bit (CR.CHxM) for the channel is zero (channel is in Single Transfer Mode)
5. When the transfer completes, the CHxEN bit is automatically cleared. If the Channel Complete interrupt for the channel is enabled, the corresponding bit in the Interrupt Status Register (ISR.CHxC) is set.

21.6 Descriptor Mode

The Descriptor Mode (DM) performs a series of single transfers. Data describing the transfers to be performed are written to memory by software, forming a queue of descriptors, each descriptor describing a transfer to be performed.

21.6.1 Setting Up and Using the Descriptors

Before being able to use the Descriptor Mode, the channel's Descriptor Start Address (DSARx) register must be initialized to point to the first descriptor in the queue. Thereafter, the Current Descriptor Address Register (CDARx) must be initialized to the same value.

When the CR.CHxEN bit is written to one, hardware will read the first descriptor in the queue, and perform the transfer described therein. When this transfer has finished, the hardware will update the descriptor associated with the transfer, clearing the V bit in the descriptor data structure located in memory. Thereafter, the hardware will read the next descriptor in the queue. The address of this descriptor is dependent on the L bit of the descriptor that just completed, see Section 21.6.2.

If the new descriptor has its V bit set, the transfer described by this descriptor will be performed. When the transfer is complete, the descriptor will be written back to memory, with the V bit cleared. Thereafter, the next descriptor will be read. This continues until a descriptor with a cleared V bit is read. The queue is then empty, and all transfers described in the queue have been performed. The CR.CHxEN bit will then be cleared, and the channel will become idle.

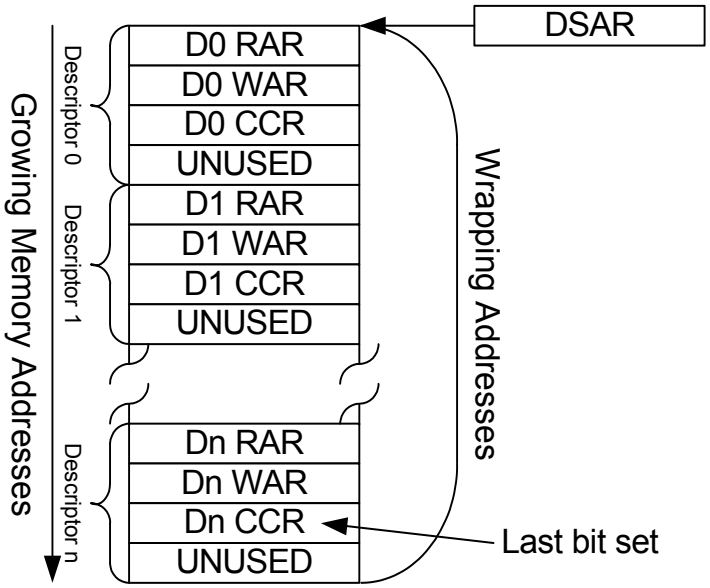
In order to restart the channel, the descriptor pointed to by the Current Descriptor Address Register (CDARx) for the channel must be updated by writing to the appropriate memory locations. Thereafter, the CHxEN bit must be written to one. This will cause the descriptor to be read into the MDMA, and the transfer will start.

21.6.2 Descriptor Organization

The descriptor list is implemented as a ring of descriptors placed in memory. The length of the descriptor ring is programmable; the last descriptor in the ring has its L bit set, indicating that the next element in the ring is at the address pointed to by the Descriptor Start Address Register (DSARx).

Each descriptor consists of four words. When a descriptor is loaded into the MDMA, the first three of these words are read into the appropriate registers, while the fourth word is discarded.

Figure 21-2. Descriptors in Memory



21.6.3 Adding Descriptors to a List

In order to add descriptors to a list, the following actions must be performed:

1. Check if there are free entries in the list for the desired channel. Any entry with the V bit cleared is free. If no entries are free, wait until an entry becomes free.
2. Find the first free entry in the list. This can be done by scanning the descriptor list from the entry pointed to by CDARx downto the first descriptor with the V bit cleared.
3. Update the free entry by writing the desired values to the correct memory locations.
4. Make sure the CHxM bit for the channel is one (channel is in Descriptor Mode), and write a one to the CHxEN bit. Writing a one to CHxEN when the CHxEN bit is one has no effect, so the user does not need to check the state of CHxEN before adding descriptors to the list.

21.6.4 Interrupts in Descriptor Mode

Interrupt on transfer complete and bus error can be enabled as described in Section 21.4.5.

21.7 User interface

Table 21-1. MDMA Register Memory Map

Offset	Register	Register Name	Channel	Access	Reset
0x00	Control Register	CR	-	Read/Write	0x00000000
0x04	Interrupt Enable Register	IER	-	Write-only	0x00000000
0x08	Interrupt Disable Register	IDR	-	Write-only	0x00000000
0x0C	Interrupt Mask Register	IMR	-	Read-only	0x00000000
0x10	Interrupt Status Register	ISR	-	Read-only	0x00000000
0x14	Interrupt Clear Register	ICR	-	Write-only	0x00000000
0x18	Parameter Register	PR	-	Read-only	- (1)
0x1C	Version Register	VR	-	Read-only	- (1)
0x20	Descriptor Start Address 0	DSAR0	0	Read/Write	0x00000000
0x24	Descriptor Start Address 1	DSAR1	1	Read/Write	0x00000000
0x28	Descriptor Start Address 2	DSAR2	2	Read/Write	0x00000000
0x2C	Descriptor Start Address 3	DSAR3	3	Read/Write	0x00000000
0x30-3C	RESERVED	-	-	Read-only	0x00000000
0x40	Current Descriptor Address Register 0	CDAR0	0	Read/Write	0x00000000
0x44	Read Address Register 0	RAR0	0	Read/Write	0x00000000
0x48	Write Address Register 0	WAR0	0	Read/Write	0x00000000
0x4C	Channel Control Register 0	CCR0	0	Read/Write	0x00000000
0x50	Current Descriptor Address Register 1	CDAR1	1	Read/Write	0x00000000
0x54	Read Address Register 1	RAR1	1	Read/Write	0x00000000
0x58	Write Address Register 1	WAR1	1	Read/Write	0x00000000
0x5C	Channel Control Register 1	CCR1	1	Read/Write	0x00000000
0x60	Current Descriptor Address Register 2	CDAR2	2	Read/Write	0x00000000
0x64	Read Address Register 2	RAR2	2	Read/Write	0x00000000
0x68	Write Address Register 2	WAR2	2	Read/Write	0x00000000
0x6C	Channel Control Register 2	CCR2	2	Read/Write	0x00000000
0x70	Current Descriptor Address Register 3	CDAR3	3	Read/Write	0x00000000
0x74	Read Address Register 3	RAR3	3	Read/Write	0x00000000
0x78	Write Address Register 3	WAR3	3	Read/Write	0x00000000
0x7C	Channel Control Register 3	CCR3	3	Read/Write	0x00000000

1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

Note: The number of DMA channels is device-specific. Not all devices will implement all four channels. These devices will not implement all user interface registers. Unimplemented registers will always read as 0.

21.7.1 Control Register

Name: CR

Access Type : Read/Write

Offset: 0x00

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	ARB
23	22	21	20	19	18	17	16
-	-	-	-	CH3DIS	CH2DIS	CH1DIS	CH0DIS
15	14	13	12	11	10	9	8
-	-	-	-	CH3M	CH2M	CH1M	CH0M
7	6	5	4	3	2	1	0
-	-	-	-	CH3EN	CH2EN	CH1EN	CH0EN

- **ARB: Arbitration Mode**
  0: The MDMA is in Fixed Priority Mode.
  1: The MDMA is in Round-Robin Mode.
- **CHxDIS: Channel Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the MDMA channel after the current transfer has completed.
  To avoid hazards, CHxDIS bits can only be changed by writing a value to CR where the corresponding CHxEN bit is set.
  This bit is automatically cleared by hardware when the corresponding channel has been disabled.
- **CHxM: Channel Mode**
  0: The channel is in Single Transfer Mode.
  1: The channel is in Descriptor Mode.
  To avoid hazards, CHxM bits can only be changed by writing a value to CR where the corresponding CHxEN bit is set.
- **CHxEN: Channel Enable**
  Writing a zero to this bit this bit has no effect.
  Writing a one to this bit enables the channel for DMA transfer.
  This bit is automatically cleared if the transfer completes when the channel is in single transfer mode.
  This bit is automatically cleared if the transfer completes when the channel is in descriptor mode, and the next descriptor read in has a cleared Valid bit.
  This bit is automatically cleared when the corresponding channel has been disabled by writing a one to CHxDIS.

21.7.2 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x04

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	BERR3	BERR2	BERR1	BERR0

7	6	5	4	3	2	1	0
-	-	-	-	CH3C	CH2C	CH1C	CH0C

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register clears the corresponding bit in Interrupt Mask Register (IMR).

21.7.3 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x08

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	BERR3	BERR2	BERR1	BERR0

7	6	5	4	3	2	1	0
-	-	-	-	CH3C	CH2C	CH1C	CH0C

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register sets the corresponding bit in Interrupt Mask Register (IMR).

21.7.4 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x0C

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	BERR3	BERR2	BERR1	BERR0

7	6	5	4	3	2	1	0
-	-	-	-	CH3C	CH2C	CH1C	CH0C

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

21.7.5 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x10

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	BERR3	BERR2	BERR1	BERR0

7	6	5	4	3	2	1	0
-	-	-	-	CH3C	CH2C	CH1C	CH0C

- **BERRx: Channel Bus Error**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set when the channel has encountered a bus error and has an interrupt request pending. Upon receiving a bus error, the affected channel is automatically disabled.
- **CHxC: Channel Complete**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set when the channel has completed a transfer and has an interrupt request pending.

21.7.6 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x14

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	BERR3	BERR2	BERR1	BERR0

7	6	5	4	3	2	1	0
-	-	-	-	CH3C	CH2C	CH1C	CH0C

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register clears the corresponding bit in ISR and the corresponding interrupt request.

21.7.7 Parameter Register

Name: PR

Access Type: Read-only

Offset: 0x18

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	BURST	

7	6	5	4	3	2	1	0
-	-	-	-	CH3I	CH2I	CH1I	CH0I

- **BURST: Maximum Burst Size**
  The maximum burst size that can be used is a function of the FIFO size, which can be different in different devices. This field gives the largest burst size that is supported by the device.
  0: Single transfer
  1: 4-beat burst
  2: 8-beat burst
  3: 16-beat burst
- **CHxI: Channel Implemented**
  0: The channel is not implemented in the current device, and cannot be used.
  1: The channel is implemented in the current device.

21.7.8 Version Register

Name: VR

Access Type: Read-only

Offset: 0x1C

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VERSION: Version Number**
  Version number of the module. No functionality associated.

21.7.9 Descriptor Start Address Register x

Name: DSAR0, DSAR1, DSAR2, DSAR3

Access Type: Read/Write

Offset: 0x20, 0x24, 0x28, 0x2C

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
DSAR[31:24]							

23	22	21	20	19	18	17	16
DSAR[23:16]							

15	14	13	12	11	10	9	8
DSAR[15:8]							

7	6	5	4	3	2	1	0
DSAR[7:0]							

- **DSAR: Descriptor Start Address Register**

  The address of the first descriptor in the chain. When the hardware has read a descriptor with the CCRx.L bit set, the next descriptor will be read from the address in DSAR. Must be aligned to word size.

21.7.10 Current Descriptor Address Register x

Name: CDAR0, CDAR1, CDAR2, CDAR3

Access Type: Read/Write

Offset: 0x40, 0x50, 0x60, 0x70

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
CDAR[31:24]							

23	22	21	20	19	18	17	16
CDAR[23:16]							

15	14	13	12	11	10	9	8
CDAR[15:8]							

7	6	5	4	3	2	1	0
CDAR[7:0]							

- **CDAR: Current Descriptor Address Register**

  The memory address pointing to the currently active descriptor. This is either the descriptor of the current access if the channel is enabled, or the address of the descriptor that will be loaded if the channel is not enabled.

  Must be word-aligned.

  Not used if the channel is not in Descriptor Mode.

21.7.11 Read Address Register x

Name: RAR0, RAR1, RAR2, RAR3

Access Type: Read/Write

Offset: 0x44, 0x54, 0x64, 0x74

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
RAR[31:24]							

23	22	21	20	19	18	17	16
RAR[23:16]							

15	14	13	12	11	10	9	8
RAR[15:8]							

7	6	5	4	3	2	1	0
RAR[7:0]							

- **RAR: Read Address Register**
  The memory address that the next read access will be done from. Must be aligned according to the transfer size.

21.7.12 Write Address Register x

Name: WAR0, WAR1, WAR2, WAR3

Access Type: Read/Write

Offset: 0x48, 0x58, 0x68, 0x78

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
WAR[31:24]							

23	22	21	20	19	18	17	16
WAR[23:16]							

15	14	13	12	11	10	9	8
WAR[15:8]							

7	6	5	4	3	2	1	0
WAR[7:0]							

- **WAR: Write Address Register**
  The memory address that the next write access will be done to. Must be aligned according to the transfer size.

21.7.13 Channel Control Register x

Name: CCR0, CCR1, CCR2, CCR3

Access Type: Read/Write

Offset: 0x4C, 0x5C, 0x6C, 0x7C

Reset Value: 0x000000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	BSWP	

23	22	21	20	19	18	17	16
-	L	V	TCIE	BURST		SIZE	

15	14	13	12	11	10	9	8
TCNT[15:8]							

7	6	5	4	3	2	1	0
TCNT[7:0]							

- **BSWP: Byte Swap**
  Allows swapping of the transferred bytes, see Section 21.4.2. Assuming the word output from the zero-extension module contains the bytes {a, b, c, d}. The following will be put into the FIFO for transmission:
  0: {a, b, c, d}
  1: {d, c, b, a}
  2: {c, d, a, b}
  3: {b, a, d, c}
- **L: Last Descriptor in Chain**
  Used only if the channel is in Descriptor Mode.
  0: The descriptor read in is not the last descriptor in the chain. The next descriptor to be read is located at the address of the previously read descriptor + 4 words.
  1: The descriptor read in is the last descriptor in the chain. The next descriptor to be read is located at the address in the Descriptor Start Address Register (DSARx) for the channel.
- **V: Descriptor Valid**
  Used only if the channel is in Descriptor Mode.
  0: The descriptor read in is not valid.
  1: The descriptor read in is valid.
- **TCIE: Transfer Complete Interrupt Enable**
  0: Transfer Complete does not set the ISR.CHxC bit.
  1: Transfer Complete sets the ISR.CHxC bit.
- **BURST: Transfer Burst Size**
  Indicates the size of the burst used for data transfer. The MDMA will always try to use this burst size to perform transfers, but may be forced to use smaller sizes since the transfer count may not be perfectly divisible by the transfer data size.
  0: Single transfer
  1: 4-beat burst
  2: 8-beat burst
  3: 16-beat burst

- **SIZE: Transfer Data Size**
  Indicates the size of data to transfer.
  0: Byte
  1: Halfword
  2: Word
  3: Unused
- **TCNT: Transfer Count**
  The number of data to transfer. The size of each data is given in the SIZE field.
  Configuring a transfer with a TCNT of 0 is illegal, and may result in UNDEFINED behavior.

21.8 Module Configuration

The specific configuration for each MDMA instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Refer to the Power Manager chapter for details.

Table 21-2. Module Configuration

Feature	MDMA
Channels	1
Maximum burst size	Single Transfer

Table 21-3. Module Clock Name

Module name	Clock name	Description
MDMA	CLK_MDMA_HSB	HSB clock
	CLK_MDMA_PB	Peripheral Bus clock from the PBC clock domain

Table 21-4. Register Reset Values

Register	Reset Value
VR	0x00000101
PR	0x00000001

22. Secure Access Unit (SAU)

Rev: 1.1.1.3

22.1 Features

- **Remaps registers in memory regions protected by the MPU to regions not protected by the MPU**
- **Programmable physical address for each channel**
- **Two modes of operation: Locked and Open**
  - **In Locked Mode, access to a channel must be preceded by an unlock action**
    - **An unlocked channel remains open only for a specific amount of time, if no access is performed during this time, the channel is relocked**
    - **Only one channel can be open at a time, opening a channel while another one is open locks the first one**
    - **Access to a locked channel is denied, a bus error and optionally an interrupt is returned**
    - **If a channel is relocked due to an unlock timeout, an interrupt can optionally be generated**
  - **In Open Mode, all channels are permanently unlocked**

22.2 Overview

In many systems, erroneous access to peripherals can lead to catastrophic failure. An example of such a peripheral is the Pulse Width Modulator (PWM) used to control electric motors. The PWM outputs a pulse train that controls the motor. If the control registers of the PWM module are inadvertently updated with wrong values, the motor can start operating out of control, possibly causing damage to the application and the surrounding environment. However, sometimes the PWM control registers must be updated with new values, for example when modifying the pulse train to accelerate the motor. A mechanism must be used to protect the PWM control registers from inadvertent access caused by for example:

- Errors in the software code
- Transient errors in the CPU caused by for example electrical noise altering the execution path of the program

To improve the security in a computer system, the AVR32UC implements a Memory Protection Unit (MPU). The MPU can be set up to limit the accesses that can be performed to specific memory addresses. The MPU divides the memory space into regions, and assigns a set of access restrictions on each region. Access restrictions can for example be read/write if the CPU is in supervisor mode, and read-only if the CPU is in application mode. The regions can be of different size, but each region is usually quite large, e.g. protecting 1 kilobyte of address space or more. Furthermore, access to each region is often controlled by the execution state of the CPU, i.e. supervisor or application mode. Such a simple control mechanism is often too inflexible (too coarse-grained chunks) and with too much overhead (often requiring system calls to access protected memory locations) for simple or real-time systems such as embedded microcontrollers.

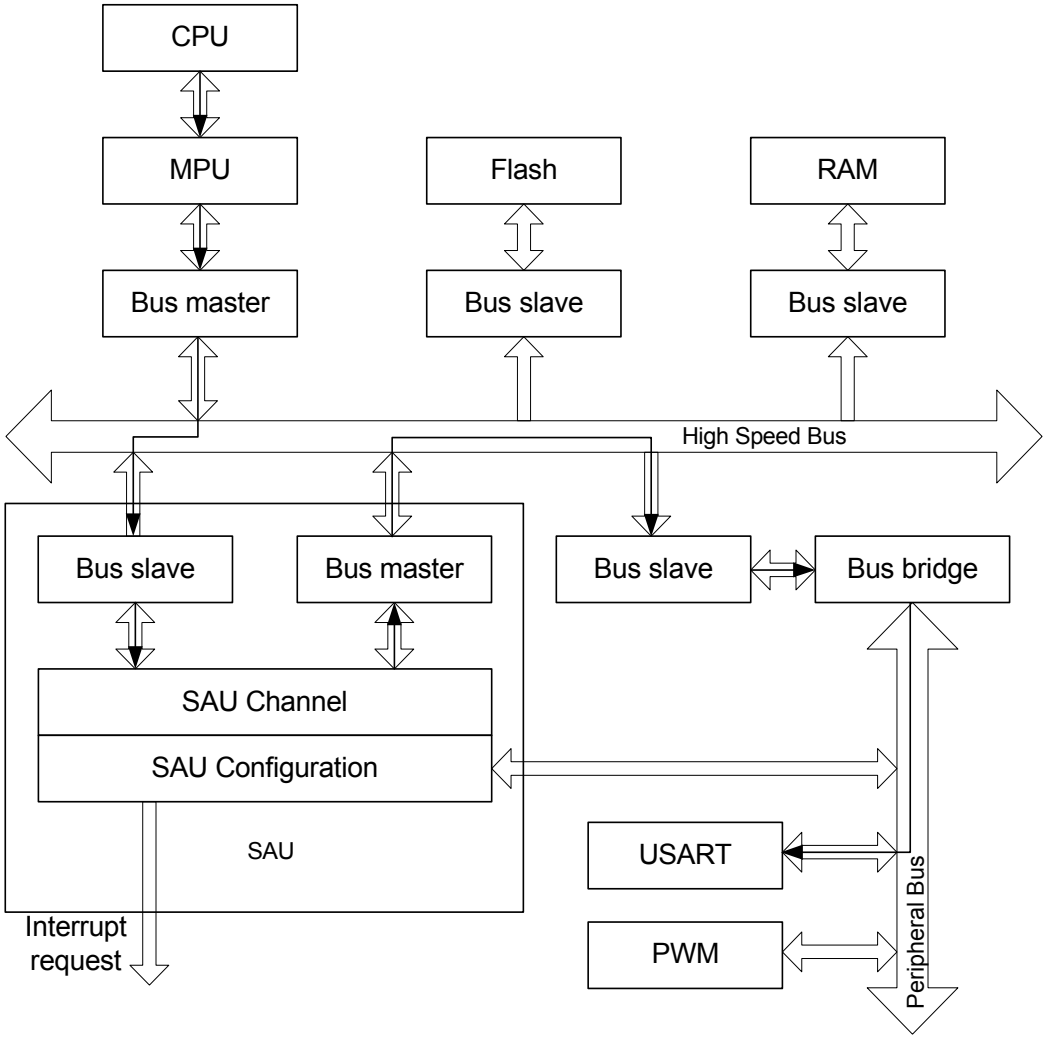
Usually, the Secure Access Unit (SAU) is used together with the MPU to provide the required security and integrity. The MPU is set up to protect regions of memory, while the SAU is set up to provide a secure channel into specific memory locations that are protected by the MPU. These specific locations can be thought of as fine-grained overrides of the general coarse-grained protection provided by the MPU.

22.3 Block Diagram

Figure 22-1 presents the SAU integrated in an example system with a CPU, some memories, some peripherals, and a bus system. The SAU is connected to both the Peripheral Bus (PB) and the High Speed Bus (HSB). Configuration of the SAU is done via the PB, while memory accesses are done via the HSB. The SAU receives an access on its HSB slave interface, remaps it, checks that the channel is unlocked, and if so, initiates a transfer on its HSB master interface to the remapped address.

The thin arrows in Figure 22-1 exemplifies control flow when using the SAU. The CPU wants to read the RX Buffer in the USART. The MPU has been configured to protect all registers in the USART from user mode access, while the SAU has been configured to remap the RX Buffer into a memory space that is not protected by the MPU. This unprotected memory space is mapped into the SAU HSB slave space. When the CPU reads the appropriate address in the SAU, the SAU will perform an access to the desired RX buffer register in the USART, and thereafter return the read results to the CPU. The return data flow will follow the opposite direction of the control flow arrows in Figure 22-1.

Figure 22-1. SAU Block Diagram



22.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

22.4.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the SAU, the SAU will stop functioning and resume operation after the system wakes up from sleep mode.

22.4.2 Clocks

The SAU has two bus clocks connected: One High Speed Bus clock (CLK_SAU_HSB) and one Peripheral Bus clock (CLK_SAU_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by writing to the Power Manager. The user has to ensure that CLK_SAU_HSB is not turned off before accessing the SAU. Likewise, the user must ensure that no bus access is pending in the SAU before disabling CLK_SAU_HSB. Failing to do so may deadlock the High Speed Bus.

22.4.3 Interrupt

The SAU interrupt request line is connected to the interrupt controller. Using the SAU interrupt requires the interrupt controller to be programmed first.

22.4.4 Debug Operation

When an external debugger forces the CPU into debug mode, the SAU continues normal operation. If the SAU is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

22.5 Functional Description

22.5.1 Enabling the SAU

The SAU is enabled by writing a one to the Enable (EN) bit in the Control Register (CR). This will set the SAU Enabled (EN) bit in the Status Register (SR).

22.5.2 Configuring the SAU Channels

The SAU has a set of channels, mapped in the HSB memory space. These channels can be configured by a Remap Target Register (RTR), located at the same memory address. When the SAU is in normal mode, the SAU channel is addressed, and when the SAU is in setup mode, the RTR can be addressed.

Before the SAU can be used, the channels must be configured and enabled. To configure a channel, the corresponding RTR must be programmed with the Remap Target Address. To do this, make sure the SAU is in setup mode by writing a one to the Setup Mode Enable (SEN) bit in CR. This makes sure that a write to the RTR address accesses the RTR, not the SAU channel. Thereafter, the RTR is written with the address to remap to, typically the address of a specific PB register. When all channels have been configured, return to normal mode by writing a one to the Setup Mode Disable (SDIS) in CR. The channels can now be enabled by writing ones to the corresponding bits in the Channel Enable Registers (CERH/L).

The SAU is only able to remap addresses above 0xFFFC0000.

22.5.2.1 *Protecting SAU configuration registers*

In order to prevent the SAU configuration registers to be changed by malicious or runaway code, they should be protected by the MPU as soon as they have been configured. Maximum security is provided in the system if program memory does not contain any code to unprotect the configuration registers in the MPU. This guarantees that runaway code can not accidentally unprotect and thereafter change the SAU configuration registers.

22.5.3 Lock Mechanism

The SAU can be configured to use two different access mechanisms: Open and Locked. In Open Mode, SAU channels can be accessed freely after they have been configured and enabled. In order to prevent accidental accesses to remapped addresses, it is possible to configure the SAU in Locked Mode. Writing a one to the Open Mode bit in the CONFIG register (CONFIG.OPEN) will enable Open Mode. Writing a zero to CONFIG.OPEN will enable Locked Mode.

When using Locked Mode, the lock mechanism must be configured by writing a user defined key value to the Unlock Key (UKEY) field in the Configuration Register (CONFIG). The number of CLK_SAU_HSB cycles the channel remains unlocked must be written to the Unlock Number of Clock Cycles (UCYC) field in CONFIG.

Access control to the SAU channels is enabled by means of the Unlock Register (UR), which resides in the same address space as the SAU channels. Before a channel can be accessed, the unlock register must be written with th correct key and channel number (single write access). Access to the channel is then permitted for the next CONFIG.UCYC clock cycles, or until a successful access to the unlocked channel has been made.

Only one channel can be unlocked at a time. If any other channel is unlocked at the time of writing UR, this channel will be automatically locked before the channel addressed by the UR write is unlocked.

An attempted access to a locked channel will be aborted, and the Channel Access Unsuccessful bit (SR.CAU) will be set.

Any pending errors bits in SR must be cleared before it is possible to access UR. The following SR bits are defined as error bits: EXP, CAU, URREAD, URKEY, URES, MBERROR, RTRADR. If any of these bits are set while writing to UR, the write is aborted and the Unlock Register Error Status (URES) bit in SR is set.

22.5.4 Normal Operation

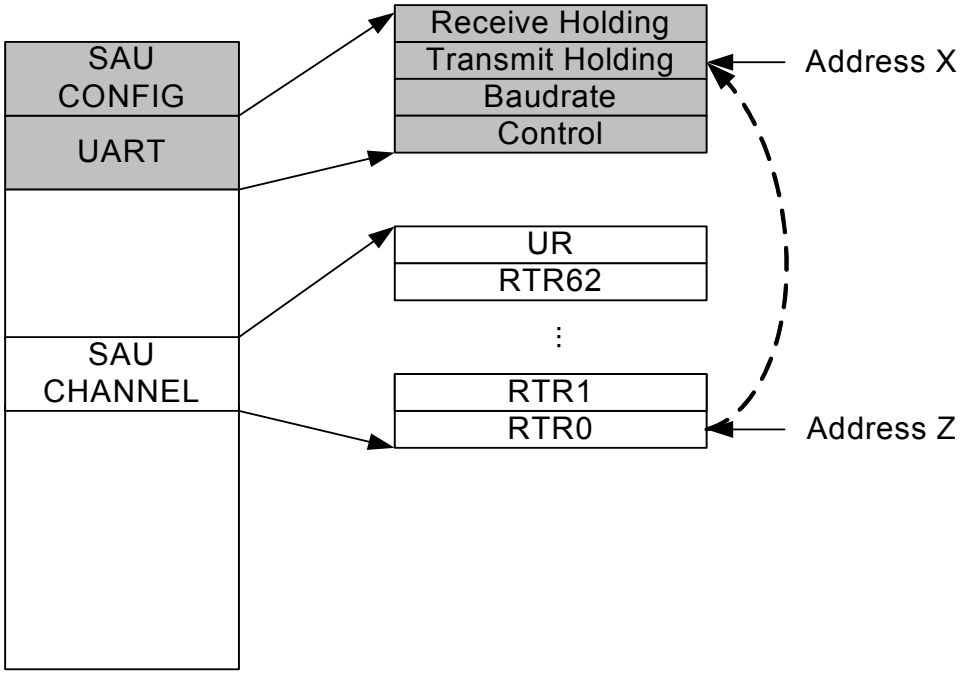
The following sequence must be used in order to access a SAU channel in normal operation (CR.SEN=0):

1. If not in Open Mode, write the unlock key to UR.KEY and the channel number to UR.CHANNEL.
2. Perform the read or write operation to the SAU channel. If not in Open Mode, this must be done within CONFIG.UCYC clock cycles of unlocking the channel. The SAU will use its HSB master interface to remap the access to the target address pointed to by the corresponding RTR.
3. To confirm that the access was successful, wait for the IDLE transfer status bit (SR.IDLE) to indicate the operation is completed. Then check SR for possible error conditions. The SAU can be configured to generate interrupt requests or a Bus Error Exception if the access failed.

22.5.4.1 *Operation example*

Figure 22-2 shows a typical memory map, consisting of some memories, some simple peripherals, and a SAU with multiple channels and an Unlock Register (UR). Imagine that the MPU has been set up to disallow all accesses from the CPU to the grey modules. Thus the CPU has no way of accessing for example the Transmit Holding register in the UART, present on address X on the bus. Note that the SAU RTRs are not protected by the MPU, thus the RTRs can be accessed. If for example RTR0 is configured to point to address X, an access to RTR0 will be remapped by the SAU to address X according to the algorithm presented above. By programming the SAU RTRs, specific addresses in modules that have generally been protected by the MPU can be performed.

Figure 22-2. Example Memory Map for a System with SAU



22.5.5 Interrupts

The SAU can generate an interrupt request to signal different events. All events that can generate an interrupt request have dedicated bits in the Status Register (SR). An interrupt request will be generated if the corresponding bit in the Interrupt Mask Register (IMR) is set. Bits in IMR are set by writing a one to the corresponding bit in the Interrupt Enable Register (IER), and cleared by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The interrupt request remains active until the corresponding bit in SR is cleared by writing a one to the corresponding bit in the Interrupt Clear Register (ICR).

The following SR bits are used for signalling the result of SAU accesses:

- RTR Address Error (RTRADR) is set if an illegal address is written to the RTRs. Only addresses in the range 0xFFFC0000-0xFFFFFFFF are allowed.
- Master Interface Bus Error (MBERROR) is set if any of the conditions listed in Section 22.5.7 occurred.

- Unlock Register Error Status (URES) is set if an attempt was made to unlock a channel by writing to the Unlock Register while one or more error bits in SR were set (see Section 22.5.6). The unlock operation was aborted.
- Unlock Register Key Error (URKEY) is set if the Unlock Register was attempted written with an invalid key.
- Unlock Register Read (URREAD) is set if the Unlock Register was attempted read.
- Channel Access Unsuccessful (CAU) is set if the channel access was unsuccessful.
- Channel Access Successful (CAS) is set if the channel access was successful.
- Channel Unlock Expired (EXP) is set if the channel lock expired, with no channel being accessed after the channel was unlocked.

22.5.6 Error bits

If error bits are set when attempting to unlock a channel, SR.URES will be set. The following SR bits are considered error bits:

- EXP
- CAU
- URREAD
- URKEY
- URES
- MBERROR
- RTRADR

22.5.7 Bus Error Responses

By writing a one to the Bus Error Response Enable bit (CR.BERREN), serious access errors will be configured to return a bus error to the CPU. This will cause the CPU to execute its Bus Error Data Fetch exception routine.

The conditions that can generate a bus error response are:

- Reading the Unlock Register
- Trying to access a locked channel
- The SAU HSB master receiving a bus error response from its addressed slave

22.5.8 Disabling the SAU

To disable the SAU, the user must first ensure that no SAU bus operations are pending. This can be done by checking that the SR.IDLE bit is set.

The SAU may then be disabled by writing a one to the Disable (DIS) bit in CR.

22.6 User Interface

The following addresses are used by SAU channel configuration registers. All offsets are relative to the SAU's PB base address.

Table 22-1. SAU Configuration Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	0x00000000
0x04	Configuration Register	CONFIG	Write-only	0x00000000
0x08	Channel Enable Register High	CERH	Read/Write	0x00000000
0x0C	Channel Enable Register Low	CERL	Read/Write	0x00000000
0x10	Status Register	SR	Read-only	0x00000400
0x14	Interrupt Enable Register	IER	Write-only	0x00000000
0x18	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x20	Interrupt Clear Register	ICR	Write-only	0x00000000
0x24	Parameter Register	PARAMETER	Read-only	-(1)
0x28	Version Register	VERSION	Read-only	-(1)

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

The following addresses are used by SAU channel registers. All offsets are relative to the SAU's HSB base address. The number of channels implemented is device specific, refer to the Module Configuration section at the end of this chapter.

Table 22-2. SAU Channel Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Remap Target Register 0	RTR0	Read/Write	N/A
0x04	Remap Target Register 1	RTR1	Read/Write	N/A
0x08	Remap Target Register 2	RTR2	Read/Write	N/A
...	...	...	...	...
0x04*n	Remap Target Register n	RTRn	Read/Write	N/A
0xFC	Unlock Register	UR	Write-only	N/A

22.6.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	BERRDIS	BERREN	SDIS	SEN	DIS	EN

- **BERRDIS: Bus Error Response Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables Bus Error Response from the SAU.
- **BERREN: Bus Error Response Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables Bus Error Response from the SAU.
- **SDIS: Setup Mode Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit exits setup mode.
- **SEN: Setup Mode Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enters setup mode.
- **DIS: SAU Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the SAU.
- **EN: SAU Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the SAU.

22.6.2 Configuration Register

Name: CONFIG

Access Type: Write-only

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	OPEN

15	14	13	12	11	10	9	8
UCYC							

7	6	5	4	3	2	1	0
UKEY							

- **OPEN: Open Mode Enable**
  Writing a zero to this bit disables open mode.
  Writing a one to this bit enables open mode.
- **UCYC: Unlock Number of Clock Cycles**
  Once a channel has been unlocked, it remains unlocked for this amount of CLK_SAU_HSB clock cycles or until one access to a channel has been made.
- **UKEY: Unlock Key**
  The value in this register must be written into UR.KEY to unlock a channel.

22.6.3 Channel Enable Register High

Name: CERH

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	CERH[30:24]						

23	22	21	20	19	18	17	16
CERH[23:16]							

15	14	13	12	11	10	9	8
CERH[15:8]							

7	6	5	4	3	2	1	0
CERH[7:0]							

- **CERH[n]: Channel Enable Register High**
  0: Channel (n+32) is not enabled.
  1: Channel (n+32) is enabled.

22.6.4 Channel Enable Register Low

Name: CERL

Access Type: Read/Write

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CERL[31:24]							

23	22	21	20	19	18	17	16
CERL[23:16]							

15	14	13	12	11	10	9	8
CERL[15:8]							

7	6	5	4	3	2	1	0
CERL[7:0]							

- **CERL[n]: Channel Enable Register Low**
  0: Channel n is not enabled.
  1: Channel n is enabled.

22.6.5 Status Register

Name: SR

Access Type: Read-only

Offset: 0x10

Reset Value: 0x00000400

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	IDLE	SEN	EN

7	6	5	4	3	2	1	0
RTRADR	MBERROR	URES	URKEY	URREAD	CAU	CAS	EXP

- **IDLE**
  This bit is cleared when a read or write operation to the SAU channel is started.
  This bit is set when the operation is completed and no SAU bus operations are pending.
- **SEN: SAU Setup Mode Enable**
  This bit is cleared when the SAU exits setup mode.
  This bit is set when the SAU enters setup mode.
- **EN: SAU Enabled**
  This bit is cleared when the SAU is disabled.
  This bit is set when the SAU is enabled.
- **RTRADR: RTR Address Error**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set if, in the configuration phase, an RTR was written with an illegal address, i.e. the upper 16 bits in the address were different from 0xFFFC, 0xFFFD, 0xFFFE or 0xFFFF.
- **MBERROR: Master Interface Bus Error**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set if a channel access generated a transfer on the master interface that received a bus error response from the addressed slave.
- **URES: Unlock Register Error Status**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set if an attempt was made to unlock a channel by writing to the Unlock Register while one or more error bits were set in SR. The unlock operation was aborted.
- **URKEY: Unlock Register Key Error**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set if the Unlock Register was attempted written with an invalid key.
- **URREAD: Unlock Register Read**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set if the Unlock Register was read.

- **CAU: Channel Access Unsuccessful**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set if channel access was unsuccessful, i.e. an access was attempted to a locked or disabled channel.
- **CAS: Channel Access Successful**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set if channel access successful, i.e. one access was made after the channel was unlocked.
- **EXP: Channel Unlock Expired**
  This bit is cleared when the corresponding bit in ICR is written to one.
  This bit is set if channel unlock has expired, i.e. no access being made after the channel was unlocked.

22.6.6 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RTRADR	MBERROR	URES	URKEY	URREAD	CAU	CAS	EXP

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

22.6.7 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RTRADR	MBERROR	URES	URKEY	URREAD	CAU	CAS	EXP

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

22.6.8 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RTRADR	MBERROR	URES	URKEY	URREAD	CAU	CAS	EXP

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

22.6.9 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RTRADR	MBERROR	URES	URKEY	URREAD	CAU	CAS	EXP

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and any corresponding interrupt request.

22.6.10 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x24

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	CHANNELS					

- **CHANNELS: Number of channels implemented**

22.6.11 Version Register

Name: VERSION

Access Type: Write-only

Offset: 0x28

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

22.6.12 Remap Target Register n

Name: RTRn

Access Type: Read/Write

Offset: n*4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RTR[31:24]							

23	22	21	20	19	18	17	16
RTR[23:16]							

15	14	13	12	11	10	9	8
RTR[15:8]							

7	6	5	4	3	2	1	0
RTR[7:0]							

- **RTR: Remap Target Address for Channel n**

  RTR[31:16] must have one of the following values, any other value will result in UNDEFINED behavior:

  0xFFFC

  0xFFFD

  0xFFFE

  0xFFFF

  RTR[1:0] must be written to 0, any other value will result in UNDEFINED behavior.

22.6.13 Unlock Register

Name: UR

Access Type : Write-only

Offset: 0xFC

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
-							

15	14	13	12	11	10	9	8
KEY							

7	6	5	4	3	2	1	0
-	-	CHANNEL					

- **KEY: Unlock Key**
  The correct key must be written in order to unlock a channel. The key value written must correspond to the key value defined in CONFIG.UKEY.
- **CHANNEL: Channel Number**
  Number of the channel to unlock.

22.7 Module configuration

The specific configuration for each SAU instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Refer to the Power Manager chapter for details.

Table 22-3. Module configuration

Feature	SAU
SAU Channels	16

Table 22-4. Module clock name

Module name	Clock name	Description
SAU	CLK_SAU_HSB	HSB clock
	CLK_SAU_PB	Peripheral Bus clock from the PBB clock domain

Table 22-5. Register Reset Values

Register	Reset Value
VERSION	0x111
PARAMETER	0x010

23. General-Purpose Input/Output Controller (GPIO)

Rev: 2.1.2.5

23.1 Features

- **Each GPIO line features:**
  - **Configurable pin-change, rising-edge, or falling-edge interrupt**
  - **Glitch filter providing rejection of pulses shorter than one clock cycle**
  - **Input visibility and output control**
  - **Multiplexing of peripheral functions on I/O pins**
  - **Programmable internal pull-up resistor**
  - **Programmable internal pull-down resistor**
  - **Programmable output driver strength**
  - **Optional locking of configuration to avoid accidental reconfiguration**

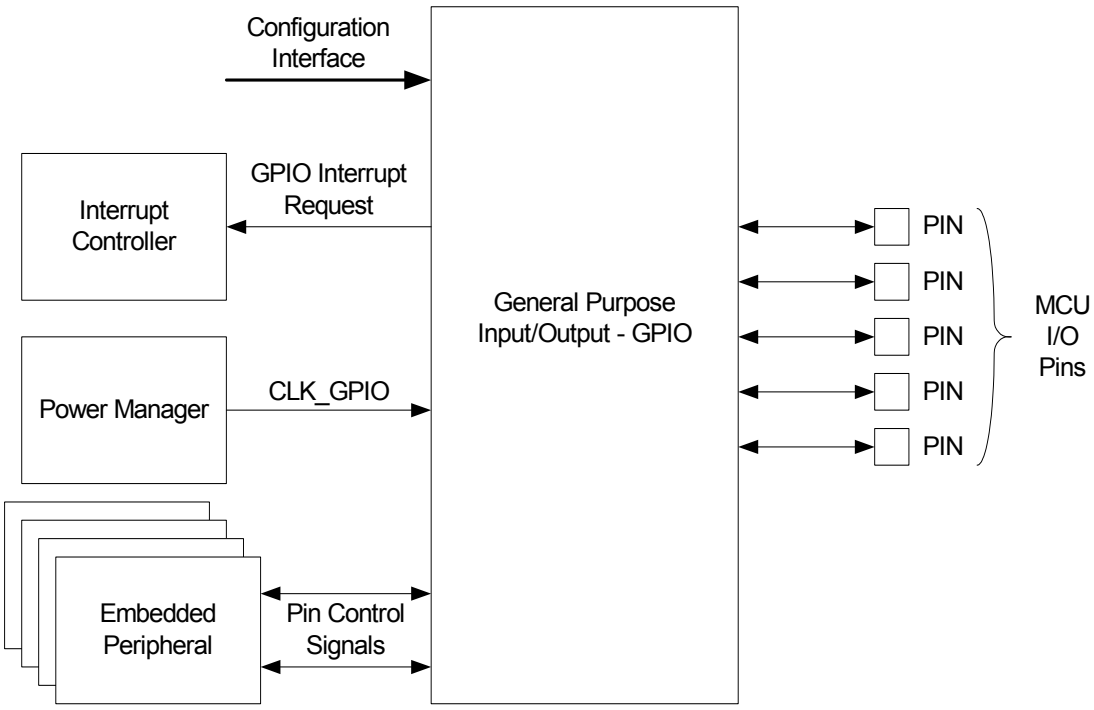
23.2 Overview

The General Purpose Input/Output Controller (GPIO) controls the I/O pins of the microcontroller. Each GPIO pin may be used as a general-purpose I/O or be assigned to a function of an embedded peripheral.

The GPIO is configured using the Peripheral Bus (PB). Some registers can also be configured using the low latency CPU Local Bus. See Section 23.6.2.8 for details.

23.3 Block Diagram

Figure 23-1. GPIO Block Diagram



23.4 I/O Lines Description

Pin Name	Description	Type
GPIOn	GPIO pin n	Digital

23.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

23.5.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the GPIO, the GPIO will stop functioning and resume operation after the system wakes up from sleep mode.

If a peripheral function is configured for a GPIO pin, the peripheral will be able to control the GPIO pin even if the GPIO clock is stopped.

23.5.2 Clocks

The GPIO is connected to a Peripheral Bus clock (CLK_GPIO). This clock is generated by the Power Manager. CLK_GPIO is enabled at reset, and can be disabled by writing to the Power Manager. CLK_GPIO must be enabled in order to access the configuration registers of the GPIO or to use the GPIO interrupts. After configuring the GPIO, the CLK_GPIO can be disabled by writing to the Power Manager if interrupts are not used.

If the CPU Local Bus is used to access the configuration interface of the GPIO, the CLK_GPIO must be equal to the CPU clock to avoid data loss.

23.5.3 Interrupts

The GPIO interrupt request lines are connected to the interrupt controller. Using the GPIO interrupts requires the interrupt controller to be programmed first.

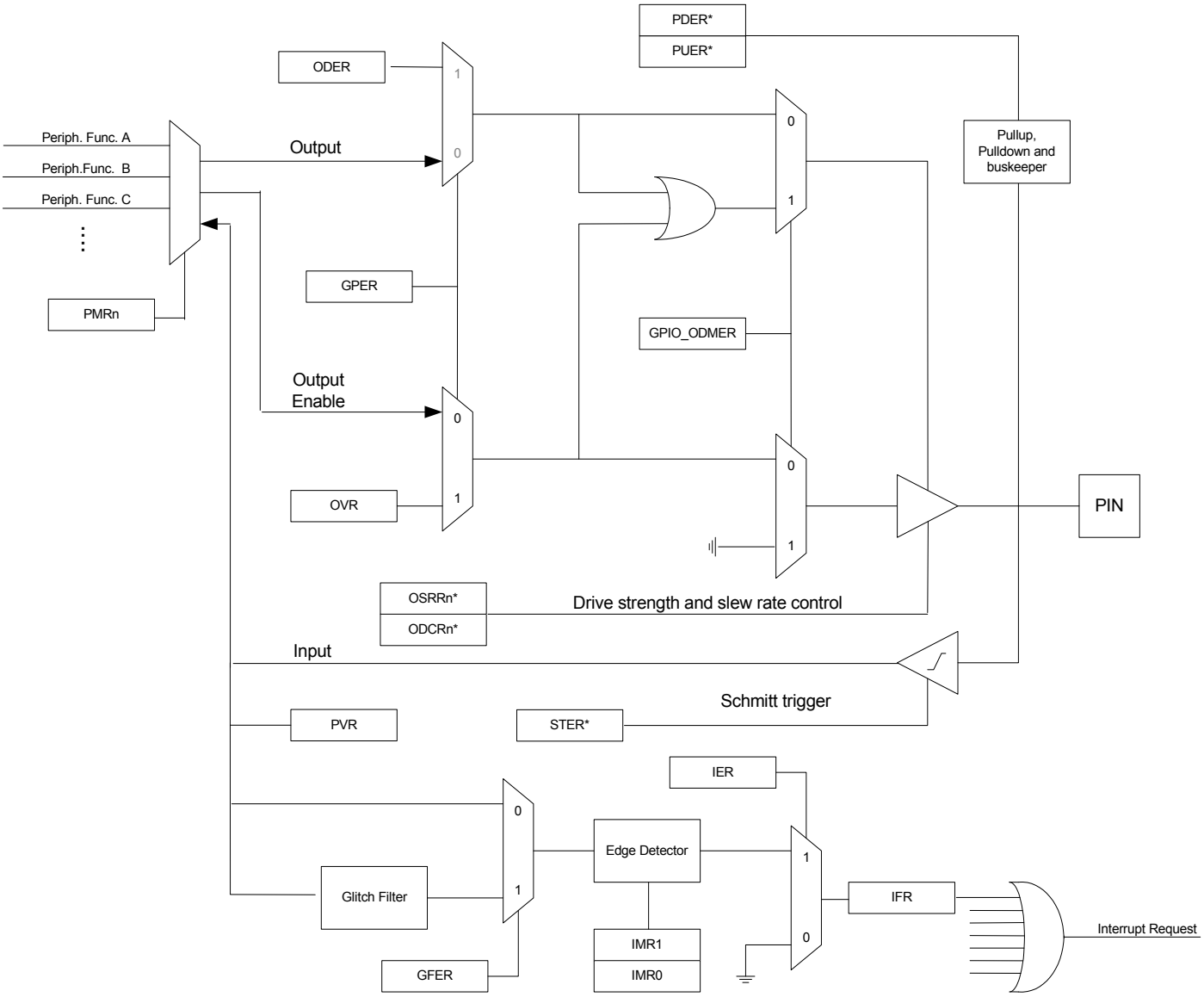
23.5.4 Debug Operation

When an external debugger forces the CPU into debug mode, the GPIO continues normal operation. If the GPIO is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

23.6 Functional Description

The GPIO controls the I/O pins of the microcontroller. The control logic associated with each pin is shown in the figure below.

Figure 23-2. Overview of the GPIO



*) Register value is overrided if a peripheral function that support this function is enabled

23.6.1 Basic Operation

23.6.1.1 Module Configuration

The GPIO user interface registers are organized into ports and each port controls 32 different GPIO pins. Most of the registers supports bit wise access operations such as set, clear and toggle in addition to the standard word access. For details regarding interface registers, refer to Section 23.7.

23.6.1.2 Available Features

Most of the GPIO features are configurable for each product. The programmer must refer to the Module Configuration section and the GPIO Function Multiplexing section in the Package and Pinout chapter for the configuration used in this product.

Product specific settings includes:

- Number of GPIO pins
- Functions implemented on each pin
- Peripheral function(s) multiplexed on each GPIO pin
- Reset state of registers

23.6.1.3 Inputs

The level on each GPIO pin can be read through the Pin Value Register (PVR). This register indicates the level of the GPIO pins regardless of the pins being driven by the GPIO or by an external component. Note that due to power saving measures, the PVR register will only be updated when the corresponding bit in GPER is one or if an interrupt is enabled for the pin, i.e. IER is one for the corresponding pin.

23.6.1.4 Output Control

When the GPIO pin is assigned to a peripheral function, i.e. the corresponding bit in GPER is zero, the peripheral determines whether the pin is driven or not.

When the GPIO pin is controlled by the GPIO, the value of Output Driver Enable Register (ODER) determines whether the pin is driven or not. When a bit in this register is one, the corresponding GPIO pin is driven by the GPIO. When the bit is zero, the GPIO does not drive the pin.

The level driven on a GPIO pin can be determined by writing the value to the corresponding bit in the Output Value Register (OVR).

23.6.1.5 Peripheral Muxing

The GPIO allows a single GPIO pin to be shared by multiple peripheral pins and the GPIO itself. Peripheral pins sharing the same GPIO pin are arranged into peripheral functions that can be selected one at a time. Peripheral functions are configured by writing the selected function value to the Peripheral Mux Registers (PMRn). To allow a peripheral pin access to the shared GPIO pin, GPIO control must be disabled for that pin, i.e. the corresponding bit in GPER must read zero.

A peripheral function value is set by writing bit zero to PMR0 and bit one to the same index position in PMR1 and so on. In a system with 4 peripheral functions A,B,C, and D, peripheral function C for GPIO pin four is selected by writing a zero to bit four in PMR0 and a one to the same bit index in PMR1. Refer to the GPIO Function Multiplexing chapter for details regarding pin function configuration for each GPIO pin.

23.6.2 Advanced Operation

23.6.2.1 Peripheral I/O Pin Control

When a GPIO pin is assigned to a peripheral function, i.e. the corresponding bit in GPER is zero, output and output enable is controlled by the selected peripheral pin. In addition the peripheral may control some or all of the other GPIO pin functions listed in Table 23-1, if the peripheral supports those features. All pin features not controlled by the selected peripheral is controlled by the GPIO.

Refer to the Module Configuration section for details regarding implemented GPIO pin functions and to the Peripheral chapter for details regarding I/O pin function control.

Table 23-1. I/O Pin function Control

Function name	GPIO mode	Peripheral mode
Output	OVR	Peripheral
Output enable	ODER	Peripheral
Pull-up	PUER	Peripheral if supported, else GPIO
Pull-down	PDER	Peripheral if supported, else GPIO
Drive strength	ODCRn	Peripheral if supported, else GPIO

23.6.2.2 Pull-up Resistor, Pull-down Resistor Control

Pull-up and pull-down can be configured for each GPIO pin. Pull-up allows the pin and any connected net to be pulled up to VDD if the net is not driven. Pull-down pulls the net to GND.

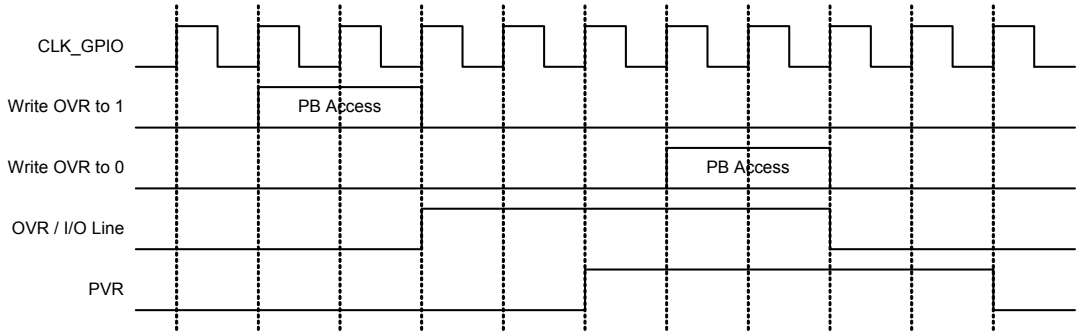
Pull-up and pull-down are useful for detecting if a pin is unconnected or if a mechanical button is pressed, for various communication protocols and to keep unconnected pins from floating.

Pull-up can be enabled and disabled by writing a one and a zero respectively to the corresponding bit in the Pull-up Enable Register (PUER). Pull-down can be enabled and disabled by writing a one and a zero respectively to the corresponding bit in the Pull-down Enable Register (PDER).

23.6.2.3 Output Pin Timings

Figure 23-3 shows the timing of the GPIO pin when writing to the Output Value Register (OVR). The same timing applies when performing a ‘set’ or ‘clear’ access, i.e. writing to OVRS or OVRC. The timing of PVR is also shown.

Figure 23-3. Output Pin Timings



23.6.2.4 Pin Output Driver Control

The GPIO has registers for controlling output drive properties of each pin, such as output driving capability.

The driving capability is controlled by the Output Driving Capability Registers (ODCRn).

23.6.2.5 Interrupts

The GPIO can be configured to generate an interrupt when it detects a change on a GPIO pin. Interrupts on a pin are enabled by writing a one to the corresponding bit in the Interrupt Enable Register (IER). The module can be configured to generate an interrupt whenever a pin changes value, or only on rising or falling edges. This is controlled by the Interrupt Mode Registers (IMRn). Interrupts on a pin can be enabled regardless of the GPIO pin being controlled by the GPIO or assigned to a peripheral function.

An interrupt can be generated on each GPIO pin. These interrupt generators are further grouped into groups of eight and connected to the interrupt controller. An interrupt request from any of the GPIO pin generators in the group will result in an interrupt request from that group to the interrupt controller if the corresponding bit for the GPIO pin in the IER is set. By grouping interrupt generators into groups of eight, four different interrupt handlers can be installed for each GPIO port.

The Interrupt Flag Register (IFR) can be read by software to determine which pin(s) caused the interrupt. The interrupt flag must be manually cleared by writing a zero to the corresponding bit in IFR.

GPIO interrupts will only be generated when CLK_GPIO is enabled.

23.6.2.6 Input Glitch Filter

Input glitch filters can be enabled on each GPIO pin. When the glitch filter is enabled, a glitch with duration of less than 1 CLK_GPIO cycle is automatically rejected, while a pulse with duration of 2 CLK_GPIO cycles or more is accepted. For pulse durations between 1 and 2 CLK_GPIO cycles, the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be guaranteed visible it must exceed 2 CLK_GPIO cycles, whereas for a glitch to be reliably filtered out, its duration must not exceed 1 CLK_GPIO cycle. The filter introduces 2 clock cycles latency.

The glitch filters are controlled by the Glitch Filter Enable Register (GFER). When a bit in GFER is one, the glitch filter on the corresponding pin is enabled. The glitch filter affects only interrupt inputs. Inputs to peripherals or the value read through PVR are not affected by the glitch filters.

23.6.2.7 Interrupt Timings

Figure 23-4 shows the timing for rising edge (or pin-change) interrupts when the glitch filter is disabled. For the pulse to be registered, it must be sampled at the rising edge of the clock. In this example, this is not the case for the first pulse. The second pulse is sampled on a rising edge and will trigger an interrupt request.

Figure 23-4. Interrupt Timing with Glitch Filter Disabled

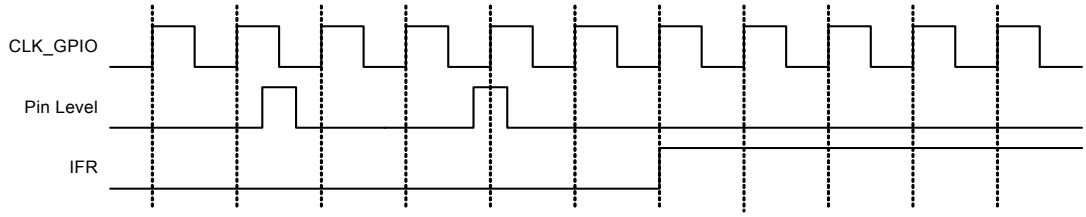
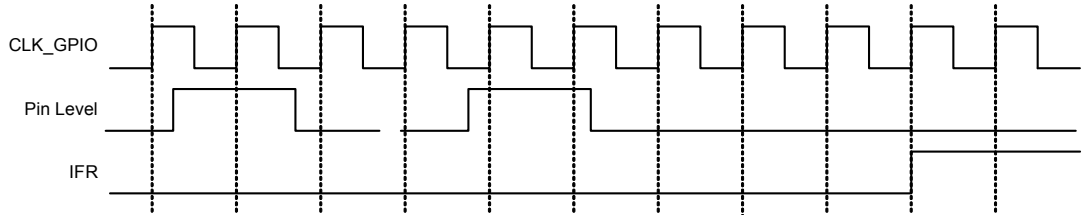


Figure 23-5 shows the timing for rising edge (or pin-change) interrupts when the glitch filter is enabled. For the pulse to be registered, it must be sampled on two subsequent rising edges. In the example, the first pulse is rejected while the second pulse is accepted and causes an interrupt request.

Figure 23-5. Interrupt Timing with Glitch Filter Enabled



23.6.2.8 CPU Local Bus

The CPU Local Bus can be used for application where low latency read and write access to the Output Value Register (OVR) and Output Drive Enable Register (ODER) is required. The CPU Local Bus allows the CPU to configure the mentioned GPIO registers directly, bypassing the shared Peripheral Bus (PB).

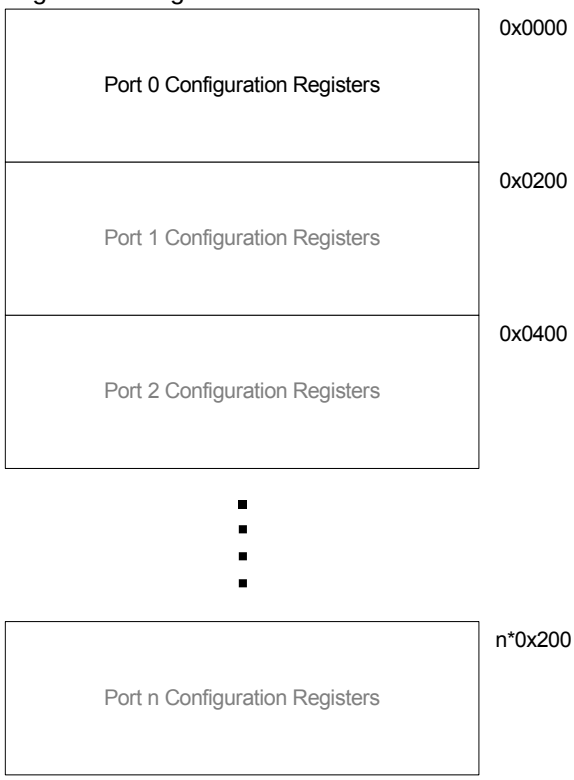
To avoid data loss when using the CPU Local Bus, the CLK_GPIO must run at the same frequency as the CLK_CPU. See Section 23.5.2 for details.

The CPU Local Bus is mapped to a different base address than the GPIO but the OVER and ODER offsets are the same. See the CPU Local Bus Mapping section in the Memories chapter for details.

23.7 User Interface

The GPIO controller manages all the GPIO pins on the 32-bit AVR microcontroller. The pins are managed as 32-bit ports that are configurable through a Peripheral Bus (PB) interface. Each port has a set of configuration registers. The overall memory map of the GPIO is shown below. The number of pins and hence the number of ports is product specific.

Figure 23-6. Port Configuration Registers



In the peripheral muxing table in the Package and Pinout chapter each GPIO pin has a unique number. Note that the PA, PB, PC, and PX ports do not necessarily directly correspond to the GPIO ports. To find the corresponding port and pin the following formulas can be used:

GPIO port = floor((GPIO number) / 32), example: floor((36)/32) = 1

GPIO pin = GPIO number % 32, example: 36 % 32 = 4

Table 23-2 shows the configuration registers for one port. Addresses shown are relative to the port address offset. The specific address of a configuration register is found by adding the register offset and the port offset to the GPIO start address. One bit in each of the configuration registers corresponds to a GPIO pin.

23.7.1 Access Types

Most configuration register can be accessed in four different ways. The first address location can be used to write the register directly. This address can also be used to read the register value. The following addresses facilitate three different types of write access to the register. Performing a “set” access, all bits written to one will be set. Bits written to zero will be unchanged by the operation. Performing a “clear” access, all bits written to one will be cleared. Bits written to zero will be unchanged by the operation. Finally, a toggle access will toggle the value of all bits writ-

ten to one. Again all bits written to zero remain unchanged. Note that for some registers (e.g. IFR), not all access methods are permitted.

Note that for ports with less than 32 bits, the corresponding control registers will have unused bits. This is also the case for features that are not implemented for a specific pin. Writing to an unused bit will have no effect. Reading unused bits will always return 0.

23.7.2 Configuration Protection

In order to protect the configuration of individual GPIO pins from software failure, configuration bits for individual GPIO pins may be locked by writing a one to the corresponding bit in the LOCK register. While this bit is one, any write to the same bit position in any lockable GPIO register using the Peripheral Bus (PB) will not have an effect. The CPU Local Bus is not checked and thus allowed to write to all bits in a CPU Local Bus mapped register no mather the LOCK value.

The registers required to clear bits in the LOCK register are protected by the access protection mechanism described in Section 23.7.3, ensuring the LOCK mechanism itself is robust against software failure.

23.7.3 Access Protection

In order to protect critical registers from software failure, some registers are protected by a key protection mechanism. These registers can only be changed by first writing the UNLOCK register, then the protected register. Protected registers are indicated in Table 23-2. The UNLOCK register contains a key field which must always be written to 0xAA, and an OFFSET field corresponding to the offset of the register to be modified.

The next write operation resets the UNLOCK register, so if the register is to be modified again, the UNLOCK register must be written again.

Attempting to write to a protected register without first writing the UNLOCK register results in the write operation being discarded, and the Access Error bit in the Access Status Register (ASR.AE) will be set.

Table 23-2. GPIO Register Memory Map

Offset	Register	Function	Register Name	Access	Reset	Config. Protection	Access Protection
0x000	GPIO Enable Register	Read/Write	GPER	Read/Write	-(1)	Y	N
0x004	GPIO Enable Register	Set	GPERS	Write-only		Y	N
0x008	GPIO Enable Register	Clear	GPERC	Write-only		Y	N
0x00C	GPIO Enable Register	Toggle	GPERT	Write-only		Y	N
0x010	Peripheral Mux Register 0	Read/Write	PMR0	Read/Write	-(1)	Y	N
0x014	Peripheral Mux Register 0	Set	PMR0S	Write-only		Y	N
0x018	Peripheral Mux Register 0	Clear	PMR0C	Write-only		Y	N
0x01C	Peripheral Mux Register 0	Toggle	PMR0T	Write-only		Y	N
0x020	Peripheral Mux Register 1	Read/Write	PMR1	Read/Write	-(1)	Y	N
0x024	Peripheral Mux Register 1	Set	PMR1S	Write-only		Y	N
0x028	Peripheral Mux Register 1	Clear	PMR1C	Write-only		Y	N

Table 23-2. GPIO Register Memory Map

Offset	Register	Function	Register Name	Access	Reset	Config. Protection	Access Protection
0x02C	Peripheral Mux Register 1	Toggle	PMR1T	Write-only		Y	N
0x030	Peripheral Mux Register 2	Read/Write	PMR2	Read/Write	-(1)	Y	N
0x034	Peripheral Mux Register 2	Set	PMR2S	Write-only		Y	N
0x038	Peripheral Mux Register 2	Clear	PMR2C	Write-only		Y	N
0x03C	Peripheral Mux Register 2	Toggle	PMR2T	Write-only		Y	N
0x040	Output Driver Enable Register	Read/Write	ODER	Read/Write	-(1)	Y	N
0x044	Output Driver Enable Register	Set	ODERS	Write-only		Y	N
0x048	Output Driver Enable Register	Clear	ODERC	Write-only		Y	N
0x04C	Output Driver Enable Register	Toggle	ODERT	Write-only		Y	N
0x050	Output Value Register	Read/Write	OVR	Read/Write	-(1)	N	N
0x054	Output Value Register	Set	OVRS	Write-only		N	N
0x058	Output Value Register	Clear	OVRC	Write-only		N	N
0x05c	Output Value Register	Toggle	OVRT	Write-only		N	N
0x060	Pin Value Register	Read	PVR	Read-only	Depending on pin states	N	N
0x064	Pin Value Register	-	-	-		N	N
0x068	Pin Value Register	-	-	-		N	N
0x06c	Pin Value Register	-	-	-		N	N
0x070	Pull-up Enable Register	Read/Write	PUER	Read/Write	-(1)	Y	N
0x074	Pull-up Enable Register	Set	PUERS	Write-only		Y	N
0x078	Pull-up Enable Register	Clear	PUERC	Write-only		Y	N
0x07C	Pull-up Enable Register	Toggle	PUERT	Write-only		Y	N
0x080	Pull-down Enable Register	Read/Write	PDER	Read/Write	(1)	Y	N
0x084	Pull-down Enable Register	Set	PDERS	Write-only		Y	N
0x088	Pull-down Enable Register	Clear	PDERC	Write-only		Y	N
0x08C	Pull-down Enable Register	Toggle	PDERT	Write-only		Y	N
0x090	Interrupt Enable Register	Read/Write	IER	Read/Write	-(1)	N	N
0x094	Interrupt Enable Register	Set	IERS	Write-only		N	N
0x098	Interrupt Enable Register	Clear	IERC	Write-only		N	N
0x09C	Interrupt Enable Register	Toggle	IERT	Write-only		N	N
0x0A0	Interrupt Mode Register 0	Read/Write	IMR0	Read/Write	-(1)	N	N
0x0A4	Interrupt Mode Register 0	Set	IMR0S	Write-only		N	N
0x0A8	Interrupt Mode Register 0	Clear	IMR0C	Write-only		N	N
0x0AC	Interrupt Mode Register 0	Toggle	IMR0T	Write-only		N	N

Table 23-2. GPIO Register Memory Map

Offset	Register	Function	Register Name	Access	Reset	Config. Protection	Access Protection
0x0B0	Interrupt Mode Register 1	Read/Write	IMR1	Read/Write	-(1)	N	N
0x0B4	Interrupt Mode Register 1	Set	IMR1S	Write-only		N	N
0x0B8	Interrupt Mode Register 1	Clear	IMR1C	Write-only		N	N
0x0BC	Interrupt Mode Register 1	Toggle	IMR1T	Write-only		N	N
0x0C0	Glitch Filter Enable Register	Read/Write	GFER	Read/Write	-(1)	N	N
0x0C4	Glitch Filter Enable Register	Set	GFERS	Write-only		N	N
0x0C8	Glitch Filter Enable Register	Clear	GFERC	Write-only		N	N
0x0CC	Glitch Filter Enable Register	Toggle	GFERT	Write-only		N	N
0x0D0	Interrupt Flag Register	Read	IFR	Read-only	-(1)	N	N
0x0D4	Interrupt Flag Register	-	-	-		N	N
0x0D8	Interrupt Flag Register	Clear	IFRC	Write-only		N	N
0x0DC	Interrupt Flag Register	-	-	-		N	N
0x100	Output Driving Capability Register 0	Read/Write	ODCR0	Read/Write	-(1)	Y	N
0x104	Output Driving Capability Register 0	Set	ODCR0S	Write-only		Y	N
0x108	Output Driving Capability Register 0	Clear	ODCR0C	Write-only		Y	N
0x10C	Output Driving Capability Register 0	Toggle	ODCR0T	Write-only		Y	N
0x110	Output Driving Capability Register 1	Read	ODCR1	Read/Write	-(1)	Y	N
0x114	Output Driving Capability Register 1	Set	ODCR1S	Write-only		Y	N
0x118	Output Driving Capability Register 1	Clear	ODCR1C	Write-only		Y	N
0x11C	Output Driving Capability Register 1	Toggle	ODCR1T	Write-only		Y	N
0x1A0	Lock Register	Read/Write	LOCK	Read/Write	-(1)	N	Y
0x1A4	Lock Register	Set	LOCKS	Write-only		N	N
0x1A8	Lock Register	Clear	LOCKC	Write-only		N	Y
0x1AC	Lock Register	Toggle	LOCKT	Write-only		N	Y
0x1E0	Unlock Register	Read/Write	UNLOCK	Write-only		N	N
0x1E4	Access Status Register	Read/Write	ASR	Read/Write			N
0x1F8	Parameter Register	Read	PARAMETER	Read-only	-(1)	N	N
0x1FC	Version Register	Read	VERSION	Read-only	-(1)	N	N

Note: 1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

23.7.4 GPIO Enable Register

Name: GPER

Access: Read/Write, Set, Clear, Toggle

Offset: 0x000, 0x004, 0x008, 0x00C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: GPIO Enable**
  0: A peripheral function controls the corresponding pin.
  1: The GPIO controls the corresponding pin.

23.7.5 Peripheral Mux Register 0

Name: PMR0

Access: Read/Write, Set, Clear, Toggle

Offset: 0x010, 0x014, 0x018, 0x01C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Peripheral Multiplexer Select bit 0**

23.7.6 Peripheral Mux Register 1

Name: PMR1

Access: Read/Write, Set, Clear, Toggle

Offset: 0x020, 0x024, 0x028, 0x02C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Peripheral Multiplexer Select bit 1**

23.7.7 Peripheral Mux Register 2

Name: PMR2

Access: Read/Write, Set, Clear, Toggle

Offset: 0x030, 0x034, 0x038, 0x03C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Peripheral Multiplexer Select bit 2**

{PMR2, PMR1, PMR0}	Selected Peripheral Function
000	A
001	B
010	C
011	D
100	E
101	F
110	G
111	H

23.7.8 Output Driver Enable Register

Name: ODER

Access: Read/Write, Set, Clear, Toggle

Offset: 0x040, 0x044, 0x048, 0x04C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Output Driver Enable**
  0: The output driver is disabled for the corresponding pin.
  1: The output driver is enabled for the corresponding pin.

23.7.9 Output Value Register

Name: OVR

Access: Read/Write, Set, Clear, Toggle

Offset: 0x050, 0x054, 0x058, 0x05C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Output Value**
  0: The value to be driven on the GPIO pin is 0.
  1: The value to be driven on the GPIO pin is 1.

23.7.10 Pin Value Register

Name: PVR

Access: Read-only

Offset: 0x060, 0x064, 0x068, 0x06C

Reset Value: Depending on pin states

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Pin Value**
  0: The GPIO pin is at level zero.
  1: The GPIO pin is at level one.
  Note that the level of a pin can only be read when the corresponding pin in GPER is one or interrupt is enabled for the pin.

23.7.11 Pull-up Enable Register

Name: PUER

Access: Read/Write, Set, Clear, Toggle

Offset: 0x070, 0x074, 0x078, 0x07C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Pull-up Enable**
  Writing a zero to a bit in this register will disable pull-up on the corresponding pin.
  Writing a one to a bit in this register will enable pull-up on the corresponding pin.

23.7.12 Pull-down Enable Register

Name: PDER

Access: Read/Write, Set, Clear, Toggle

Offset: 0x080, 0x084, 0x088, 0x08C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Pull-down Enable**

{PUER, PDER}	Selected Function
00	Disabled
01	Pull-down enabled
10	Pull-up enabled
11	Buskeeper enabled

23.7.13 Interrupt Enable Register

Name: IER

Access: Read/Write, Set, Clear, Toggle

Offset: 0x090, 0x094, 0x098, 0x09C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Interrupt Enable**
  0: Interrupt is disabled for the corresponding pin.
  1; Interrupt is enabled for the corresponding pin.

23.7.14 Interrupt Mode Register 0

Name: IMR0

Access: Read/Write, Set, Clear, Toggle

Offset: 0x0A0, 0x0A4, 0x0A8, 0x0AC

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Interrupt Mode Bit 0**

23.7.15 Interrupt Mode Register 1

Name: IMR1

Access: Read/Write, Set, Clear, Toggle

Offset: 0x0B0, 0x0B4, 0x0B8, 0x0BC

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Interrupt Mode Bit 1**

{IMR1, IMR0}	Interrupt Mode
00	Pin Change
01	Rising Edge
10	Falling Edge
11	Reserved

23.7.16 Glitch Filter Enable Register

Name: GFER

Access: Read/Write, Set, Clear, Toggle

Offset: 0x0C0, 0x0C4, 0x0C8, 0x0CC

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Glitch Filter Enable**
  0: Glitch filter is disabled for the corresponding pin.
  1: Glitch filter is enabled for the corresponding pin.
  NOTE! The value of this register should only be changed when the corresponding bit in IER is zero. Updating GFER while interrupt on the corresponding pin is enabled can cause an unintentional interrupt to be triggered.

23.7.17 Interrupt Flag Register

Name: IFR

Access: Read, Clear

Offset: 0x0D0, 0x0D8

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Interrupt Flag**

  0: No interrupt condition has been detected on the corresponding pin.

  1: An interrupt condition has been detected on the corresponding pin.

  The number of interrupt request lines depends on the number of GPIO pins on the MCU. Refer to the product specific data for details. Note also that a bit in the Interrupt Flag register is only valid if the corresponding bit in IER is one.

23.7.18 Output Driving Capability Register 0

Name: ODCR0

Access: Read/Write, Set, Clear, Toggle

Offset: 0x100, 0x104, 0x108, 0x10C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Output Driving Capability Register Bit 0**

23.7.19 Output Driving Capability Register 1

Name: ODCR1

Access: Read/Write, Set, Clear, Toggle

Offset: 0x110, 0x114, 0x118, 0x11C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Output Driving Capability Bit 1**

{ODCR1, ODCR0}	Interrupt Mode
00	Lowest drive strength
01	...
10	...
11	Highest drive strength

For the actual drive strength of the pin, please refer to the Electrical Characteristics chapter.

23.7.20 Lock Register

Name: LOCK

Access: Read/Write, Set, Clear, Toggle

Offset: 0x1A0, 0x1A4, 0x1A8, 0x1AC

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Lock State**

  0: Pin is unlocked. The corresponding bit can be changed in any GPIO register for this port.

  1: Pin is locked. The corresponding bit can not be changed in any GPIO register for this port.

  The value of LOCK determines which bits are locked in the lockable registers.

  The LOCK, LOCKC, and LOCKT registers are protected, which means they can only be written immediately after a write to the UNLOCK register with the proper KEY and OFFSET.

  LOCKS is not protected, and can be written at any time.

23.7.21 Unlock Register

Name: UNLOCK

Access: Write-only

Offset: 0x1E0

Reset Value: -

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

- **OFFSET: Register Offset**
  This field must be written with the offset value of the LOCK, LOCKC or LOCKT register to unlock. This offset must also include the port offset for the register to unlock. LOCKS can not be locked so no unlock is required before writing to this register.
- **KEY: Unlocking Key**
  This bitfield must be written to 0xAA for a write to this register to have an effect.

  This register always reads as zero.

23.7.22 Access Status Register

Name: ASR

Access: Read/Write

Offset: 0x1E4

Reset Value: -

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

- **AE: Access Error**
  This bit is set when a write to a locked register occurs.
  This bit can be written to 0 by software.

23.7.23 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x1F8

Reset Value: -

31	30	29	28	27	26	25	24
PARAMETER							

23	22	21	20	19	18	17	16
PARAMETER							

15	14	13	12	11	10	9	8
PARAMETER							

7	6	5	4	3	2	1	0
PARAMETER							

- **PARAMETER:**
  0: The corresponding pin is not implemented in this GPIO port.
  1: The corresponding pin is implemented in this GPIO port.
  There is one PARAMETER register per GPIO port. Each bit in the Parameter Register indicates whether the corresponding GPER bit is implemented.

23.7.24 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x1FC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

23.8 Module Configuration

The specific configuration for each GPIO instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Refer to the Power Manager chapter for details.

Table 23-3. Module Configuration

Feature	GPIO
Number of GPIO ports	4
Number of peripheral functions	4

Table 23-4. Implemented Pin Functions

Pin Function	Implemented	Notes
Pull-up	Yes	Controlled by PUER or peripheral
Pull-down	Yes	Controlled by PDER
Drive strength	Yes	Writing to ODCR0 control the drive strength of the pads Writing to ODCR1 has no effect
Slew rate	No	OSRRn registers are not implemented
Open Drain	No	ODMERn registers are not implemented
Bus keeper	No	Setting {PUER, PDER} to 0x3 in a pin does not enable the bus keeper on this pin

Table 23-5. Module Clock Name

Module name	Clock Name	Description
GPIO	CLK_GPIO	Peripheral Bus clock from the PBA clock domain

The reset values for all GPIO registers are zero, with the following exceptions:

Table 23-6. Register Reset Values

Port	Register	Reset Value
0	GPER	0x3FF9FFFF
0	PMR0	0x00000001
0	PMR1 - PMR2	0x00000000
0	ODER - OVR	0x00000000
0	PUER	0x00000001
0	PDER	0x00000000
0	IER - IMR0 - IMR1 - IFR	0x00000000
0	GFER	0x3FF9FFFF
0	ODCR0	0x00000000
0	LOCK	0x00000000
0	PARAMETER	0x3FF9FFFF

Table 23-6. Register Reset Values

Port	Register	Reset Value
0	VERSION	0x00000212
1	GPER	0xFFFFFFFF
1	PMR0	0x00000002
1	PMR1 - PMR2	0x00000000
1	ODER - OVR	0x00000000
1	PUER - PDER	0x00000000
1	IER - IMR0 - IMR1 - IFR	0x00000000
1	GFER	0xFFFFFFFF
1	ODCR0	0x00000000
1	LOCK	0x00000000
1	PARAMETER	0x3FFFFFFF
1	VERSION	0x00000212
2	GPER	0xFFFFFFFF
2	PMR0 - PMR1 - PMR2	0x00000000
2	ODER - OVR	0x00000000
2	PUER - PDER	0x00000000
2	IER - IMR0 - IMR1 - IFR	0x00000000
2	GFER	0xFFFFFFFF
2	ODCR0	0x00000000
2	LOCK	0x00000000
2	PARAMETER	0xFFFFFFFF
2	VERSION	0x00000212
3	GPER	0x7FFFFFFF
3	PMR0 - PMR1 - PMR2	0x00000000
3	ODER - OVR	0x00000000
3	PUER - PDER	0x00000000
3	IER - IMR0 - IMR1 - IFR	0x00000000
3	GFER	0x7FFFFFFF
3	ODCR0	0x00000000
3	LOCK	0x00000000
3	PARAMETER	0x7FFFFFFF
3	VERSION	0x00000212

24. Ethernet MAC (MACB)

Rev: 1.1.2.0

24.1 Features

- **Compatible with IEEE Standard 802.3**
- **10 and 100 Mbit/s Operation**
- **Full- and Half-duplex Operation**
- **Statistics Counter Registers**
- **MII/RMII Interface to the Physical Layer**
- **Interrupt Generation to Signal Receive and Transmit Completion**
- **DMA Master on Receive and Transmit Channels**
- **Transmit and Receive FIFOs**
- **Automatic Pad and CRC Generation on Transmitted Frames**
- **Automatic Discard of Frames Received with Errors**
- **Address Checking Logic Supports Up to Four Specific 48-bit Addresses**
- **Supports Promiscuous Mode Where All Valid Received Frames are Copied to Memory**
- **Hash Matching of Unicast and Multicast Destination Addresses**
- **External Address Matching of Received Frames**
- **Physical Layer Management through MDIO Interface**
- **Half-duplex Flow Control by Forcing Collisions on Incoming Frames**
- **Full-duplex Flow Control with Recognition of Incoming Pause Frames and Hardware Generation of Transmitted Pause Frames**
- **Support for 802.1Q VLAN Tagging with Recognition of Incoming VLAN and Priority Tagged Frames**
- **Multiple Buffers per Receive and Transmit Frame**
- **Wake-on-LAN Support**
- **Jumbo Frames Up to 10240 bytes Supported**

24.2 Overview

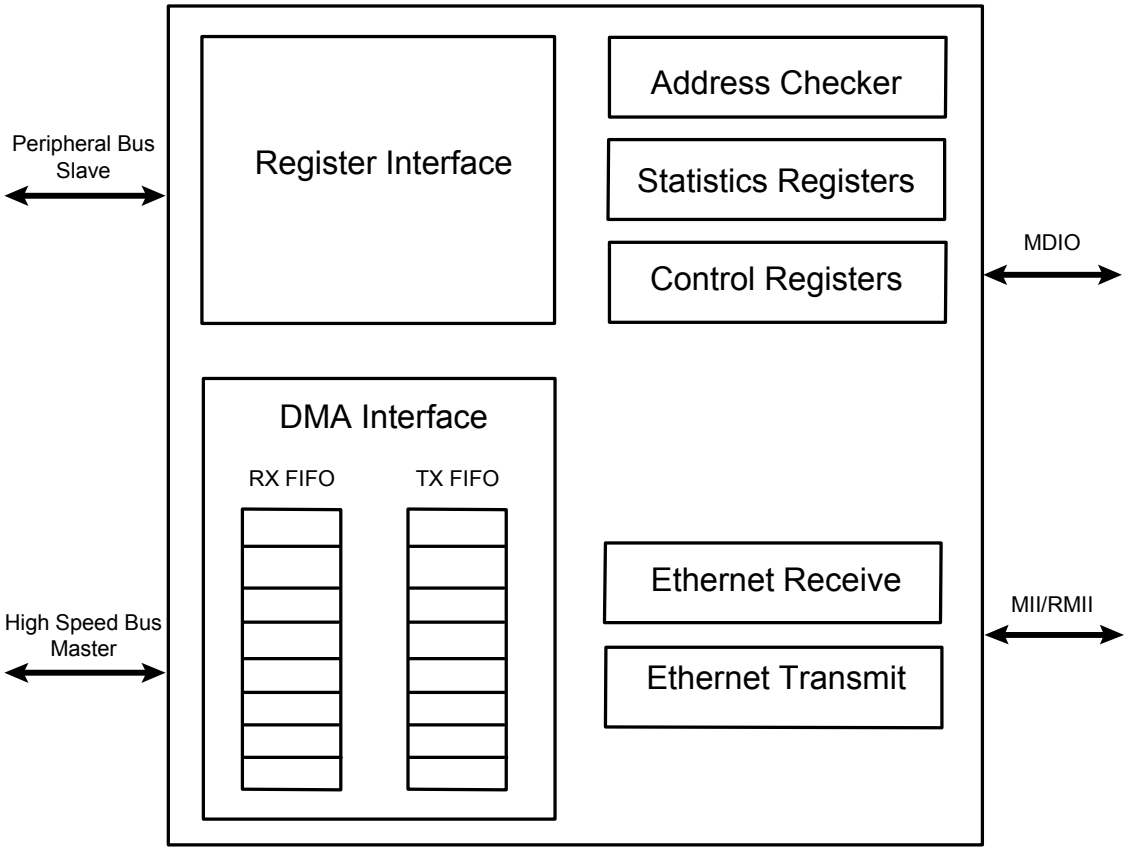
The MACB module implements a 10/100 Ethernet MAC compatible with the IEEE 802.3 standard using an address checker, statistics and control registers, receive and transmit submodules, and a DMA interface.

The address checker recognizes four specific 48-bit addresses and contains a 64-bit hash register for matching multicast and unicast addresses. It can recognize the broadcast address of all ones, copy all frames, and act on an external address match signal.

The statistics register sub-module contains registers for counting various types of events associated with transmit and receive operations. These registers, along with the status words stored in the receive buffer list, enable software to generate network management statistics compatible with IEEE 802.3.

24.3 Block Diagram

Figure 24-1. MACB Block Diagram



24.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

24.4.1 I/O Lines

The pins used for interfacing the MACB may be multiplexed with the I/O Controller lines. The programmer must first program the I/O Controller to assign the desired MACB pins to their peripheral function. If I/O lines of the MACB are not used by the application, they can be used for other purposes by the I/O Controller.

24.4.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the MACB, the MACB will stop functioning and resume operation after the system wakes up from sleep mode.

To prevent bus errors the MACB operation must be terminated before entering sleep mode.

24.4.3 Clocks

The clocks for the MACB bus interface (CLK_MACB_PB/CLK_MACB_HSB) are generated by the Power Manager. These clocks are enabled at reset, and can be disabled in the Power Man-

ager. It is recommended to disable the MACB before disabling the clocks, to avoid freezing the MACB in an undefined state.

The synchronization module in the MACB requires that the bus clock (CLK_MACB_HSB) runs on at least the speed of the macb_tx/RX_CLK, which is 25MHz in 100Mbps, and 2.5MHZ in 10Mbps in MII mode and 50MHz in 100Mbps, and 5MHZ in 10Mbps in RMII mode.

24.4.4 Interrupt

The MACB interrupt request line is connected to the interrupt controller. Using the MACB interrupt requires the interrupt controller to be programmed first.

24.4.5 Debug Operation

When an external debugger forces the CPU into debug mode, the MACB continues normal operation. If the MACB is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

24.5 Functional Description

The control registers drive the MDIO interface, setup DMA activity, start frame transmission and select modes of operation such as full or half-duplex.

The receive sub-module checks for valid preamble, Frame Check Sequence (FCS), alignment and length, and presents received frames to the address checking sub-module and DMA interface.

The transmit sub-module takes data from the DMA interface, adds preamble and, if necessary, pad and FCS, and transmits data according to the Carrier Sense Multiple Access with Collision Detect (CSMA/CD) protocol. The start of transmission is deferred if Carrier Sense (CRS) is active.

If Collision (COL) becomes active during transmission, a jam sequence is asserted and the transmission is retried after a random back off. CRS and COL have no effect in full duplex mode.

The DMA interface can access external memory through its High Speed Bus (HSB). It contains receive and transmit FIFOs for buffering frame data. It loads the transmit FIFO and empties the receive FIFO using HSB bus master operations. Received data is not sent to memory until the address checking logic has determined that the frame should be copied. Received or transmitted frames are stored in one or more buffers. Receive buffers have a fixed length of 128 bytes. Transmit buffers range in length between 0 and 2047 bytes, and up to 128 buffers are allowed per frame. The DMA interface manages the transmit and receive frame buffer queues. These queues can hold multiple frames.

24.5.1 Memory Interface

Frame data is transferred to and from the MACB by the DMA interface. All transfers are 32-bit words and may be single accesses or bursts of 2, 3 or 4 words. Burst accesses do not cross sixteen-byte boundaries. Bursts of 4 words are the default data transfer; single accesses or bursts of less than four words may be used to transfer data at the beginning or the end of a buffer.

The DMA interface performs six types of operation on the bus. In order of priority, these are:

1. Receive buffer manager write
2. Receive buffer manager read

3. Transmit data DMA read
4. Receive data DMA write
5. Transmit buffer manager read
6. Transmit buffer manager write

24.5.2 FIFO

FIFO depths are 124 bytes.

Data is typically transferred in and out of the FIFOs in bursts of four words. In reception, a bus request is asserted when the FIFO contains four words and has space for three more. For transmission, a bus request is generated when there is space for four words, or when there is space for two words if the next transfer is only one or two words.

Thus the bus latency is less than the time it takes to load the FIFO and transmit or receive three words (12 bytes) of data.

At 100 Mbit/s, it takes 960 ns to transmit or receive 12 bytes of data. In addition, six PB clock cycles should be allowed for data to be loaded from the bus and to propagate through the FIFOs. For a 60 MHz PB clock this takes 100 ns, making the bus latency requirement 860 ns.

24.5.3 Receive Buffers

Received frames, optionally including CRC/FCS, are written to receive buffers stored in memory. Each receive buffer size is 128 bytes. The start location for each receive buffer is stored in memory in a list of receive buffer descriptors at a location pointed to by the receive buffer queue pointer register. The receive buffer start location is a word address. For the first buffer of a frame, the start location can be offset by up to three bytes depending on the value written to bits 14 and 15 of the network configuration register. If the start location of the buffer is offset the available length of the first buffer of a frame is reduced by the corresponding number of bytes.

Each list entry consists of two words, the first being the address of the receive buffer and the second being the receive status. If the length of a receive frame exceeds the buffer length, the status word for the used buffer is written with zeroes except for the “start of frame” bit and the offset field, if appropriate. Bit zero of the address field is written to one to show the buffer has been used. The receive buffer manager then reads the location of the next receive buffer and fills that with receive frame data. The final buffer descriptor status word contains the complete frame status. Refer to Table 24-1 for details of the receive buffer descriptor list.

Table 24-1. Receive Buffer Descriptor Entry

Bit	Function
Word 0	
31:2	Address of beginning of buffer
1	Wrap - marks last descriptor in receive buffer descriptor list.
0	Ownership - needs to be zero for the MACB to write data to the receive buffer. The MACB sets this to one once it has successfully written a frame to memory. Software has to clear this bit before the buffer can be used again.
Word 1	
31	Global all ones broadcast address detected
30	Multicast hash match
29	Unicast hash match

Table 24-1. Receive Buffer Descriptor Entry (Continued)

Bit	Function
28	External address match
27	Reserved for future use
26	Specific address register 1 match
25	Specific address register 2 match
24	Specific address register 3 match
23	Specific address register 4 match
22	Type ID match
21	VLAN tag detected (i.e., type id of 0x8100)
20	Priority tag detected (i.e., type id of 0x8100 and null VLAN identifier)
19:17	VLAN priority (only valid if bit 21 is set)
16	Concatenation format indicator (CFI) bit (only valid if bit 21 is set)
15	End of frame - when set the buffer contains the end of a frame. If end of frame is not set, then the only other valid status are bits 12, 13 and 14.
14	Start of frame - when set the buffer contains the start of a frame. If both bits 15 and 14 are set, then the buffer contains a whole frame.
13:12	Receive buffer offset - indicates the number of bytes by which the data in the first buffer is offset from the word address. Updated with the current values of the network configuration register. If jumbo frame mode is enabled through bit 3 of the network configuration register, then bits 12 and 13 of the receive buffer descriptor entry are used to indicate bits 12 and 13 of the frame length.
11:0	Length of frame including FCS (if selected). Bits 12 and 13 are also used if jumbo frame mode is selected.

To receive frames, the buffer descriptors must be initialized by writing the right address to bits 2 through 31in the first word of each list entry. Bit zero must be written with zero. Bit one is the wrap bit and indicates the last entry in the list.

The start location of the receive buffer descriptor list must be written to the receive buffer queue pointer register before setting the receive enable bit in the network control register to enable receive. As soon as the receive sub-module starts writing received frame data to the receive FIFO, the receive buffer manager reads the first receive buffer location pointed to by the receive buffer queue pointer register.

If the filter sub-module indicates that the frame should be copied to memory, the receive data DMA operation starts writing data into the receive buffer. If an error occurs, the buffer is recovered. If the current buffer pointer has its wrap bit set or is the 1024th descriptor, the next receive buffer location is read from the beginning of the receive descriptor list. Otherwise, the next receive buffer location is read from the next word in memory.

There is an 11-bit counter to count out the 2048 word locations of a maximum length, receive buffer descriptor list. This is added with the value originally written to the receive buffer queue pointer register to produce a pointer into the list. A read of the receive buffer queue pointer register returns the pointer value, which is the queue entry currently being accessed. The counter is reset after receive status is written to a descriptor that has its wrap bit set or rolls over to zero after 1024 descriptors have been accessed. The value written to the receive buffer pointer register may be any word-aligned address, provided that there are at least 2048 word locations available between the pointer and the top of the memory.

The System Bus specification requires that bursts should not cross 1K boundaries. As receive buffer manager writes are bursts of two words, to ensure that this does not occur, it is best to write the pointer register with the least three significant bits set to zero. As receive buffers are used, the receive buffer manager sets bit zero of the first word of the descriptor to indicate *used*. If a receive error is detected the receive buffer currently being written is recovered. Previous buffers are not recovered. Software should search through the used bits in the buffer descriptors to find out how many frames have been received. It should be checking the start-of-frame and end-of-frame bits, and not rely on the value returned by the receive buffer queue pointer register which changes continuously as more buffers are used.

For CRC errored frames, excessive length frames or length field mismatched frames, all of which are counted in the statistics registers, it is possible that a frame fragment might be stored in a sequence of receive buffers. Software can detect this by looking for start of frame bit set in a buffer following a buffer with no end of frame bit set.

For a properly working Ethernet system, there should be no excessively long frames or frames greater than 128 bytes with CRC/FCS errors. Collision fragments are less than 128 bytes long. Therefore, it is a rare occurrence to find a frame fragment in a receive buffer.

If bit zero is set when the receive buffer manager reads the location of the receive buffer, then the buffer has already been used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the DMA interface sets the buffer not available bit in the receive status register and triggers an interrupt.

If bit zero is set when the receive buffer manager reads the location of the receive buffer and a frame is being received, the frame is discarded and the receive resource error statistics register is incremented.

A receive overrun condition occurs when bus was not granted in time or because HRESP was not OK (bus error). In a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame received with an address that is recognized reuses the buffer.

If bit 17 of the network configuration register is set, the FCS of received frames shall not be copied to memory. The frame length indicated in the receive status field shall be reduced by four bytes in this case.

24.5.4 Transmit Buffer

Frames to be transmitted are stored in one or more transmit buffers. Transmit buffers can be between 0 and 2047 bytes long, so it is possible to transmit frames longer than the maximum length specified in IEEE Standard 802.3. Zero length buffers are allowed. The maximum number of buffers permitted for each transmit frame is 128.

The start location for each transmit buffer is stored in memory in a list of transmit buffer descriptors at a location pointed to by the transmit buffer queue pointer register. Each list entry consists of two words, the first being the byte address of the transmit buffer and the second containing the transmit control and status. Frames can be transmitted with or without automatic CRC generation. If CRC is automatically generated, padding is also automatically generated to take frames to a minimum length of 64 bytes. Table 24-2 on page 494 defines an entry in the transmit buffer descriptor list. To transmit frames, the buffer descriptors must be initialized by writing the right byte address to bits 0 through 31 in the first word of each list entry. The second transmit buffer descriptor is initialized with control information that indicates the length of the buffer,

whether or not it is to be transmitted with CRC and whether the buffer is the last buffer in the frame.

After transmission, the control bits are written back to the second word of the first buffer along with the “used” bit and other status information. Before a transmission, bit 31 is the “used” bit which must be zero when the control word is read. It is written to one when a frame has been transmitted. Bits 27, 28 and 29 indicate various transmit error conditions. Bit 30 is the “wrap” bit which can be set for any buffer within a frame. If no wrap bit is encountered after 1024 descriptors, the queue pointer rolls over to the start.

The transmit buffer queue pointer register must not be written while transmit is active. If a new value is written to the transmit buffer queue pointer register, the queue pointer resets itself to point to the beginning of the new queue. If transmit is disabled by writing to bit 3 of the network control, the transmit buffer queue pointer register resets to point to the beginning of the transmit queue. Note that disabling receive does not have the same effect on the receive queue pointer.

Once the transmit queue is initialized, transmit is activated by writing to bit 9, the *Transmit Start* bit of the network control register. Transmit is halted when a buffer descriptor with its *used* bit set is read, or if a transmit error occurs, or by writing to the transmit halt bit of the network control register. (Transmission is suspended if a pause frame is received while the pause enable bit is set in the network configuration register.) Rewriting the start bit while transmission is active is allowed.

Transmission control is implemented with a Tx_go variable which is readable in the transmit status register at bit location 3. The Tx_go variable is reset when:

– transmit is disabled

– a buffer descriptor with its ownership bit set is read

– a new value is written to the transmit buffer queue pointer register

– bit 10, tx_halt, of the network control register is written

– there is a transmit error such as too many retries or a transmit underrun.

To set tx_go, write to bit 9, tx_start, of the network control register. Transmit halt does not take effect until any ongoing transmit finishes. If a collision occurs during transmission of a multi-buffer frame, transmission automatically restarts from the first buffer of the frame. If a “used” bit is read midway through transmission of a multi-buffer frame, this is treated as a transmit error. Transmission stops, TX_ER is asserted and the FCS is bad.

If transmission stops due to a transmit error, the transmit queue pointer resets to point to the beginning of the transmit queue. Software needs to re-initialize the transmit queue after a transmit error.

If transmission stops due to a “used” bit being read at the start of the frame, the transmission queue pointer is not reset and transmit starts from the same transmit buffer descriptor when the transmit start bit is written

Table 24-2. Transmit Buffer Descriptor Entry

Bit	Function
Word 0	
31:0	**Byte Address of buffer**
Word 1	

Table 24-2. Transmit Buffer Descriptor Entry (Continued)

Bit	Function
31	Used. Needs to be zero for the MACB to read data from the transmit buffer. The MACB sets this to one for the first buffer of a frame once it has been successfully transmitted. Software has to clear this bit before the buffer can be used again. Note: This bit is only set for the first buffer in a frame unlike receive where all buffers have the Used bit set once used.
30	Wrap. Marks last descriptor in transmit buffer descriptor list.
29	Retry limit exceeded, transmit error detected
28	Transmit underrun, occurs either when hresp is not OK (bus error) or the transmit data could not be fetched in time or when buffers are exhausted in mid frame.
27	Buffers exhausted in mid frame
26:17	Reserved
16	No CRC. When set, no CRC is appended to the current frame. This bit only needs to be set for the last buffer of a frame.
15	Last buffer. When set, this bit indicates the last buffer in the current frame has been reached.
14:11	Reserved
10:0	Length of buffer

24.5.5 Transmit Sub-module

This sub-module transmits frames in accordance with the Ethernet IEEE 802.3 CSMA/CD protocol. Frame assembly starts by adding preamble and the start frame delimiter. Data is taken from the transmit FIFO a word at a time. Data is transmitted least significant nibble first. If necessary, padding is added to increase the frame length to 60 bytes. CRC is calculated as a 32-bit polynomial. This is inverted and appended to the end of the frame, taking the frame length to a minimum of 64 bytes. If the No CRC bit is set in the second word of the last buffer descriptor of a transmit frame, neither pad nor CRC are appended.

In full-duplex mode, frames are transmitted immediately. Back-to-back frames are transmitted at least 96 bit times apart to guarantee the interframe gap.

In half-duplex mode, the transmitter checks carrier sense. If asserted, it waits for it to de-assert and then starts transmission after the interframe gap of 96 bit times. If the collision signal is asserted during transmission, the transmitter transmits a jam sequence of 32 bits taken from the data register and retries transmission after the back off time has elapsed.

The back-off time is based on an XOR of the 10 least significant bits of the data coming from the transmit FIFO and a 10-bit pseudo random number. The number of bits used depends on the number of collisions seen. After the first collision, 1 bit is used, after the second 2, and so on up to 10. Above 10, all 10 bits are used. An error is indicated and no further attempts are made if 16 attempts cause collisions.

If transmit DMA underruns, bad CRC is automatically appended using the same mechanism as jam insertion and TX_ER is asserted. In a properly configured system, this should never happen.

If the back pressure bit is set in the network control register in half duplex mode, the transmit sub-module transmits 64 bits of data, which can consist of 16 nibbles of 1011 or in bit-rate mode

64 1s, whenever it sees an incoming frame to force a collision. This provides a way of implementing flow control in half-duplex mode.

24.5.6 Pause Frame Support

The start of an 802.3 pause frame is as follows:

Table 24-3. Start of an 802.3 Pause Frame

Destination Address	Source Address	Type (Mac Control Frame)	Pause Opcode	Pause Time
0x0180C2000001	6 bytes	0x8808	0x0001	2 bytes

The network configuration register contains a receive pause enable bit (13). If a valid pause frame is received, the pause time register is updated with the frame's pause time, regardless of its current contents and regardless of the state of the configuration register bit 13. An interrupt (12) is triggered when a pause frame is received, assuming it is enabled in the interrupt mask register. If bit 13 is set in the network configuration register and the value of the pause time register is non-zero, no new frame is transmitted until the pause time register has decremented to zero.

The loading of a new pause time, and hence the pausing of transmission, only occurs when the MACB is configured for full-duplex operation. If the MACB is configured for half-duplex, there is no transmission pause, but the pause frame received interrupt is still triggered.

A valid pause frame is defined as having a destination address that matches either the address stored in specific address register 1 or matches 0x0180C2000001 and has the MAC control frame type ID of 0x8808 and the pause opcode of 0x0001. Pause frames that have FCS or other errors are treated as invalid and are discarded. Valid pause frames received increment the Pause Frame Received statistic register.

The pause time register decrements every 512 bit times (i.e., 128 RX_CLK in nibble mode) once transmission has stopped. For test purposes, the register decrements every RX_CLK cycle once transmission has stopped if bit 12 (retry test) is set in the network configuration register. If the pause enable bit (13) is not set in the network configuration register, then the decrementing occurs regardless of whether transmission has stopped or not.

An interrupt (13) is asserted whenever the pause time register decrements to zero (assuming it is enabled in the interrupt mask register). Automatic transmission of pause frames is supported through the transmit pause frame bits of the network control register and the tx_pause and tx_pause_zero inputs. If either bit 11 or bit 12 of the network control register is written to with a 1, or if the input signal tx_pause is toggled, a pause frame is transmitted only if full duplex is selected in the network configuration register and transmit is enabled in the network control register.

Pause frame transmission occurs immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted. The transmitted pause frame is comprised of the items in the following list:

- a destination address of 01-80-C2-00-00-01
- a source address taken from the specific address 1 register
- a type ID of 88-08 (MAC control frame)
- a pause opcode of 00-01
- a pause quantum

- fill of 00 to take the frame to minimum frame length
- valid FCS

The pause quantum used in the generated frame depends on the trigger source for the frame as follows:

1. If bit 11 is written with a one, the pause quantum comes from the transmit pause quantum register. The Transmit Pause Quantum register resets to a value of 0xFFFF giving a maximum pause quantum as a default.
2. If bit 12 is written with a one, the pause quantum is zero.
3. If the tx_pause input is toggled and the tx_pause_zero input is held low until the next toggle, the pause quantum comes from the transmit pause quantum register.
4. If the tx_pause input is toggled and the tx_pause_zero input is held high until the next toggle, the pause quantum is zero.

After transmission, no interrupts are generated and the only statistics register that is incremented is the pause frames transmitted register.

24.5.7 Receive Sub-module

The receive sub-module checks for valid preamble, FCS, alignment and length, presents received frames to the DMA interface and stores the frames destination address for use by the address checking sub-module. If, during frame reception, the frame is found to be too long or RX_ER is asserted, a bad frame indication is sent to the DMA interface. The DMA interface then stops sending data to memory. At the end of frame reception, the receive sub-module indicates to the DMA interface whether the frame is good or bad. The DMA interface recovers the current receive buffer if the frame was bad. The receive sub-module signals the register sub-module to increment the alignment error, the CRC (FCS) error, the short frame, long frame, jabber error, the receive symbol error statistics and the length field mismatch statistics.

The enable bit for jumbo frames in the network configuration register allows the MACB to receive jumbo frames of up to 10240 bytes in size. This operation does not form part of the IEEE802.3 specification and is disabled by default. When jumbo frames are enabled, frames received with a frame size greater than 10240 bytes are discarded.

24.5.8 Address Checking Sub-module

The address checking (or filter) sub-module indicates to the DMA interface which receive frames should be copied to memory. Whether a frame is copied depends on what is enabled in the network configuration register, the state of the external match pin, the contents of the specific address and hash registers and the frame's destination address. In this implementation of the MACB, the frame's source address is not checked. If bit 18 of the Network Configuration register is not set, a frame is not copied to memory if the MACB is transmitting in half duplex mode at the time a destination address is received. If bit 18 of the Network Configuration register is set, frames can be received while transmitting in half-duplex mode.

Ethernet frames are transmitted a byte at a time, least significant bit first. The first six bytes (48 bits) of an Ethernet frame make up the destination address. The first bit of the destination address, the LSB of the first byte of the frame, is the group/individual bit: this is One for multicast addresses and Zero for unicast. The All Ones address is the broadcast address, and a special case of multicast.

The MACB supports recognition of four specific addresses. Each specific address requires two registers, specific address register bottom and specific address register top. Specific address

register bottom stores the first four bytes of the destination address and specific address register top contains the last two bytes. The addresses stored can be specific, group, local or universal.

The destination address of received frames is compared against the data stored in the specific address registers once they have been activated. The addresses are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. If a receive frame address matches an active address, the frame is copied to memory.

The following example illustrates the use of the address match registers for a MAC address of 21:43:65:87:A9:CB.

Preamble 55

SFD D5

DA (Octet0 - LSB) 21

DA(Octet 1) 43

DA(Octet 2) 65

DA(Octet 3) 87

DA(Octet 4) A9

DA (Octet5 - MSB) CB

SA (LSB) 00

SA 00

SA 00

SA 00

SA 00

SA (MSB) 43

SA (LSB) 21

The sequence above shows the beginning of an Ethernet frame. Byte order of transmission is from top to bottom as shown. For a successful match to specific address 1, the following address matching registers must be set up:

- Base address + 0x98 0x87654321 (Bottom)
- Base address + 0x9C 0x0000CBA9 (Top)

And for a successful match to the Type ID register, the following should be set up:

- Base address + 0xB8 0x00004321

24.5.9 Broadcast Address

The broadcast address of 0xFFFFFFFFFFFF is recognized unless the ‘no broadcast’ bit in the network configuration register is set.

24.5.10 Hash Addressing

The hash address register is 64 bits long and takes up two locations in the memory map. The least significant bits are stored in hash register bottom and the most significant bits in hash register top.

The unicast hash enable and the multicast hash enable bits in the network configuration register enable the reception of hash matched frames. The destination address is reduced to a 6-bit index into the 64-bit hash register using the following hash function. The hash function is an *exclusive or* of every sixth bit of the destination address.

hash_index[5] = da[5] ^ da[11] ^ da[17] ^ da[23] ^ da[29] ^ da[35] ^ da[41] ^ da[47]

hash_index[4] = da[4] ^ da[10] ^ da[16] ^ da[22] ^ da[28] ^ da[34] ^ da[40] ^ da[46]

hash_index[3] = da[3] ^ da[09] ^ da[15] ^ da[21] ^ da[27] ^ da[33] ^ da[39] ^ da[45]

hash_index[2] = da[2] ^ da[08] ^ da[14] ^ da[20] ^ da[26] ^ da[32] ^ da[38] ^ da[44]

hash_index[1] = da[1] ^ da[07] ^ da[13] ^ da[19] ^ da[25] ^ da[31] ^ da[37] ^ da[43]

hash_index[0] = da[0] ^ da[06] ^ da[12] ^ da[18] ^ da[24] ^ da[30] ^ da[36] ^ da[42]

`da[0]` represents the least significant bit of the first byte received, that is, the multicast/unicast indicator, and `da[47]` represents the most significant bit of the last byte received.

If the hash index points to a bit that is set in the hash register, then the frame is matched according to whether the frame is multicast or unicast.

A multicast match is signalled if the multicast hash enable bit is set. da[0] is 1 and the hash index points to a bit set in the hash register.

A unicast match is signalled if the unicast hash enable bit is set. da[0] is 0 and the hash index points to a bit set in the hash register.

To receive all multicast frames, the hash register should be set with all ones and the multicast hash enable bit should be set in the network configuration register.

24.5.11 External Address Matching

The external address signal (eam) is enabled by bit 9 in the network configuration register. When enabled, the filter sub-module sends the store frame and the external address match status signal to the DMA interface if the external address match signal is asserted (from a source external to the MACB) and the destination address has been received and the frame has not completed.

For the DMA interface to be able to copy the frame to memory, the external address signal must be asserted before four words have been loaded into the receive FIFO.

24.5.12 Copy All Frames (or Promiscuous Mode)

If the copy all frames bit is set in the network configuration register, then all non-errored frames are copied to memory. For example, frames that are too long, too short, or have FCS errors or RX_ER asserted during reception are discarded and all others are received. Frames with FCS errors are copied to memory if bit 19 in the network configuration register is set.

24.5.13 Type ID Checking

The contents of the type_id register are compared against the length/type ID of received frames (i.e., bytes 13 and 14). Bit 22 in the receive buffer descriptor status is set if there is a match. The reset state of this register is zero which is unlikely to match the length/type ID of any valid Ethernet frame.

Note: A type ID match does not affect whether a frame is copied to memory.

24.5.14 VLAN Support

An Ethernet encoded 802.1Q VLAN tag looks like this:

Table 24-4. 802.1Q VLAN Tag

TPID (Tag Protocol Identifier) 16 bits	TCI (Tag Control Information) 16 bits
0x8100	First 3 bits priority, then CFI bit, last 12 bits VID

The VLAN tag is inserted at the 13th byte of the frame, adding an extra four bytes to the frame. If the VID (VLAN identifier) is null (0x000), this indicates a priority-tagged frame. The MAC can support frame lengths up to 1536 bytes, 18 bytes more than the original Ethernet maximum frame length of 1518 bytes. This is achieved by setting bit 8 in the network configuration register.

The following bits in the receive buffer descriptor status word give information about VLAN tagged frames:

- Bit 21 set if receive frame is VLAN tagged (i.e. type id of 0x8100)
- Bit 20 set if receive frame is priority tagged (i.e. type id of 0x8100 and null VID). (If bit 20 is set bit 21 is set also.)
- Bit 19, 18 and 17 set to priority if bit 21 is set
- Bit 16 set to CFI if bit 21 is set

24.5.15 Wake-on LAN Support

The receive module supports wake-on LAN by detecting the following events on incoming receive frames:

- Magic packet
- ARP request to the device IP address
- Specific address 1 filter match
- Multicast hash filter match

If one of these events occurs wake-on LAN detection is indicated by asserting WOL output pin for 64 RX_CLK cycles. These events can be individually enabled by bits MAG, ARP, SA1 & MTI in wake-on LAN register (WOL). Also, for wake-on LAN detection to occur, receive enable must be set in the network control register (NCR), however a receive buffer does not have to be available.

WOL assertion due to ARP request, specific address 1 or multicast filter events will occur even if the frame is errored. For magic packet event, the frame must be correctly formed and error free.

A magic packet event is detected if all of the following are true:

- magic packet events are enabled by WOL.MAG bit
- the frame's destination address matches specific address 1
- the frame is correctly formed with no errors
- the frame contains at least 6 bytes of 0xFF for synchronization
- there are 16 repetitions of the contents of specific address 1 register immediately following the synchronization

An ARP packet event is detected if all of the following are true:

- ARP request are enabled by WOL.ARP bit
- broadcasts are allowed by NCFG.CAF

- the frame has a broadcast destination address (bytes 1 to 6)
- the frame has a typeID field of 0x0806 (bytes 13 and 14)
- the frame has an ARP operation field of 0x0001 (bytes 21 and 22)
- the least significant 16 bits of the frame ARP target protocol (bytes 41 and 42) match the value written in WOL.IP.

The decoding of the ARP fields adjusts automatically if a VLAN tag is detected within the frame. The reserved value of 0x0000 for wake-on LAN target address value will not cause an ARP request event, even if matched by the frame.

A specific address 1 filter match event will occur if all of the following are true:

- specific address 1 events are enabled by WOL.SA1 bit
- the frame destination address matches the value programmed in the specific address 1 registers

A multicast filter match event will occur if all of the following are true:

- multicast hash events are enabled by WOL.MTI bit
- multicast hash filtering is enabled by NCFG.MTI bit
- the frame destination address matches against the multicast hash filter
- the frame destination address is not a broadcast

24.5.16 PHY Maintenance

The register MAN enables the MACB to communicate with a PHY by means of the MDIO interface. It is used during auto-negotiation to ensure that the MACB and the PHY are configured for the same speed and duplex configuration.

The PHY maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit two is set in the network status register (about 2000 MCK cycles later when bit ten is set to zero, and bit eleven is set to one in the network configuration register). An interrupt is generated as this bit is set. During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each Divided PB Clock (DPC) cycle. This causes transmission of a PHY management frame on MDIO.

Reading during the shift operation returns the current contents of the shift register. At the end of management operation, the bits have shifted back to their original locations. For a read operation, the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

The MDIO interface can read IEEE 802.3 clause 45 PHYs as well as clause 22 PHYs. To read clause 45 PHYs, bits [31:28] should be written to 0x0011. For a description of DPC generation, see the network configuration register in section ”Network Configuration Register” on page 510.

24.5.17 Media Independent Interface

The Ethernet MAC is capable of interfacing to both RMII and MII Interfaces. The RMII bit in the USRIO register controls the interface that is selected. When this bit is set, the RMII interface is selected, else the MII interface is selected.

The MII and RMII interface are capable of both 10Mb/s and 100Mb/s data rates as described in the IEEE 802.3u standard. The signals used by the MII and RMII interfaces are described in Table 24-5.

Table 24-5. Pin Configuration

Pin Name	MII	RMII
TX_CLK	Transmit Clock	Reference Clock
CRS	Carrier Sense	
COL	Collision Detect	
RX_DV	Data Valid	Carrier Sense/Data Valid
RXD[3:0]	RXS[3:0] 4-bit Receive Data	RXD[1:0] 2-bit Receive Data
RX_ER	Receive Error	Receive Error
RX_CLK	Receive Clock	
TX_EN	Transmit Enable	Transmit Enable
TXD[3:0]	TXD[3:0] 4-bit Transmit Data	TXD[1:0] 2-bit Transmit Data
TX_ER	Transmit Error	

The intent of the RMII is to provide a reduced pin count alternative to the IEEE 802.3u MII. It uses 2 bits for transmission (TXD[1:0]) and two bits for reception (RXD[1:0]). There are Transmit Enable (TX_EN), a Receive Error (RX_ER), a Carrier Sense (CRS), and a 50 MHz Reference Clock (TX_CLK) for 100Mb/s data rate.

24.5.17.1 RMII Transmit and Receive Operation

The same signals are used internally for both the RMII and the MII operations. The RMII maps these signals in a more pin-efficient manner. The transmit and receive bits are converted from a 4-bit parallel format to a 2-bit parallel scheme that is clocked at twice the rate. The carrier sense and data valid signals are combined into the RX_DV signal. This signal contains information on carrier sense, FIFO status, and validity of the data. Transmit error bit (TX_ER) and collision detect (COL) are not used in RMII mode.

24.6 Programming Interface

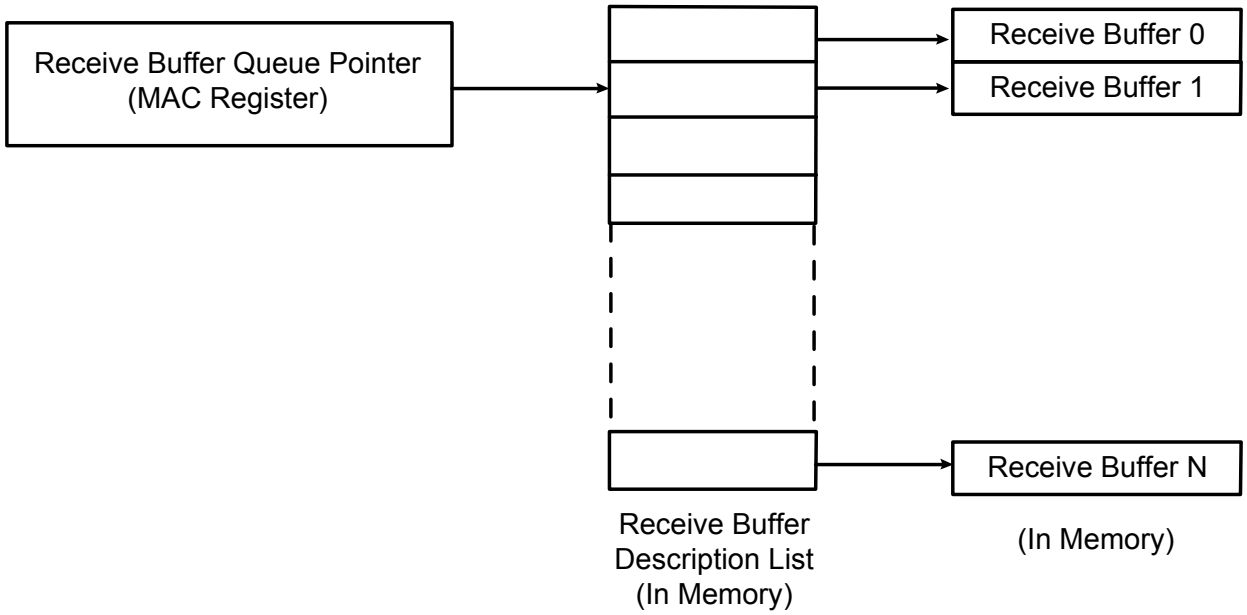
24.6.1 Configuration

Initialization of the MACB configuration (e.g. frequency ratios) must be done while the transmit and receive circuits are disabled. Network control register and network configuration register are described below.

24.6.2 Receive Buffer List

Receive data is written to areas of data (i.e., buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (receive buffer queue) is a sequence of descriptor entries as defined in ”Receive Buffer Descriptor Entry” on page 491. It points to this data structure.

Figure 24-2. Receive Buffer List



To create the list of buffers:

1. Allocate a number (*n*) of buffers of 128 bytes in system memory.
2. Allocate an area 2*n* words for the receive buffer descriptor entry in system memory and create *n* entries in this list. Mark all entries in this list as owned by MACB, i.e., bit 0 of word 0 set to 0.
3. If less than 1024 buffers are defined, the last descriptor must be marked with the wrap bit (bit 1 in word 0 set to 1).
4. Write address of receive buffer descriptor entry to MACB register receive buffer queue pointer.
5. The receive circuits can then be enabled by writing to the address recognition registers and then to the network control register.

24.6.3 Transmit Buffer List

Transmit data is read from the system memory These buffers are listed in another data structure that also resides in main memory. This data structure (Transmit Buffer Queue) is a sequence of descriptor entries (as defined in Table 24-2 on page 494) that points to this data structure.

To create this list of buffers:

1. Allocate a number (*n*) of buffers of between 1 and 2047 bytes of data to be transmitted in system memory. Up to 128 buffers per frame are allowed.
2. Allocate an area 2*n* words for the transmit buffer descriptor entry in system memory and create N entries in this list. Mark all entries in this list as owned by MACB, i.e. bit 31 of word 1 set to 0.
3. If fewer than 1024 buffers are defined, the last descriptor must be marked with the wrap bit (bit 30 in word 1 set to 1).
4. Write address of transmit buffer descriptor entry to MACB register transmit buffer queue pointer.
5. The transmit circuits can then be enabled by writing to the network control register.

24.6.4 Address Matching

The MACB register-pair hash address and the four specific address register-pairs must be written with the required values. Each register-pair comprises a bottom register and top register, with the bottom register being written first. The address matching is disabled for a particular register-pair after the bottom-register has been written and re-enabled when the top register is written. See Section “24.5.8” on page 497. for details of address matching. Each register-pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

24.6.5 Interrupts

There are 14 interrupt conditions that are detected within the MACB. These are ORed to make a single interrupt. This interrupt is handled by the interrupt controller. On receipt of the interrupt signal, the CPU enters the interrupt handler. To ascertain which interrupt has been generated, read the interrupt status register. Note that this register clears itself when read. At reset, all interrupts are disabled. To enable an interrupt, write to interrupt enable register with the pertinent interrupt bit set to 1. To disable an interrupt, write to interrupt disable register with the pertinent interrupt bit set to 1. To check whether an interrupt is enabled or disabled, read interrupt mask register: if the bit is set to 1, the interrupt is disabled.

24.6.6 Transmitting Frames

To set up a frame for transmission:

1. Enable transmit in the network control register.
2. Allocate an area of system memory for transmit data. This does not have to be contiguous, varying byte lengths can be used as long as they conclude on byte borders.
3. Set-up the transmit buffer list.
4. Set the network control register to enable transmission and enable interrupts.
5. Write data for transmission into these buffers.
6. Write the address to transmit buffer descriptor queue pointer.
7. Write control and length to word one of the transmit buffer descriptor entry.
8. Write to the transmit start bit in the network control register.

24.6.7 Receiving Frames

When a frame is received and the receive circuits are enabled, the MACB checks the address and, in the following cases, the frame is written to system memory:

- if it matches one of the four specific address registers.
- if it matches the hash address function.
- if it is a broadcast address (0xFFFFFFFFFFFF) and broadcasts are allowed.
- if the MACB is configured to copy all frames.
- if the EAM is asserted before four words have been loaded into the receive FIFO.

The register receive buffer queue pointer points to the next entry (see Table 24-1 on page 491) and the MACB uses this as the address in system memory to write the frame to. Once the frame has been completely and successfully received and written to system memory, the MACB then updates the receive buffer descriptor entry with the reason for the address match and marks the area as being owned by software. Once this is complete an interrupt receive complete is set. Software is then responsible for handling the data in the buffer and then releasing the buffer by writing the ownership bit back to 0.

If the MACB is unable to write the data at a rate to match the incoming frame, then an interrupt receive overrun is set. If there is no receive buffer available, i.e., the next buffer is still owned by software, the interrupt receive buffer not available is set. If the frame is not successfully received, a statistic register is incremented and the frame is discarded without informing software.

24.7 User Interface

Table 24-6. MACB Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Network Control Register	NCR	Read/Write	0x00000000
0x04	Network Configuration Register	NCFG	Read/Write	0x00008000
0x08	Network Status Register	NSR	Read-only	-
0x14	Transmit Status Register	TSR	Read/Write	0x00000000
0x18	Receive Buffer Queue Pointer Register	RBQP	Read/Write	0x00000000
0x1C	Transmit Buffer Queue Pointer Register	TBQP	Read/Write	0x00000000
0x20	Receive Status Register	RSR	Read/Write	0x00000000
0x24	Interrupt Status Register	ISR	Read/Write	0x00000000
0x28	Interrupt Enable Register	IER	Write-only	0x00000000
0x2C	Interrupt Disable Register	IDR	Write-only	0x00000000
0x30	Interrupt Mask Register	IMR	Read-only	0x00003FFF
0x34	Phy Maintenance Register	MAN	Read/Write	0x00000000
0x38	Pause Time Register	PTR	Read/Write	0x00000000
0x3C	Pause Frames Received Register	PFR	Read/Write	0x00000000
0x40	Frames Transmitted Ok Register	FTO	Read/Write	0x00000000
0x44	Single Collision Frames Register	SCF	Read/Write	0x00000000
0x48	Multiple Collision Frames Register	MCF	Read/Write	0x00000000
0x4C	Frames Received Ok Register	FRO	Read/Write	0x00000000
0x50	Frame Check Sequence Errors Register	FCSE	Read/Write	0x00000000
0x54	Alignment Errors Register	ALE	Read/Write	0x00000000
0x58	Deferred Transmission Frames Register	DTF	Read/Write	0x00000000
0x5C	Late Collisions Register	LCOL	Read/Write	0x00000000
0x60	Excessive Collisions Register	EXCOL	Read/Write	0x00000000
0x64	Transmit Underrun Errors Register	TUND	Read/Write	0x00000000
0x68	Carrier Sense Errors Register	CSE	Read/Write	0x00000000
0x6C	Receive Resource Errors Register	RRE	Read/Write	0x00000000
0x70	Receive Overrun Errors Register	ROV	Read/Write	0x00000000
0x74	Receive Symbol Errors Register	RSE	Read/Write	0x00000000
0x78	Excessive Length Errors Register	ELE	Read/Write	0x00000000
0x7C	Receive Jabbers Register	RJA	Read/Write	0x00000000
0x80	Undersize Frames Register	USF	Read/Write	0x00000000
0x84	SQE Test Errors Register	STE	Read/Write	0x00000000
0x88	Received Length Field Mismatch Register	RLE	Read/Write	0x00000000
0x8C	Transmitted Pause Frames Register	TPF	Read/Write	0x00000000
0x90	Hash Register Bottom	HRB	Read/Write	0x00000000

Table 24-6. MACB Register Memory Map (Continued)

Offset	Register	Register Name	Access	Reset
0x94	Hash Register Top	HRT	Read/Write	0x00000000
0x98	Specific Address 1 Bottom Register	SA1B	Read/Write	0x00000000
0x9C	Specific Address 1 Top Register	SA1T	Read/Write	0x00000000
0xA0	Specific Address 2 Bottom Register	SA2B	Read/Write	0x00000000
0xA4	Specific Address 2 Top Register	SA2T	Read/Write	0x00000000
0xA8	Specific Address 3 Bottom Register	SA3B	Read/Write	0x00000000
0xAC	Specific Address 3 Top Register	SA3T	Read/Write	0x00000000
0xB0	Specific Address 4 Bottom Register	SA4B	Read/Write	0x00000000
0xB4	Specific Address 4 Top Register	SA4T	Read/Write	0x00000000
0xB8	Type ID Checking Register	TID	Read/Write	0x00000000
0xBC	Transmit Pause Quantum Register	TPQ	Read/Write	0x0000FFFF
0xC0	User Input/output Register	USRIO	Read/Write	0x00000000
0xC4	Wake on LAN Register	WOL	Read/Write	0x00000000
0xFC	Version Register	VERSION	Read-only	- (1)

1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

Statistics registers (PFR to TPF) should be read frequently enough to prevent loss of data. The receive statistics registers are only incremented when the receive enable bit is set in the network control register (NCR.RE). Write access to statistics registers is allowed if NCR.WESTAT is set. Statistic registers are cleared on a read and stick at all ones when they count to their maximum value.

24.7.1 Network Control Register

Name: NCR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	TZQ	TPF	THALT	TSTART	BP

7	6	5	4	3	2	1	0
WESTAT	INCSTAT	CLRSTAT	MPE	TE	RE	LLB	LB

- **TZQ: Transmit Zero Quantum Pause Frame**
  Writing a one to this bit sends a pause frame with zero pause quantum at the next available transmitter idle time.
- **TPF: Transmit Pause Frame**
  Writing a one to this bit sends a pause frame with the pause quantum from the transmit pause quantum register at the next available transmitter idle time.
- **THALT: Transmit Halt**
  Writing a one to this bit halts transmission as soon as any ongoing frame transmission ends.
- **TSTART: Start Transmission**
  Writing a one to this bit starts transmission.
- **BP: Back Pressure**
  0: No collision are forced.
  1: In half duplex mode, forces collisions on all received frames.
- **WESTAT: Write Enable for Statistics Registers**
  0: Statistics registers are read-only.
  1: Statistics registers are writable for functional test purposes.
- **INCSTAT: Increment Statistics Registers**
  Writing a one increments all the statistics registers by one for test purposes.
- **CLRSTAT: Clear ¨Statistics Registers**
  Writing a one clears the statistics registers.
- **MPE: Management Port Enable**
  0: Forces MDIO to high impedance state and DPC low.
  1: Enables the management port.
- **TE: Transmit Enable**
  0: Transmission stops immediately, the transmit FIFO and control registers are cleared and the transmit queue pointer register resets to point to the start of the transmit descriptor list.
  1: Enables the Ethernet transmitter to send data.

- **RE: Receive Enable**
  0: Frame reception stops immediately and the receive FIFO is cleared. The receive queue pointer register is unaffected.
  1: Enables the MACB to receive data.
- **LLB: Local Loopback**
  0: Local loopback is disabled.
  1: Local loopback is enabled. It connects TXD to RXD, TX_EN to RX_DV, forces full duplex and drives RX_CLK and TX_CLK with CLK_MACB_PB divided by 4. RX_CLK and TX_CLK may glitch as the MACB is switched into and out of internal loop back. It is important that receive and transmit circuits have already been disabled when making the switch into and out of internal loop back. This function may not be supported by some instantiations of the MACB.
- **LB: Loopback**
  0: Loopback is disabled.
  1: Loopback is enabled. Asserts the loopback signal to the PHY.

24.7.2 Network Configuration Register

Name: NCFGR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00008000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	IRXFCS	EFRHD	DRFCS	RLCE

15	14	13	12	11	10	9	8
RBOF		PAE	RTY	CLK		EAE	FS

7	6	5	4	3	2	1	0
UNI	MTI	NBC	CAF	JFRAME	BR	FD	SPD

- **IRXFCS: Ignore RX FCS**
  0: Normal operation.
  1: Frames with FCS/CRC errors are not rejected and no FCS error statistics are counted.
- **EFRHD: Enable Frames to be Received in Half-Duplex mode**
  0: Disabled.
  1: Enabled (while transmitting).
- **DRFCS: Discard Receive FCS**
  0: FCS field of received frames are copied to memory.
  1: FCS field of received frames are not copied to memory.
- **RLCE: Receive Length field Checking Enable**
  0: Disabled.
  1: Frames with measured lengths shorter than their length fields are discarded. Frames containing a type ID in bytes 13 and 14 (length/type ID = 0600) are not be counted as length errors.
- **RBOF: Receive Buffer Offset**
  Indicates the number of bytes by which the received data is offset from the start of the first receive buffer.

RBOF	Offset
00	No offset from start of receive buffer
01	One-byte offset from start of receive buffer
10	Two-byte offset from start of receive buffer
11	Three-byte offset from start of receive buffer

- **PAE: Pause Enable**
  0: Pause disabled.
  1: Pause enabled. Transmission pauses when a valid pause frame is received.

- **RTY: Retry Test**
  0: Normal operation.
  1: The back off between collisions is always one slot time. It helps testing the too many retries condition. Also used in the pause frame tests to reduce the pause counters decrement time from 512 bit times, to every RX_CLK cycle.
- **CLK: PB Clock Divider**
  Determines by what number system clock is divided to generate Divided PB Clock (DPC). For conformance with 802.3, DPC must not exceed 2.5MHz (DPC is only active during MDIO read and write operations).

CLK	DPC
00	PB clock divided by 8 (PB clock up to 20 MHz)
01	PB clock divided by 16 (PB clock up to 40 MHz)
10	PB clock divided by 32 (PB clock up to 80 MHz)
11	PB clock divided by 64 (PB clock up to 160 MHz)

- **EAE: External Address Match Enable**
  0: External address match is disabled.
  1: External address match is enabled. Eam pin can be used to copy frames to memory.
- **FS: Frame Size**
  0: Reject any frames above 1518 bytes.
  1: Accept frames up to 1536 bytes.
- **UNI: Unicast Hash Enable**
  0: Unicast hash is disabled.
  1: Unicast hash is enabled. Unicast frames are received when the 6-bit hash function of the destination address points to a bit that is set in the hash register.
- **MTI: Multicast Hash Enable**
  0: Multicast hash is disabled.
  1: Multicast hash is enabled. Multicast frames are received when the 6-bit hash function of the destination address points to a bit that is set in the hash register.
- **NBC: No Broadcast**
  0: Frames addressed to the broadcast address of all ones are received.
  1: Frames addressed to the broadcast address of all ones are not received.
- **JFRAME: Jumbo Frames**
  0: Jumbo frames are disabled.
  1: Enable jumbo frames of up to 10240 bytes to be accepted.
- **CAF: Copy All Frames**
  0: Copy all frames is disabled.
  1: All valid frames are received.
- **BR: Bitrate**
  0: Data is transmitted least significant nibble first.
  1: Data is serialized and transmitted least significant bit first (10Mbps). Must be written before receive and transmit enable in the network control register. Serial interface is configured with transmit and receive data being driven out on TXD[0] and received on RXD[0] serially. Also the CRS and RX_DV are logically ORed together so either may be used as the data valid signal.
- **FD: Full Duplex**
  0: Full duplex mode is disabled.
  1: Full duplex mode is enabled. Transmit sub-module ignores the state of collision and carrier sense and allows receive while transmitting. Also controls the half duplex pin.
- **SPD: Speed**
  0: 10 Mbit/s speed.
  1: 100 Mbit/s speed. Bit value is reflected on the SPEED pin.

24.7.3 Network Status Register

Name: NSR

Access Type: Read-only

Offset: 0x08

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	IDLE	MDIO	-

- **IDLE: IDLE Status**
  0: PHY management logic is idle (i.e., has completed).
  1: PHY management logic is running.
- **MDIO: MDIO Pin Status**
  Use the PHY maintenance register for reading managed frames rather than this bit.

24.7.4 Transmit Status Register

Name: TSR

Access Type: Read/Write

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	UND	COMP	BEX	TGO	RLE	COL	UBR

This register, when read, returns details of the status of a transmit. Once read, individual bits may be cleared by writing a one to them. It is not possible to write a bit to one by writing to the register.

- **UND: Transmit Underrun**
  This bit is set when transmit DMA was not able to read data from memory, either because the bus was not granted in time or because a used bit was read midway through frame transmission. If this occurs, the transmitter forces bad CRC.
  Write a one to clear this bit.
  0: No transmit underrun.
  1: Transmit underrun.
- **COMP: Transmit Complete**
  This bit is set when a frame has been transmitted.
  Write a one to clear this bit.
  0: Transmit is not completed.
  1: Transmit is completed.
- **BEX: Buffers Exhausted Mid Frame**
  This bit is set if the buffers run out during transmission of a frame. Then transmission stops, FCS shall be bad and TX_ER is asserted.
  Write a one to clear to this bit.
  0: Buffer is not exhausted.
  1: Buffer is exhausted.
- **TGO: Transmit Go**
  0: Transmit is inactive.
  1: Transmit is active.
- **RLE: Retry Limit Exceeded**
  This bit is set when retry limit has exceeded.
  Write a one to clear this bit.
  0: Retry limit is not exceeded.

1: Retry limit is exceeded.

- **COL: Collision Occurred**
  This bit is set by the assertion of collision.
  Write a one to clear this bit.
  0: No collision detected.
  1: Collision detected.
- **UBR: Used Bit Read**
  This bit is set when a transmit buffer descriptor is read with its used bit set.
  Write a one to clear this bit.
  0: Used bit is not set.
  1: Used bit set.

24.7.5 Receive Buffer Queue Pointer Register

Name: RBQP

Access Type: Read/Write

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
ADDR[29:22]							

23	22	21	20	19	18	17	16
ADDR[21:14]							

15	14	13	12	11	10	9	8
ADDR[13:6]							

7	6	5	4	3	2	1	0
ADDR[5:0]						-	-

This register points to the entry in the receive buffer queue (descriptor list) currently being used. It is written with the start location of the receive buffer descriptor list. The lower order bits increment as buffers are used up and wrap to their original values after either 1024 buffers or when the wrap bit of the entry is set.

Reading this register returns the location of the descriptor currently being accessed. This value is incremented as buffers are used. User should not use this register to determine where to remove received frames from the queue as it constantly changes when new frames are received. User should instead use the buffer descriptor queue checking the used bits.

Receive buffer writes can be bursts of two words and, as with transmit buffer reads, it is recommended that bit 2 is always written to zero to prevent a burst crossing a 1K boundary, in violation of the System Bus specification.

- **ADDR: Receive Buffer Queue Pointer Address**
  Write this field to set the start address of the receive queue.
  Read this field to get the address of the current buffer being used.

24.7.6 Transmit Buffer Queue Pointer Register

Name: TBQP

Access Type: Read/Write

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
ADDR[29:22]							

23	22	21	20	19	18	17	16
ADDR[21:14]							

15	14	13	12	11	10	9	8
ADDR[13:6]							

7	6	5	4	3	2	1	0
ADDR[5:0]						-	-

This register points to the entry in the transmit buffer queue (descriptor list) currently being used. It is written with the start location of the transmit buffer descriptor list. The lower order bits increment as buffers are used up and wrap to their original values after either 1024 buffers or when the wrap bit of the entry is set. This register can only be written when bit TSR.TGO is low.

As transmit buffer reads can be bursts of two words, it is recommended that bit 2 is always written to zero to prevent a burst crossing a 1K boundary, in violation of the System Bus specification.

- **ADDR: Transmit buffer queue pointer address**
  Write this field to set the start address of the transmit queue.
  Read this field to get the address of the first buffer of the frame being transmitted or about to be transmitted.

24.7.7 Receive Status Register

Name: RSR

Access Type: Read/Write

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	OVR	REC	BNA

This register, when read, returns details of the status of a receive. Once read, individual bits may be cleared by writing a one to them. It is not possible to write a bit to one by writing to the register.

- **OVR: Receive Overrun**
  This bit is set when the DMA interface is unable to store the receive frame to memory, either because the bus was not granted in time or because a bus error was returned. The buffer is recovered if this happens.
  Write a one to clear this bit.
  0: No receive overrun detected.
  1: Receive overrun detected.
- **REC: Frame Received**
  This bit is set when one or more frames have been received and placed in memory.
  Write a one to clear this bit.
  0: No frame received.
  1: Frame received.
- **BNA: Buffer Not Available**
  The DMA reads the pointer each time a new frame starts, until a valid pointer is found. This bit is set at each attempt that fails even if it has not had a successful pointer read since it has been cleared.
  Write a one to clear this bit.
  0: Buffer is available.
  1: Buffer is not available because an attempt was made to get a new buffer and the pointer indicated that it was owned by the processor.

24.7.8 Interrupt Status Register

Name: ISR

Access Type: Read/Write

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	PTZ	PFR	HRESP	ROVR	-	-

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

- **PTZ: Pause Time Zero**
  This bit is set when the pause time register (PTR) decrements to zero.
  This bit is cleared after read.
  0: PTR > 0.
  1: PTR = 0.
- **PFR: Pause Frame Received**
  This bit is cleared after read.
  0: No valid pause frame received.
  1: Valid pause frame received.
- **HRESP: Hresp Not OK**
  This bit is set when the DMA interface detects a bus error.
  This bit is cleared after read.
  0: HRESP is OK.
  1: HRESP is not OK.
- **ROVR: Receive Overrun**
  This bit is set when the receive overrun status bit is set (RSR.OVR).
  This bit is cleared after read.
  0: RSR.OVR is not set.
  1: RSR.OVR has been set.
- **TCOMP: Transmit Complete**
  This bit is set when a frame has been transmitted.
  This bit is cleared after read.
  0: Transmit is not completed.
  1: Transmit is completed.
- **TXERR: Transmit Error**
  This bit is set when transmit buffers exhausted in mid-frame.

This bit is cleared after read.

0: No transmit error.

1: Transmit error detected.

- **RLE: Retry Limit Exceeded**
  This bit is cleared after read.
  0: Retry limit is not exceeded.
  1: Retry limit is exceeded.
- **TUND: Ethernet Transmit Buffer Underrun**
  This bit is set if the DMA did not fetch frame data to transmit in time or HRESP returned not OK. It is also set if a used bit is read mid-frame or when a new transmit queue pointer is written.
  This bit is cleared after read.
  0: No underrun detected for transmit buffer.
  1: Underrun detected for transmit buffer.
- **TXUBR: Transmit Used Bit Read**
  This bit is cleared after read.
  0: Normal operation.
  1: Transmit buffer descriptor is read with its used bit set.
- **RXUBR: Receive Used Bit Read**
  This bit is cleared after read.
  0: Normal operation.
  1: Receive buffer descriptor is read with its used bit set.
- **RCOMP: Receive Complete**
  This bit is set when a frame has been stored in memory.
  This bit is cleared after read.
  0: Receive is not completed.
  1: Receive is completed.
- **MFD: Management Frame Done**
  This bit is cleared after read.
  0: Management frame is not done.
  1: Management frame is done. PHY maintenance register has completed its operation.

24.7.9 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x28

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	PTZ	PFR	HRESP	ROVR	-	-

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

24.7.10 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x2C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	PTZ	PFR	HRESP	ROVR	-	-

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

24.7.11 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x30

Reset Value: 0x00003FFF

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	PTZ	PFR	HRESP	ROVR	-	-

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared when the corresponding bit in IDR is written to one.

A bit in this register is set when the corresponding bit in IER is written to one.

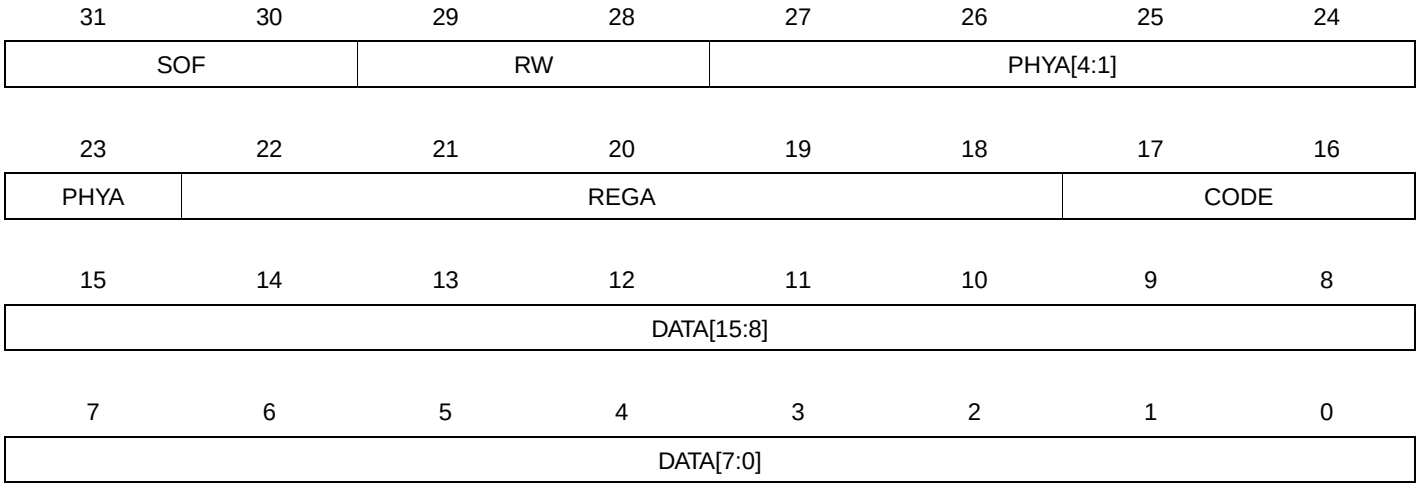
24.7.12 PHY Maintenance Register

Name: MAN

Access Type: Read/Write

Offset: 0x34

Reset Value: 0x00000000



- **SOF: Start Of Frame**
  Must be written to 01 for a valid frame.
- **RW: Read/Write**
  10: Read operation
  01: Write operation.
  Any other value is an invalid PHY management frame
- **PHYA: PHY Address**
  PHY address.
- **REGA: Register Address**
  PHY register address to access.
- **CODE: Code value**
  Must be written to 10.
- **DATA: PHY Data**
  For a write operation, write the data to be written to the PHY.
  After a read operation, contains the data read from the PHY.

24.7.13 Pause Time Register

Name: PTR

Access Type: Read/Write

Offset: 0x38

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
PTIME[15:8]							

7	6	5	4	3	2	1	0
PTIME[7:0]							

- **PTIME: Pause Time**

  Current value of the pause time register which is decremented every 512 bit times.

24.7.14 Pause Frames Received Register

Name: PFR

Access Type: Read/Write

Offset: 0x3C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
PFROK[15:8]							

7	6	5	4	3	2	1	0
PFROK[7:0]							

- **PFROK: Pauses Frames Received OK**

  Number of good pause frames received. A good frame has a length of 64 to 1518 bytes (1536 if bit NCFGR.FS is set, 10240 if bit NCFGR.JFRAME is set) and has no FCS, alignment or receive symbol errors.

24.7.15 Frames Transmitted OK Register

Name: FTO

Access Type: Read/Write

Offset: 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
FTOK[23:16]							

15	14	13	12	11	10	9	8
FTOK[15:8]							

7	6	5	4	3	2	1	0
FTOK[7:0]							

- **FTOK: Frames Transmitted OK**

  Number of frames successfully transmitted, i.e., no underrun and not too many retries.

24.7.16 Single Collision Frames Register

Name: SCF

Access Type: Read/Write

Offset: 0x44

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
SCF[15:8]							

7	6	5	4	3	2	1	0
SCF[7:0]							

- **SCF: Single Collision Frames**

  Number of frames experiencing a single collision before being successfully transmitted, i.e., no underrun.

24.7.17 Multicollision Frames Register

Name: MCF

Access Type: Read/Write

Offset: 0x48

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
MCF[15:8]							

7	6	5	4	3	2	1	0
MCF[7:0]							

- **MCF: Multicollision Frames**

  Number of frames experiencing between 2 and 15 collisions prior to being successfully transmitted, i.e., no underrun and not too many retries.

24.7.18 Frames Received OK Register

Name: FRO

Access Type: Read/Write

Offset: 0x4C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
FROK[23:16]							

15	14	13	12	11	10	9	8
FROK[15:8]							

7	6	5	4	3	2	1	0
FROK[7:0]							

- **FROK: Frames Received OK**

  Number of frames successfully received, i.e., address recognized and successfully copied to memory. A good frame has a length of 64 to 1518 bytes (1536 if bit NCFGR.FS is set, 10240 if bit NCFGR.JFRAME is set) and has no FCS, alignment or receive symbol errors.

24.7.19 Frames Check Sequence Errors Register

Name: FCSE

Access Type: Read/Write

Offset: 0x50

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
FCSE							

- **FCSE: Frame Check Sequence Errors**
  Number of frames which have an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length (1536 if bit NCFGR.FS is set, 10240 if bit NCFGR.JFRAME is set). This register is also incremented if a symbol error is detected and the frame is of valid length and has an integral number of bytes.

24.7.20 Alignment Errors Register

Name: ALE

Access Type: Read/Write

Offset: 0x54

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
ALE							

- **ALE: Alignment Errors**
  Number of frames which have not an integral number of bytes and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length (1536 if bit NCFGR.FS is set, 10240 if bit NCFGR.JFRAME is set). This register is also incremented if a symbol error is detected and the frame is of valid length and does not have an integral number of bytes.

24.7.21 Deferred Transmission Frames Register

Name: DTF

Access Type: Read/Write

Offset: 0x58

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
DTF[15:8]							

7	6	5	4	3	2	1	0
DTF[7:0]							

- **DTF: Deferred Transmission Frames**

  Number of frames experiencing deferral due to carrier sense being active on their first attempt at transmission. Frames involved in any collision are not counted nor are frames that experienced a transmit underrun.

24.7.22 Late Collisions Register

Name: LCOL

Access Type: Read/Write

Offset: 0x5C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
LCOL							

- **LCOL: Late Collisions**
  Number of frames that experience a collision after the slot time (512 bits) has expired. A late collision is counted twice; i.e., both as a collision and a late collision.

24.7.23 Excessive Collisions Register

Name: EXCOL

Access Type: Read/Write

Offset: 0x60

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
EXCOL							

- **EXCOL: Excessive Collisions**
  Number of frames that failed to be transmitted because they experienced 16 collisions.

24.7.24 Transmit Underrun Errors Register

Name: TUND

Access Type: Read/Write

Offset: 0x64

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
TUND							

- **TUND: Transmit Underruns**
  Number of frames not transmitted due to a transmit DMA underrun. If this register is incremented, then no other statistics register is incremented.

24.7.25 Carrier Sense Errors Register

Name: CSE

Access Type: Read/Write

Offset: 0x68

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
CSE							

- **CSE: Carrier Sense Errors**
  Number of frames transmitted where carrier sense was not seen during transmission or where carrier sense was deasserted after being asserted in a transmit frame without collision (no underrun). Only incremented in half-duplex mode. The only effect of a carrier sense error is to increment this register. The behavior of the other statistics registers is unaffected by the detection of a carrier sense error.

24.7.26 Received Resource Errors Register

Name: RRE

Access Type: Read/Write

Offset: 0x6C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
RRE[15:8]							

7	6	5	4	3	2	1	0
RRE[7:0]							

- **RRE: Received Resource Errors**

  Number of frames that address matched but could not be copied to memory because no receive buffer was available.

24.7.27 Received Overrun Errors Register

Name: ROVR

Access Type: Read/Write

Offset: 0x70

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
ROVR							

- **ROVR: Received Overrun Errors**

  Number of frames that are address recognized but were not copied to memory because of receive DMA overrun.

24.7.28 Received Symbol Errors Register

Name: RSE

Access Type: Read/Write

Offset: 0x74

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RSE							

- **RSE: Received Symbol Errors**

  Number of frames that had RX_ER asserted during reception. Receive symbol errors are also counted as an FCS or alignment error if the frame length is between 64 and 1518 bytes (1536 if bit NCFGR.FS is set, 10240 if bit NCFGR.JFRAME is set). If the frame is larger, it is recorded as a jabber error.

24.7.29 Excessive Length Errors Register

Name: ELE

Access Type: Read/Write

Offset: 0x78

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
EXL							

- **EXL: Excessive Length Errors**

  Number of frames received exceeding 1518 bytes (1536 if bit NCFGR.FS is set, 10240 if bit NCFGR.JFRAME is set) in length but do not have either a CRC error, an alignment error nor a receive symbol error.

24.7.30 Receive Jabbers Register

Name: RJA

Access Type: Read/Write

Offset: 0x7C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RJB							

- **RJB: Receive Jabbers**

  Number of frames received exceeding 1518 bytes (1536 if bit NCFGR.FS is set, 10240 if bit NCFGR.JFRAME is set) in length and have either a CRC error, an alignment error or a receive symbol error.

24.7.31 Undersize Frames Register

Name: USF

Access Type: Read/Write

Offset: 0x80

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
USF							

- **USF: Undersize Frames**

  Number of frames received less than 64 bytes in length but do not have either a CRC error, an alignment error or a receive symbol error.

24.7.32 SQE Test Errors Register

Name: STE

Access Type: Read/Write

Offset: 0x84

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
SQER							

- **SQER: SQE Test Errors**

  Number of frames where COL was not asserted within 96 bit times (interframe gap) of TX_EN being deasserted in half duplex mode.

24.7.33 Received Length Field Mismatch Register

Name: RLE

Access Type: Read/Write

Offset: 0x88

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RLFM							

- **RLFM: Receive Length Field Mismatch**

  Number of frames received that have a measured length shorter than extracted from its length field. Checking is enabled by bit NCFGR.RLCE. Frames containing a type ID in bytes 13 and 14 (i.e., length/type ID 0x0600) are not counted as length field errors, neither are excessive length frames.

24.7.34 Transmitted Pause Frames Register

Name: TPF

Access Type: Read/Write

Offset: 0x8C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
TPF[15:8]							

7	6	5	4	3	2	1	0
TPF[7:0]							

- **TPF: Transmitted Pause Frames**

  Number of pause frames transmitted.

24.7.35 Hash Register Bottom

Name: HRB

Access Type: Read/Write

Offset: 0x90

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
ADDR[31:24]							

23	22	21	20	19	18	17	16
ADDR[23:16]							

15	14	13	12	11	10	9	8
ADDR[15:8]							

7	6	5	4	3	2	1	0
ADDR[7:0]							

- **ADDR: Hash Address Low**

  Low value of the hash address register. See ”Hash Addressing” on page 498.

24.7.36 Hash Register Top

Name: HRT

Access Type: Read/Write

Offset: 0x94

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
ADDR[31:24]							

23	22	21	20	19	18	17	16
ADDR[23:16]							

15	14	13	12	11	10	9	8
ADDR[15:8]							

7	6	5	4	3	2	1	0
ADDR[7:0]							

- **ADDR: Hash Address High**

  High value of the hash address register. See "Hash Addressing" on page 498.

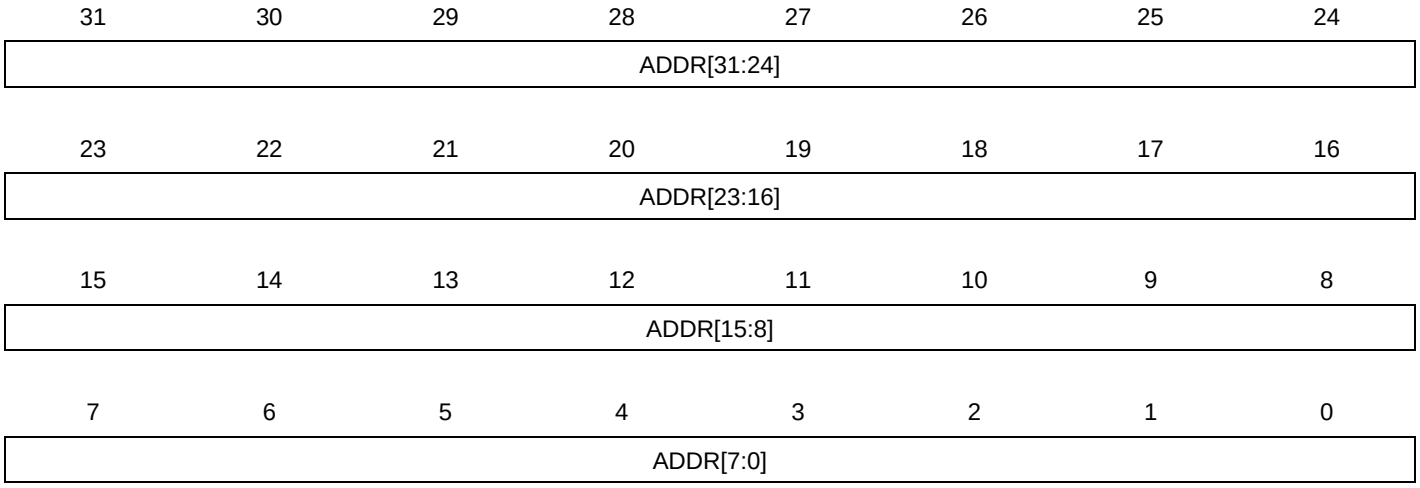
24.7.37 Specific Address 1 Bottom Register

Name: SA1B

Access Type: Read/Write

Offset: 0x98

Reset Value: 0x00000000



- **ADDR: Destination Address Low**

  Low value of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

24.7.38 Specific Address 1 Top Register

Name: SA1T

Access Type: Read/Write

Offset: 0x9C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
ADDR[15:8]							

7	6	5	4	3	2	1	0
ADDR[7:0]							

- **ADDR: Destination Address High**

  High value of the destination address (bits 32 to 47).

24.7.39 Specific Address 2 Bottom Register

Name: SA2B

Access Type: Read/Write

Offset: 0xA0

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
ADDR[31:24]							

23	22	21	20	19	18	17	16
ADDR[23:16]							

15	14	13	12	11	10	9	8
ADDR[15:8]							

7	6	5	4	3	2	1	0
ADDR[7:0]							

- **ADDR: Destination Address Low**

  Low value of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

24.7.40 Specific Address 2 Top Register

Name: SA2T

Access Type: Read/Write

Offset: 0xA4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
ADDR[15:8]							

7	6	5	4	3	2	1	0
ADDR[7:0]							

- **ADDR: Destination Address High**

  High value of the destination address (bits 32 to 47).

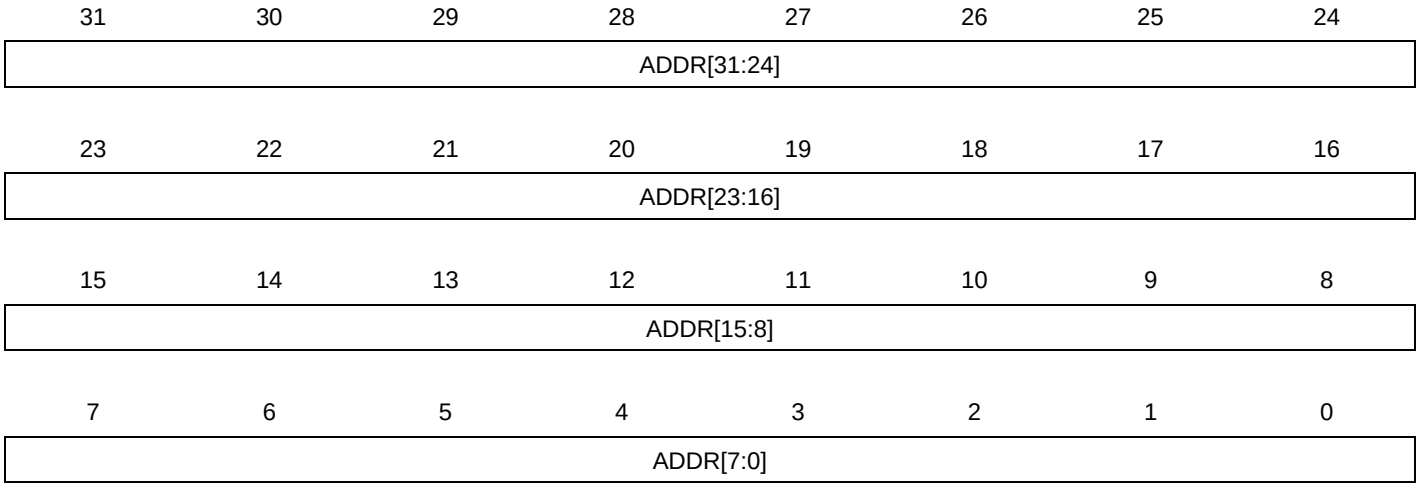
24.7.41 Specific Address 3 Bottom Register

Name: SA3B

Access Type: Read/Write

Offset: 0xA8

Reset Value: 0x00000000



- **ADDR: Destination Address Low**

  Low value of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

24.7.42 Specific Address 3 Top Register

Name: SA3T

Access Type: Read/Write

Offset: 0xAC

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
ADDR[15:8]							

7	6	5	4	3	2	1	0
ADDR[7:0]							

- **ADDR: Destination Address High**

  High value of the destination address (bits 32 to 47).

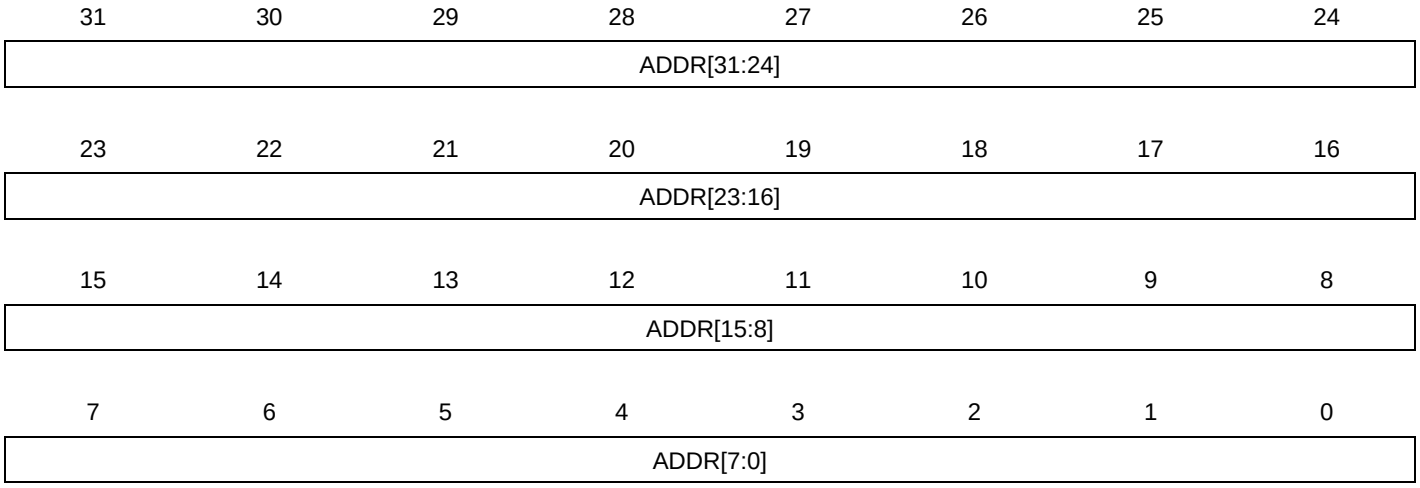
24.7.43 Specific Address 4 Bottom Register

Name: SA4B

Access Type: Read/Write

Offset: 0xB0

Reset Value: 0x00000000



- **ADDR: Destination Address Low**

  Low value of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

24.7.44 Specific Address 4 Top Register

Name: SA4T

Access Type: Read/Write

Offset: 0xB4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
ADDR[15:8]							

7	6	5	4	3	2	1	0
ADDR[7:0]							

- **ADDR: Destination Address High**

  High value of the destination address (bits 32 to 47).

24.7.45 Type ID Checking Register

Name: TID

Access Type: Read/Write

Offset: 0xB8

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
TID[15:8]							

7	6	5	4	3	2	1	0
TID[7:0]							

- **TID: Type ID Checking**

  Comparison value for received frames (TypeID/Length field).

24.7.46 Transmit Pause Quantum Register

Name: TPQ

Access Type: Read/Write

Offset: 0xBC

Reset Value: 0x0000FFFF

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
TPQ[15:8]							

7	6	5	4	3	2	1	0
TPQ[7:0]							

- **TPQ: Transmit Pause Quantum**

  Used in hardware generation of transmitted pause frames as value for pause quantum.

24.7.47 User Input/Output Register

Name: USRIO

Access Type: Read/Write

Offset: 0xC0

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	WOL	SPD	BR	HD	LB

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	TPZ	TP	EAM	RMII

This register, when read, returns details of the status of a receive. Once read, individual bits may be cleared by writing a one to them. It is not possible to write a bit to one by writing to the register.

- **WOL: Wake-On LAN**
  0: Wake-on LAN not detected.
  1: Wake-on LAN detected (read-only).
- **SPD: Speed**
  Image of NCFGR.SPD bit (read-only).
- **BR: Bitrate**
  Image of NCFGR.BR bit (read-only).
- **HD: Half Duplex**
  Inversion of NCFGR.FD bit (read-only).
- **LB: Loopback**
  Image of NCR.LB bit (read-only).
- **TPZ: Transmit Pause Frame Zero Quantum**
  0: Pause frame is TPQ register value quantum length.
  1: Pause frame is zero quantum length.
- **TP: Transmit Pause Frame**
  Toggle this bit to send a Pause frame.
- **EAM: External Address Match**
  0: No frame is copied to memory.
  1: Frame is copied to memory if NCFGR.EAE is set.
- **RMII: RMII mode**
  0: MII operation mode.
  1: RMII operation mode.

24.7.48 Wake-On-LAN Register

Name: WOL

Access Type: Read/Write

Offset: 0xC4

Reset Value: 0x0000FFFF

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	MTI	SA1	ARP	MAG

15	14	13	12	11	10	9	8
IP[15:8]							

7	6	5	4	3	2	1	0
IP[7:0]							

- **MTI: Multicast Hash Event Enable**
  0: Multicast hash events are disabled.
  1:Multicast hash events assert WOL pin.
- **SA1: Specific Address Register 1 Event Enable**
  0: SAR1 events are disabled.
  1: SAR1 events assert WOL pin.
- **ARP: ARP Request Event Enable**
  0: ARP request events are disabled.
  1: ARP request events assert WOL pin.
- **MAG: Magic Packet Event Enable**
  0: Magic packet events are disabled.
  1: Magic packet events assert WOL pin.
- **IP: ARP request IP address**
  16 LSB bits of target IP. When matched, a wake-on-LAN event is generated. Zero value does not generate an event, even if it is matched by the received frame.

24.7.49 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: -

31	30	29	28	27	26	25	24
PR[15:8]							

23	22	21	20	19	18	17	16
PR[7:0]							

15	14	13	12	11	10	9	8
VERSION[15:8]							

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **PR: Part Reference**

  Reserved. No functionality associated.
- **VERSION: Version Number**

  Version number of the module. No functionality associated.

24.8 Module Configuration

The specific configuration for each MACB instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 24-7. Module Clock Name

Module name	Clock Name	Description
MACB	CLK_MACB_HSB	HSB clock
	CLK_MACB_PB	Peripheral Bus clock from the PBB clock domain

Table 24-8. Register Reset Values

Module name	Clock name
VERSION	0x0000101D

25. Universal Synchronous Asynchronous Receiver Transmitter (USART)

Rev: 6.0.2.1

25.1 Features

- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by 16 Over-sampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Optional Modem Signal Management DTR-DSR-DCD-RI
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- RS485 with Driver Control Signal
- ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards
  - NACK Handling, Error Counter with Repetition and Iteration Limit
- IrDA Modulation and Demodulation
  - Communication at up to 115.2 Kbps
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (CLK) Frequency up to Internal Clock Frequency CLK_USART/4
- LIN Mode
  - Compliant with LIN 1.3 and LIN 2.0 specifications
  - Master or Slave
  - Processing of frames with up to 256 data bytes
  - Response Data length can be configurable or defined automatically by the Identifier
  - Self synchronization in Slave node configuration
  - Automatic processing and verification of the “Synch Break” and the “Synch Field”
  - The “Synch Break” is detected even if it is partially superimposed with a data byte
  - Automatic Identifier parity calculation/sending and verification
  - Parity sending and verification can be disabled
  - Automatic Checksum calculation/sending and verification
  - Checksum sending and verification can be disabled
  - Support both “Classic” and “Enhanced” checksum types
  - Full LIN error checking and reporting
  - Frame Slot Mode: the Master allocates slots to the scheduled frames automatically.
  - Generation of the Wakeup signal
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of Two Peripheral DMA Controller Channels (PDCA)
  - Offers Buffer Transfer without Processor Intervention

25.2 Overview

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programma-

ble (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485, LIN and SPI buses, with ISO7816 T = 0 or T = 1 smart card slots, infrared transceivers and connection to modem ports. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the Peripheral DMA Controller, which enables data transfers to the transmitter and from the receiver. The Peripheral DMA Controller provides chained buffer management without any intervention of the processor.

25.3 Block Diagram

Figure 25-1. USART Block Diagram

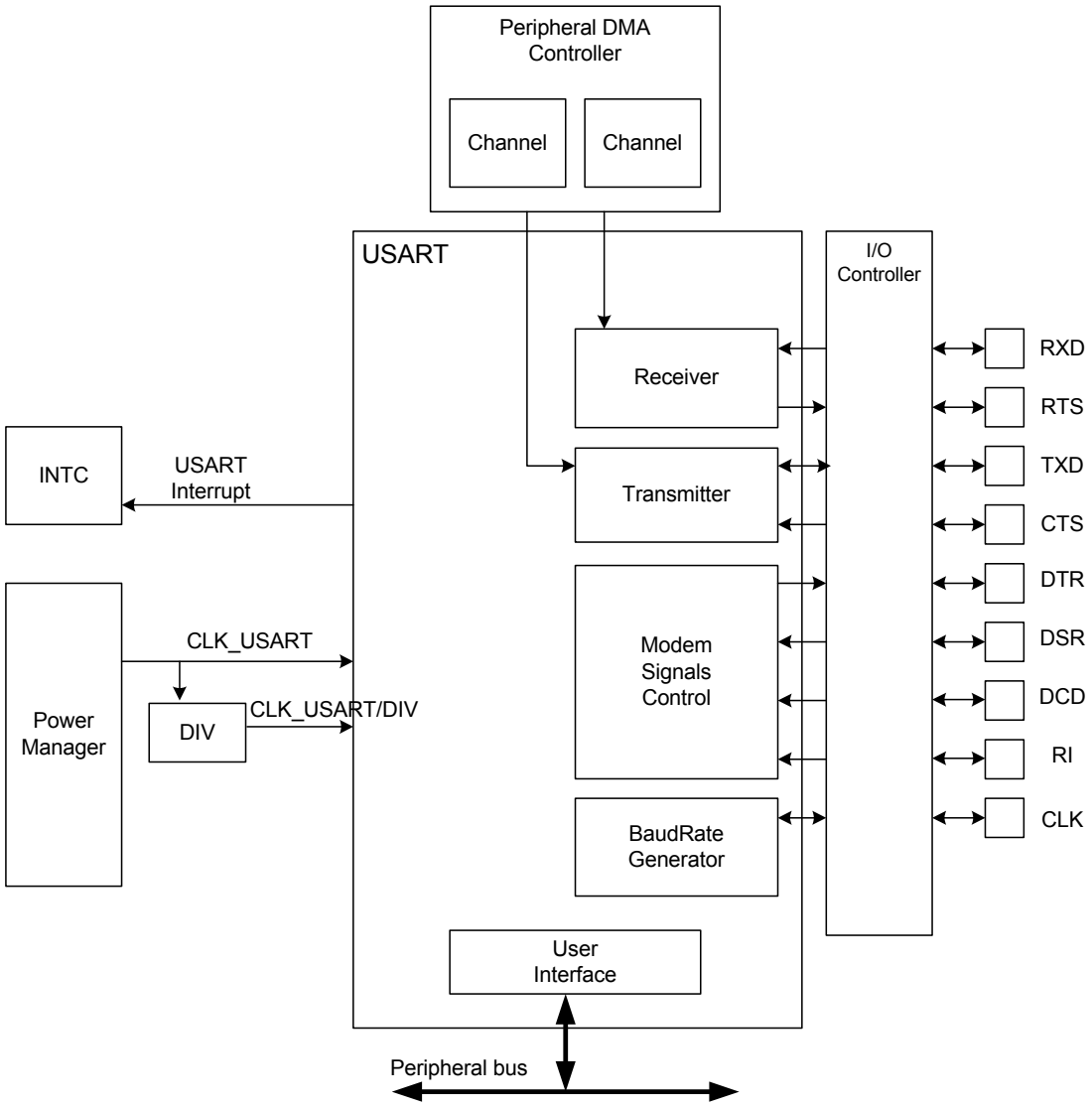


Table 25-1. SPI Operating Mode

PIN	USART	SPI Slave	SPI Master
RXD	RXD	MOSI	MISO
TXD	TXD	MISO	MOSI
RTS	RTS	–	CS
CTS	CTS	CS	–

25.4 I/O Lines Description

Table 25-2. I/O Lines Description

Name	Description	Type	Active Level
CLK	Serial Clock	I/O	
TXD	Transmit Serial Data or Master Out Slave In (MOSI) in SPI Master Mode or Master In Slave Out (MISO) in SPI Slave Mode	Output	
RXD	Receive Serial Data or Master In Slave Out (MISO) in SPI Master Mode or Master Out Slave In (MOSI) in SPI Slave Mode	Input	
RI	Ring Indicator	Input	Low
DSR	Data Set Ready	Input	Low
DCD	Data Carrier Detect	Input	Low
DTR	Data Terminal Ready	Output	Low
CTS	Clear to Send or Slave Select (NSS) in SPI Slave Mode	Input	Low
RTS	Request to Send or Slave Select (NSS) in SPI Master Mode	Output	Low

25.5 Product Dependencies

25.5.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the I/O Controller lines. The programmer must first program the I/O Controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the I/O Controller.

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory. If the hardware handshaking feature or Modem mode is used, the internal pull up on TXD must also be enabled.

All the pins of the modems may or may not be implemented on the USART. On USARTs not equipped with the corresponding pins, the associated control bits and statuses have no effect on the behavior of the USART.

25.5.2 Clocks

The clock for the USART bus interface (CLK_USART) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the USART before disabling the clock, to avoid freezing the USART in an undefined state.

25.5.3 Interrupts

The USART interrupt request line is connected to the interrupt controller. Using the USART interrupt requires the interrupt controller to be programmed first.

25.6 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- High-speed 5- to 9-bit full-duplex synchronous serial communication
  - MSB- or LSB-first
  - 1 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- RS485 with driver control signal
- ISO7816, T0 or T1 protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit, inverted data
- InfraRed IrDA Modulation and Demodulation
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (CLK) Frequency up to Internal Clock Frequency CLK_USART/4
- LIN Mode
  - Compliant with LIN 1.3 and LIN 2.0 specifications
  - Master or Slave
  - Processing of frames with up to 256 data bytes
  - Response Data length can be configurable or defined automatically by the Identifier
  - Self synchronization in Slave node configuration
  - Automatic processing and verification of the “Synch Break” and the “Synch Field”
  - The “Synch Break” is detected even if it is partially superimposed with a data byte
  - Automatic Identifier parity calculation/sending and verification
  - Parity sending and verification can be disabled
  - Automatic Checksum calculation/sending and verification
  - Checksum sending and verification can be disabled
  - Support both “Classic” and “Enhanced” checksum types

- Full LIN error checking and reporting
- Frame Slot Mode: the Master allocates slots to the scheduled frames automatically.
- Generation of the Wakeup signal

• Test modes
- Remote loopback, local loopback, automatic echo

25.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

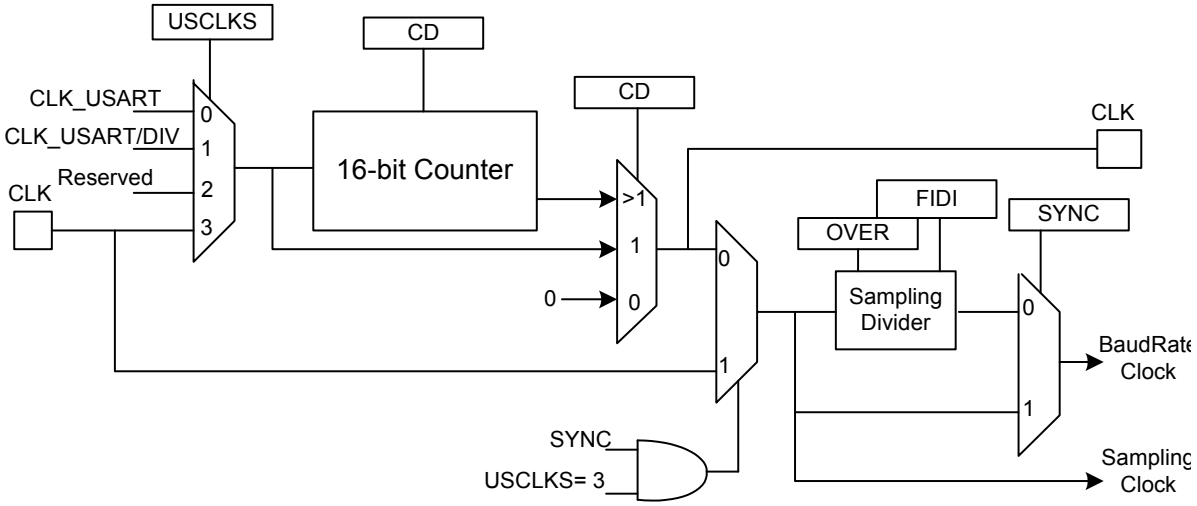
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (MR) between:

- CLK_USART
- a division of CLK_USART, the divider being product dependent, but generally set to 8
- the external clock, available on the CLK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (BRGR). If CD is programmed at 0, the Baud Rate Generator does not generate any clock. If CD is programmed at 1, the divider is bypassed and becomes inactive.

If the external CLK clock is selected, the duration of the low and high levels of the signal provided on the CLK pin must be longer than a CLK_USART period. The frequency of the signal provided on CLK must be at least 4.5 times lower than CLK_USART.

Figure 25-2. Baud Rate Generator



25.6.1.1 *Baud Rate in Asynchronous Mode*

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2-Over)CD)}$$

This gives a maximum baud rate of CLK_USART divided by 8, assuming that CLK_USART is the highest possible clock and that OVER is programmed at 1.

25.6.1.2 Baud Rate Calculation Example

Table 25-3 shows calculations of CD to obtain a baud rate at 38400 bauds for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 25-3. Baud Rate Example (OVER = 0)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%
60 000 000	38 400	97.66	98	38 265.31	0.35%

The baud rate is calculated with the following formula:

$$BaudRate = (CLKUSART)/(CD \times 16)$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

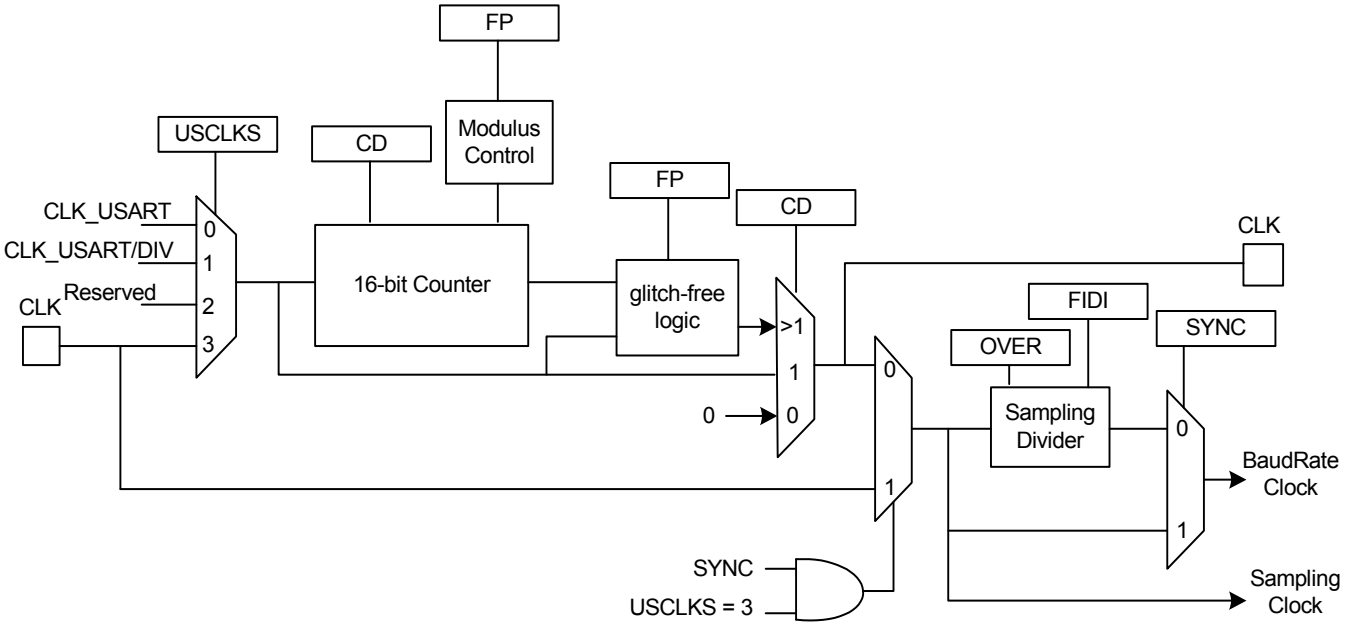
25.6.1.3 Fractional Baud Rate in Asynchronous Mode

The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain Baud Rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in the Baud Rate Generator Register (BRGR). If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. This feature is only available when using USART normal mode. The fractional Baud Rate is calculated using the following formula:

$$Baudrate = \frac{SelectedClock}{\left(8(2 - Over)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented below:

Figure 25-3. Fractional Baud Rate Generator



25.6.1.4 Baud Rate in Synchronous Mode or SPI Mode

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART CLK pin. No division is active. The value written in BRGR has no effect. The external clock frequency must be at least 4.5 times lower than the system clock.

When either the external clock CLK or the internal clock divided (CLK_USART/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the CLK pin. If the internal clock CLK_USART is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the CLK pin, even if the value programmed in CD is odd.

25.6.1.5 *Baud Rate in ISO 7816 Mode*

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 25-4.

Table 25-4. Binary and Decimal Values for Di

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 25-5.

Table 25-5. Binary and Decimal Values for Fi

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 25-6 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 25-6. Possible Values for the Fi/Di Ratio

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (MR) is first divided by the value programmed in the field CD in the Baud Rate

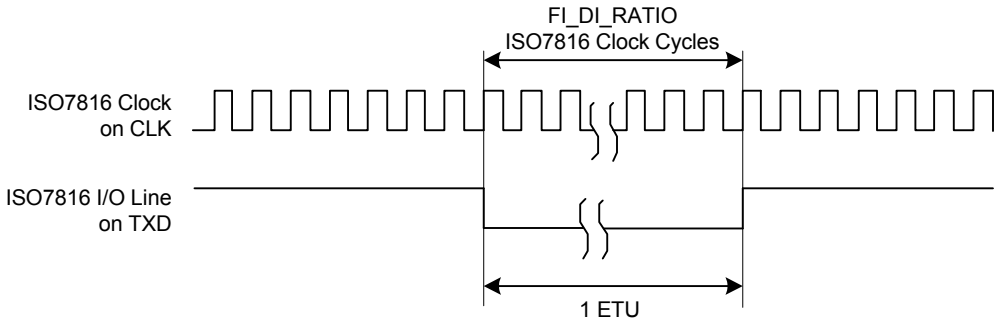
Generator Register (BRGR). The resulting clock can be provided to the CLK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 Mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 25-4 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 25-4. Elementary Time Unit (ETU)



25.6.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the Control Register (CR). However, the transmitter registers can be programmed before being enabled.

The Receiver and the Transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the Control Register (CR). The software resets clear the status flag and reset internal state machines but the user interface configuration registers hold the value configured prior to software reset. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding Register (THR). If a time-guard is programmed, it is handled normally.

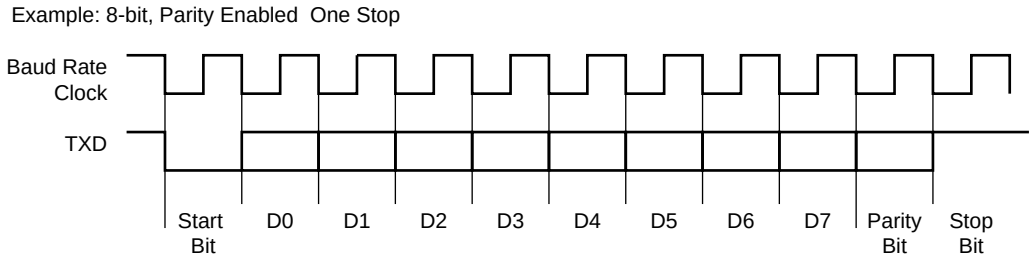
25.6.3 Synchronous and Asynchronous Modes

25.6.3.1 Transmitter Operations

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (MR). Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in MR configures which data bit is sent first. If written at 1, the most significant bit is sent first. At 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in MR. The 1.5 stop bit is supported in asynchronous mode only.

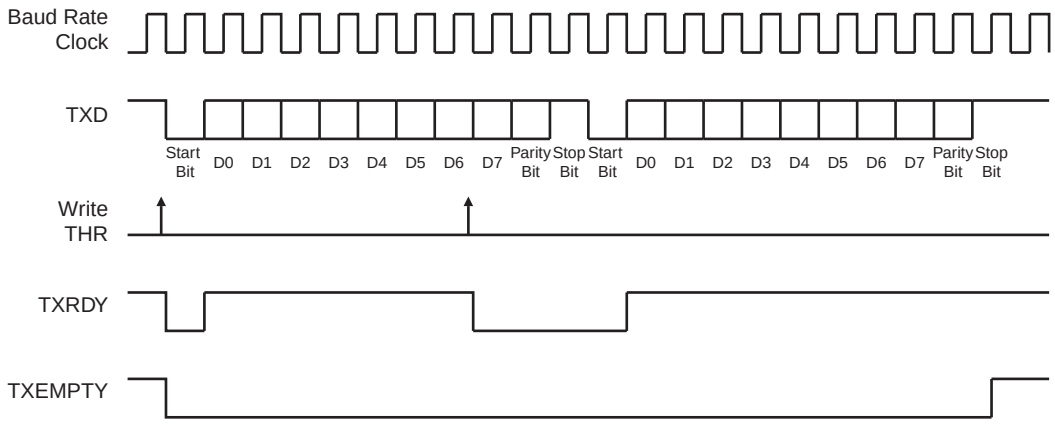
Figure 25-5. Character Transmit



The characters are sent by writing in the Transmit Holding Register (THR). The transmitter reports two status bits in the Channel Status Register (CSR): TXRDY (Transmitter Ready), which indicates that THR is empty and TXEMPTY, which indicates that all the characters written in THR have been processed. When the current character processing is completed, the last character written in THR is transferred into the Shift Register of the transmitter and THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in THR while TXRDY is low has no effect and the written character is lost.

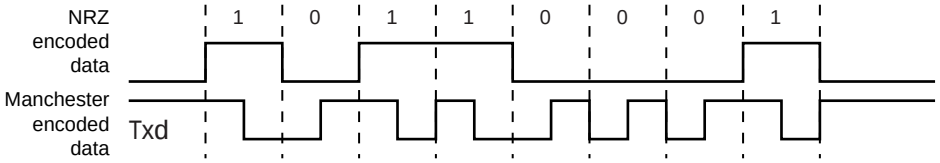
Figure 25-6. Transmitter Status



25.6.3.2 Manchester Encoder

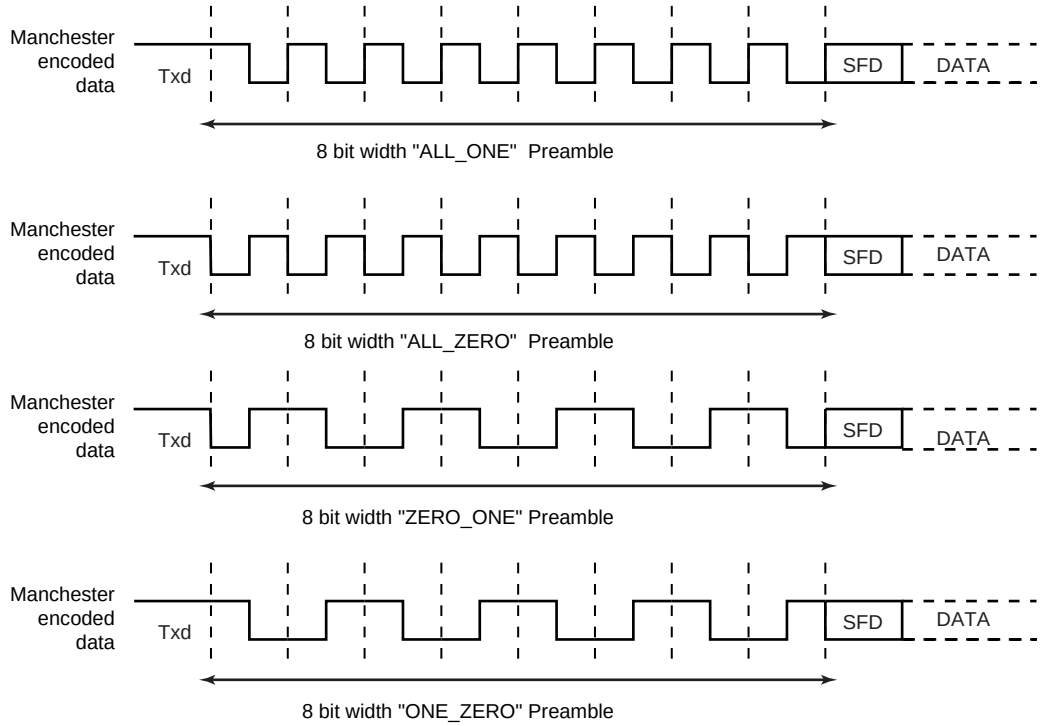
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero or zero-to-one. Thus, a transition always occurs at the midpoint of each bit time. It consumes more bandwidth than the original NRZ signal (2x) but the receiver has more error control since the expected input must show a change at the center of a bit cell. An example of Manchester encoded sequence is: the byte 0xB1 or 10110001 encodes to 10 01 10 10 01 01 01 10, assuming the default polarity of the encoder. Figure 25-7 illustrates this coding scheme.

Figure 25-7. NRZ to Manchester Encoding



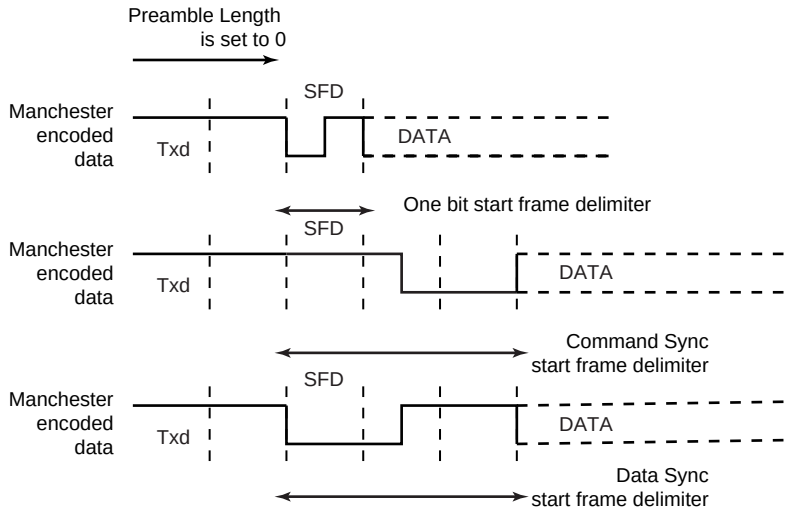
The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the MAN register, the field TX_PL is used to configure the preamble length. Figure 25-8 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to one, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

Figure 25-8. Preamble Patterns, Default Polarity Assumed



A start frame delimiter is to be configured using the ONEBIT field in the MR register. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 25-9 illustrates these patterns. If the start frame delimiter, also known as start bit, is one bit, (ONEBIT at 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT at 0), a sequence of 3 bit times is sent serially on the line to indicate the start of a new character. The sync waveform is in itself an invalid Manchester waveform as the transition occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC field in the MR register is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC field in MR register must be set to 1. In this case, the MODSYNC field in MR is bypassed and the sync configuration is held in the TXSYNH in the THR register. The USART character format is modified and includes sync information.

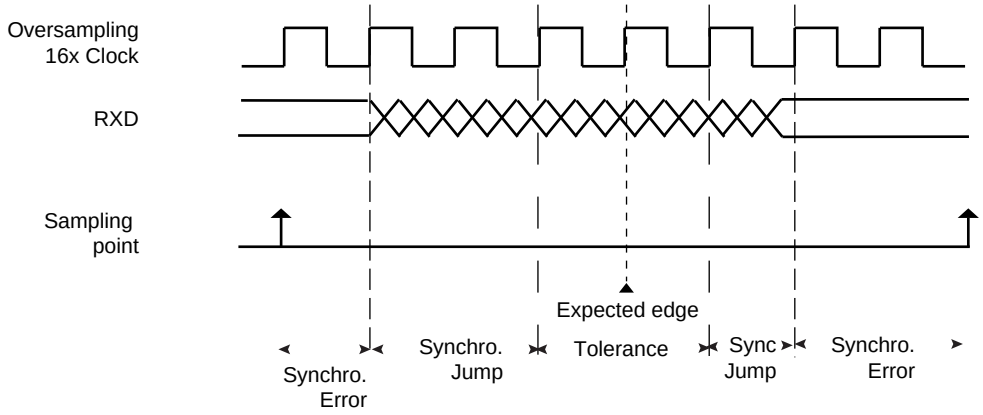
Figure 25-9. Start Frame Delimiter



Drift Compensation

Drift compensation is available only in 16X oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

Figure 25-10. Bit Resynchronization



25.6.3.3 Asynchronous Receiver

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (MR).

The receiver samples the RXD line. If the line is sampled during one half of a bit time at 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 8 sampling clock cycle.

The number of data bits, first bit sent and parity mode are selected by the same fields and bits as the transmitter, i.e. respectively CHRL, MODE9, MSBF and PAR. For the synchronization mechanism **only**, the number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 25-11 and Figure 25-12 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 25-11. Asynchronous Start Detection

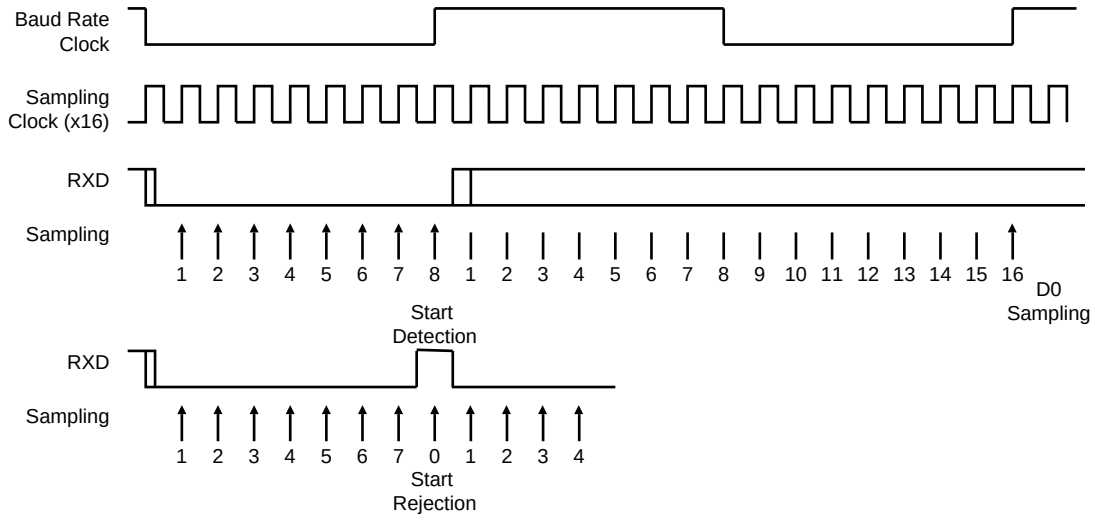
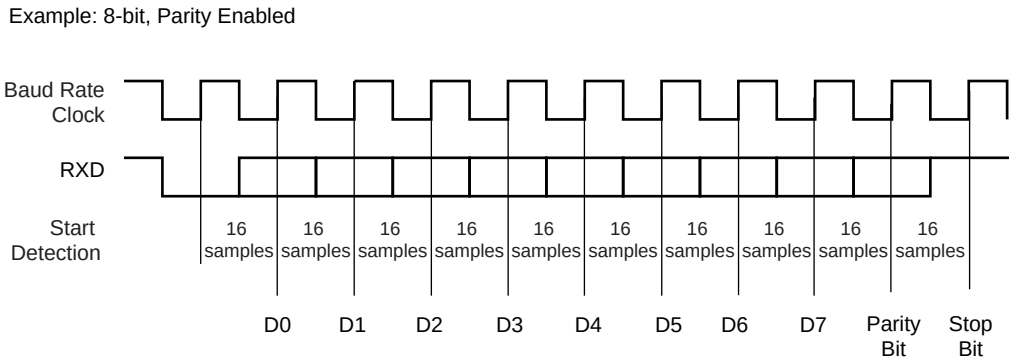


Figure 25-12. Asynchronous Character Reception



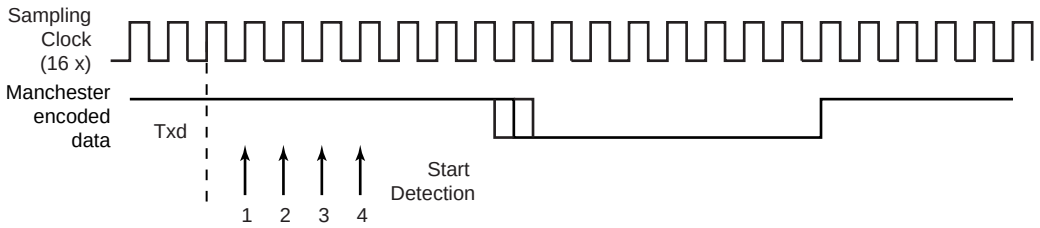
25.6.3.4 Manchester Decoder

When the MAN field in MR register is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

An optional preamble sequence can be defined, its length is user-defined and totally independent of the emitter side. Use RX_PL in MAN register to configure the length of the preamble sequence. If the length is set to 0, no preamble is detected and the function is disabled. In addition, the polarity of the input stream is programmable with RX_MPOL field in MAN register. Depending on the desired application the preamble pattern matching is to be defined via the RX_PP field in MAN. See Figure 25-8 for available preamble patterns.

Unlike preamble, the start frame delimiter is shared between Manchester Encoder and Decoder. So, if ONEBIT field is set to 1, only a zero encoded Manchester can be detected as a valid start frame delimiter. If ONEBIT is set to 0, only a sync pattern is detected as a valid start frame delimiter. Decoder operates by detecting transition on incoming stream. If RXD is sampled during one quarter of a bit time at zero, a start bit is detected. See Figure 25-13. The sample pulse rejection mechanism applies.

Figure 25-13. Asynchronous Start Bit Detection



The receiver is activated and starts Preamble and Frame Delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid pattern or a valid start frame delimiter, the receiver re-synchronizes on the next valid edge.The minimum time threshold to estimate the bit value is three quarters of a bit time.

If a valid preamble (if used) followed with a valid start frame delimiter is detected, the incoming stream is decoded into NRZ data and passed to USART for processing. Figure 25-14 illustrates Manchester pattern mismatch. When incoming data stream is passed to the USART, the receiver is also able to detect Manchester code violation. A code violation is a lack of transition in the middle of a bit cell. In this case, MANE flag in CSR register is raised. It is cleared by writing the Control Register (CR) with the RSTSTA bit at 1. See Figure 25-15 for an example of Manchester error detection during data phase.

Figure 25-14. Preamble Pattern Mismatch

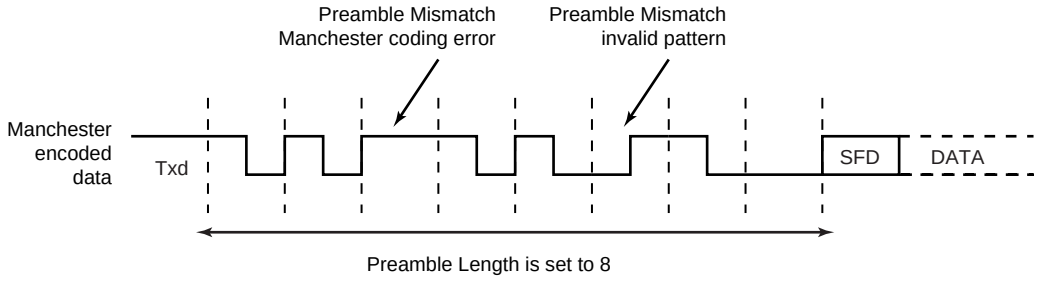
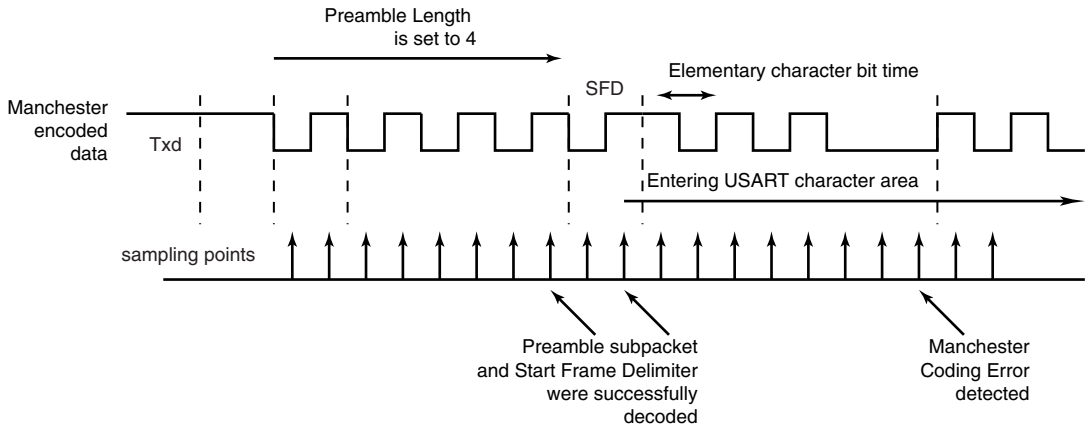


Figure 25-15. Manchester Error Flag



When the start frame delimiter is a sync pattern (ONEBIT field at 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR field in the RHR register and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

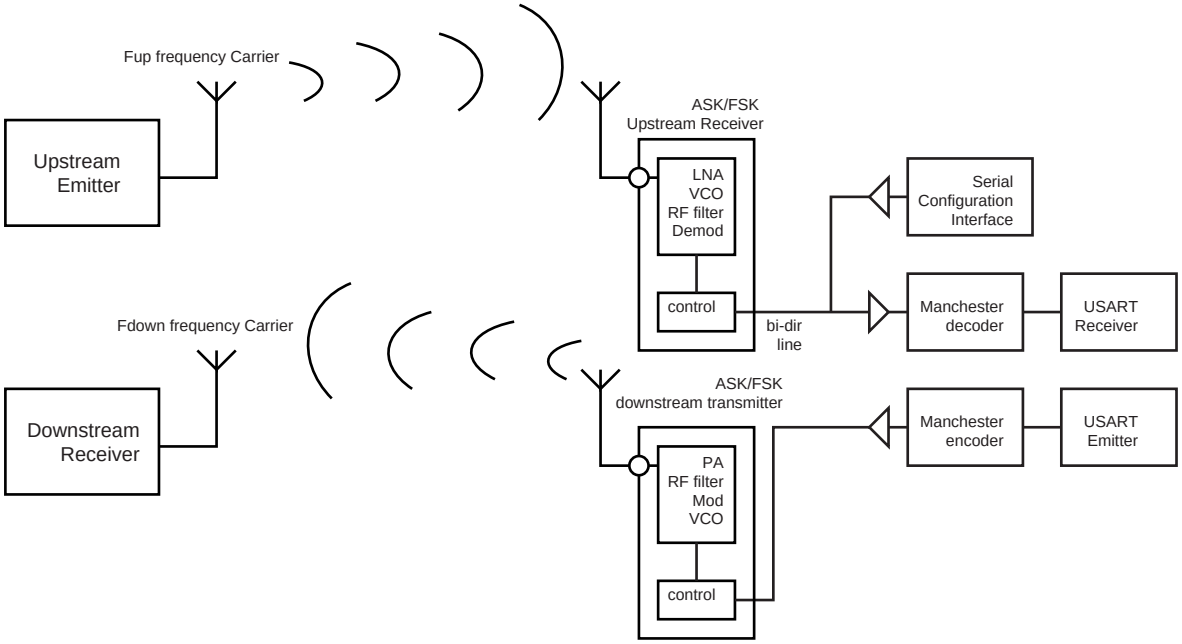
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

25.6.3.5 Radio Interface: Manchester Encoded USART Application

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 25-16.

Figure 25-16. Manchester Encoded Characters RF Transmission



The USART module is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 25-17 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 25-18.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver switches to receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

Figure 25-17. ASK Modulator Output

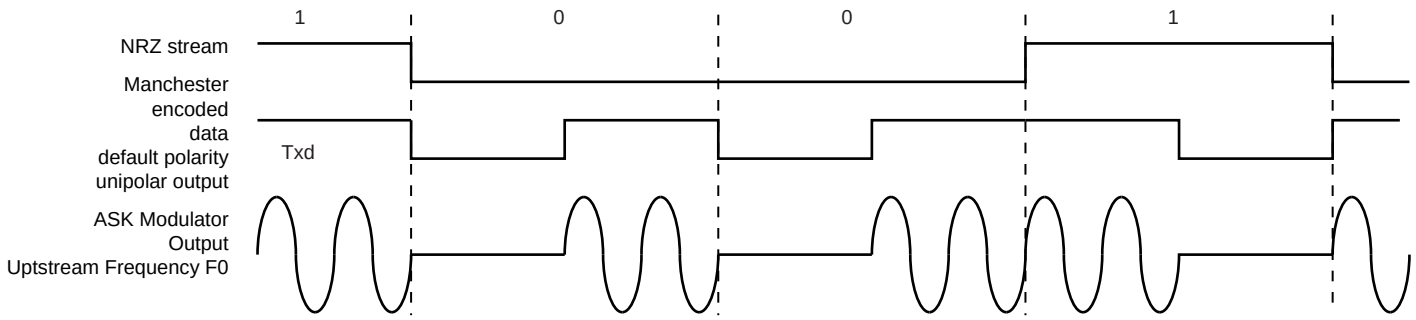
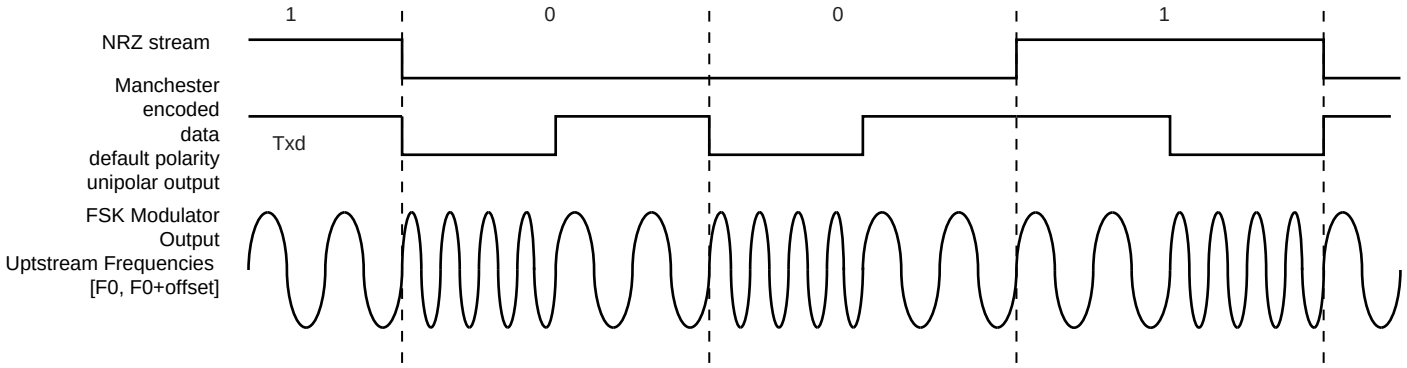


Figure 25-18. FSK Modulator Output



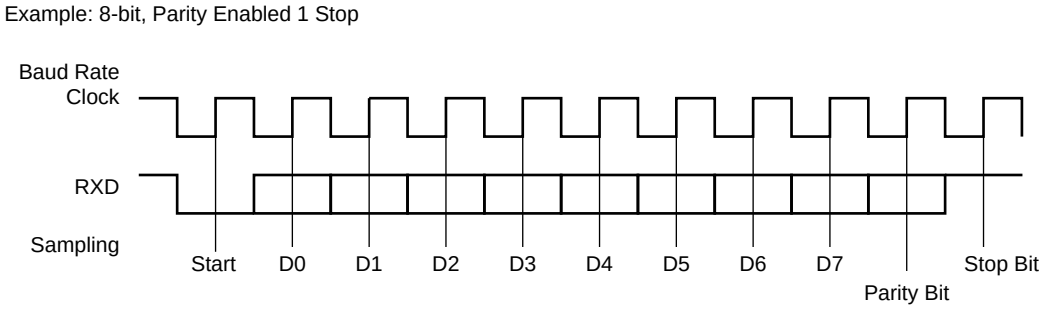
25.6.3.6 *Synchronous Receiver*

In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 25-19 illustrates a character reception in synchronous mode.

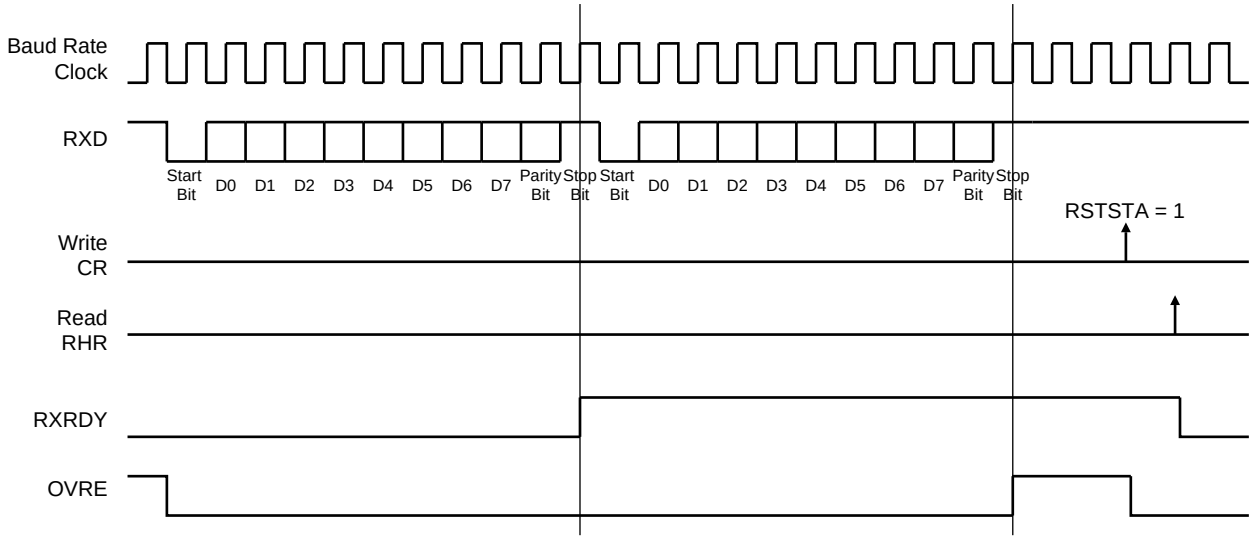
Figure 25-19. Synchronous Mode Character Reception



25.6.3.7 *Receiver Operations*

When a character reception is completed, it is transferred to the Receive Holding Register (RHR) and the RXRDY bit in the Status Register (CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (CR) with the RSTSTA (Reset Status) bit at 1.

Figure 25-20. Receiver Status



25.6.3.8 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (MR). The PAR field also enables the Multidrop mode, see "Multidrop Mode" on page 584. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit at 0 if a number of 1s in the character data bit is even, and at 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit at 1 if a number of 1s in the character data bit is even, and at 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit at 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 0. If the space parity is used, the parity generator of the transmitter drives the parity bit at 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

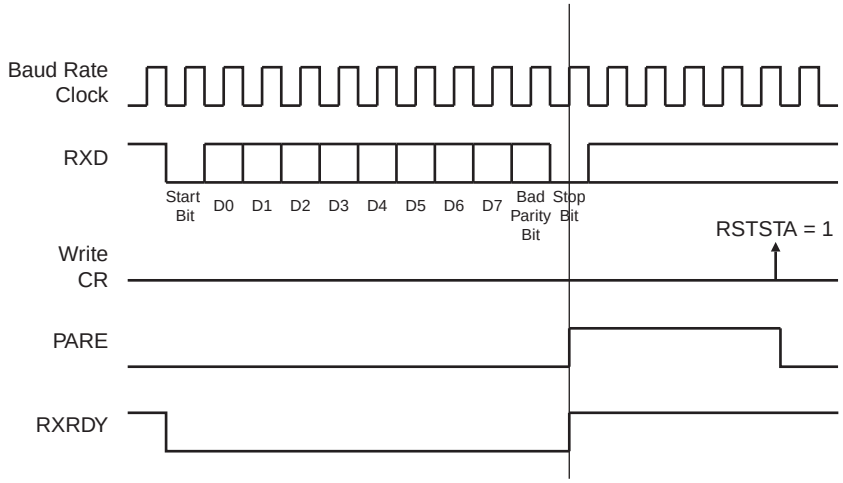
Table 25-7 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 25-7. Parity Bit Examples

Character	Hexa	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (CSR). The PARE bit can be cleared by writing the Control Register (CR) with the RSTSTA bit at 1. Figure 25-21 illustrates the parity bit status setting and clearing.

Figure 25-21. Parity Error



25.6.3.9 *Multidrop Mode*

If the PAR field in the Mode Register (MR) is programmed to the value 0x6 or 0x07, the USART runs in Multidrop Mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in multidrop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when the Control Register is written with the SENDA bit at 1.

To handle parity error, the PARE bit is cleared when the Control Register is written with the bit RSTSTA at 1.

The transmitter sends an address byte (parity bit set) when SENDA is written to CR. In this case, the next byte written to THR is transmitted as an address. Any character written in THR without having written the command SENDA is transmitted normally with the parity at 0.

25.6.3.10 *Transmitter Timeguard*

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (TTGR). When this field is programmed at zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 25-22, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains at 0 during the timeguard transmission if a character has been written in THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

Figure 25-22. Timeguard Operations

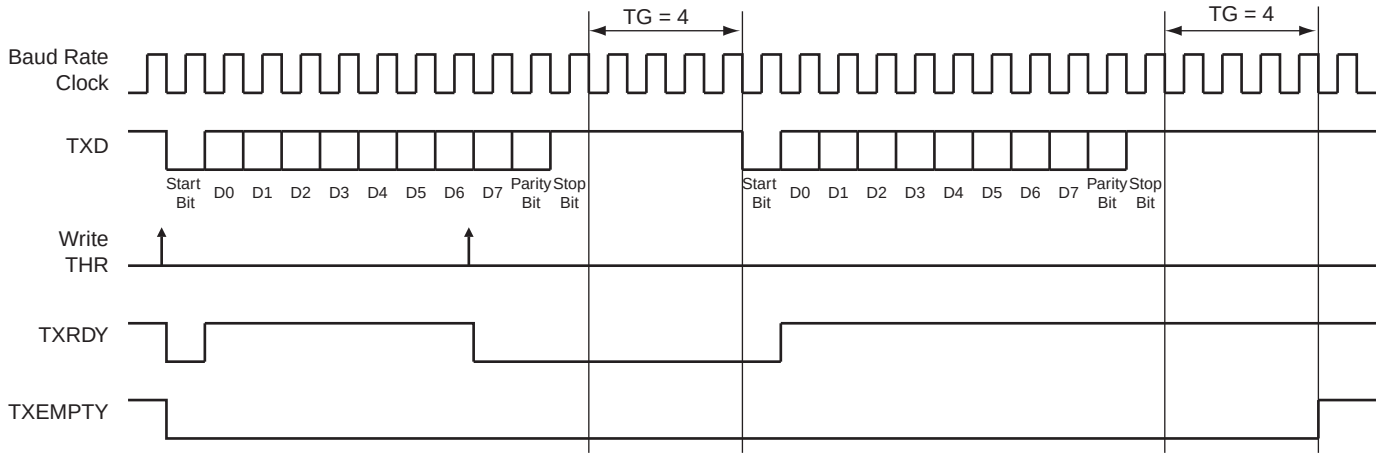


Table 25-8 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 25-8. Maximum Timeguard Length Depending on Baud Rate

Baud Rate	Bit time	Timeguard
Bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

25.6.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (RTOR). If the TO field is programmed at 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in CSR remains at 0. Otherwise, the receiver loads a counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing the Control Register (CR) with the STTTO (Start Time-out) bit at 1. In this case, the idle state on RXD before a new character is received will not provide a time-out. This prevents having to

handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.

- Obtain an interrupt while no character is received. This is performed by writing CR with the RETTO (Reload and Start Time-out) bit at 1. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 25-23 shows the block diagram of the Receiver Time-out feature.

Figure 25-23. Receiver Time-out Block Diagram

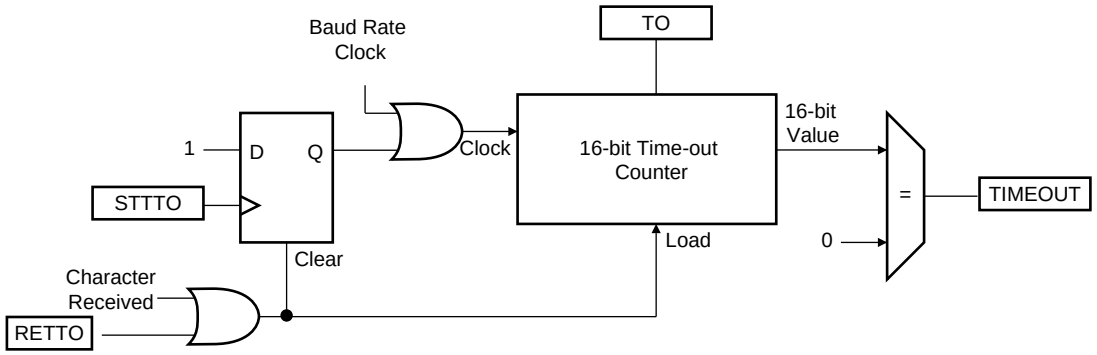


Table 25-9 gives the maximum time-out period for some standard baud rates.

Table 25-9. Maximum Time-out Period

Baud Rate	Bit Time	Time-out
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962

Table 25-9. Maximum Time-out Period (Continued)

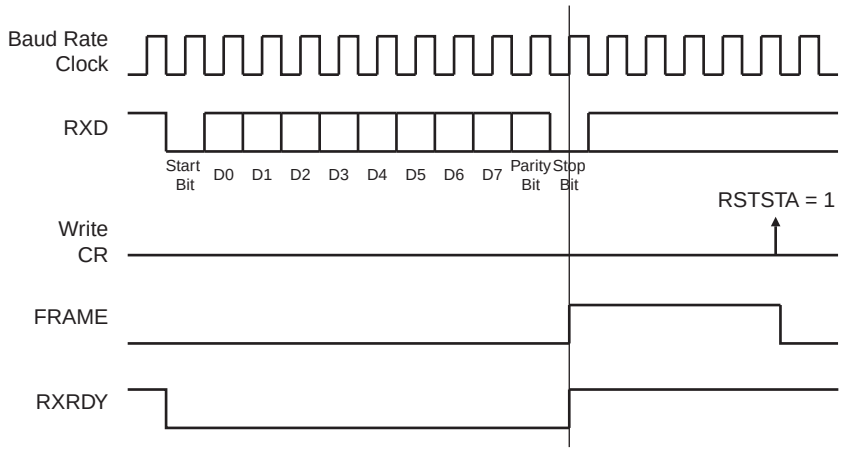
Baud Rate	Bit Time	Time-out
56000	18	1 170
57600	17	1 138
200000	5	328

25.6.3.12 *Framing Error*

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (CSR). The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing the Control Register (CR) with the RSTSTA bit at 1.

Figure 25-24. Framing Error Status



25.6.3.13 *Transmit Break*

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (CR) with the STTBRK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing CR with the STPBRK bit at 1. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBRK and STPBRK commands are taken into account only if the TXRDY bit in CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

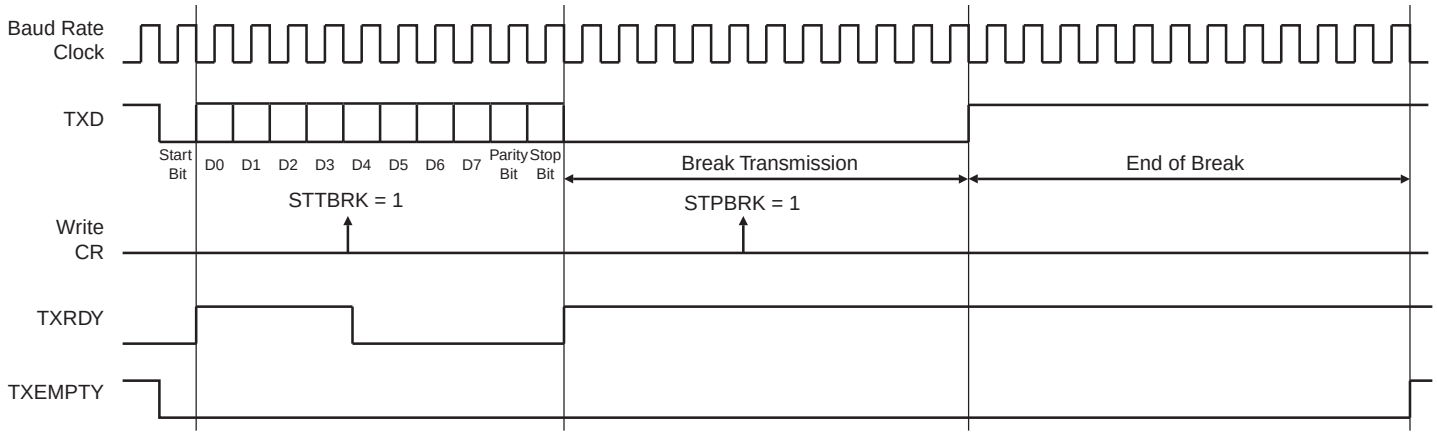
Writing CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 25-25 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 25-25. Break Transmission



25.6.3.14 Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data at 0x00, but FRAME remains low.

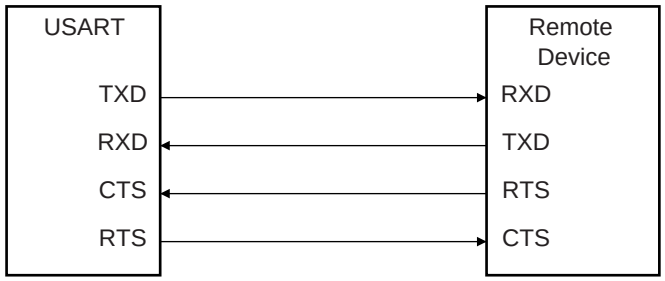
When the low stop bit is detected, the receiver asserts the RXBRK bit in CSR. This bit may be cleared by writing the Control Register (CR) with the bit RSTSTA at 1.

An end of receive break is detected by a high level for at least 2/16 of a bit period in asynchronous operating mode or one sample at high level in synchronous operating mode. The end of break detection also asserts the RXBRK bit.

25.6.3.15 Hardware Handshaking

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 25-26.

Figure 25-26. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the MODE field in the Mode Register (MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the Peripheral DMA Controller channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 25-27 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the Peripheral DMA Controller channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the Peripheral DMA Controller clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

Figure 25-27. Receiver Behavior when Operating with Hardware Handshaking

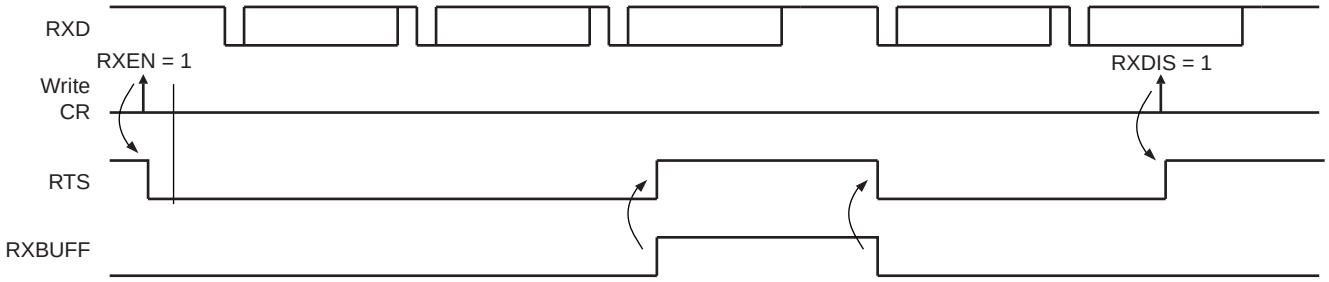
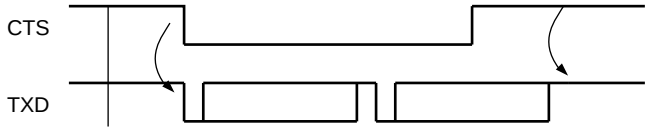


Figure 25-28 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 25-28. Transmitter Behavior when Operating with Hardware Handshaking



25.6.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

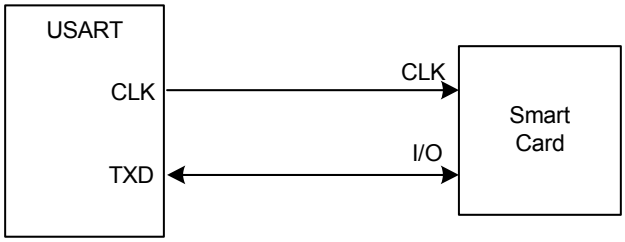
Setting the USART in ISO7816 mode is performed by writing the MODE field in the Mode Register (MR) to the value 0x4 for protocol T = 0 and to the value 0x6 for protocol T = 1.

25.6.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see "Baud Rate Generator" on page 568).

The USART connects to a smart card as shown in Figure 25-29. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the CLK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

Figure 25-29. Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first. Parity Bit (PAR) can be used to transmit in normal or inverse mode. Refer to "Mode Register" on page 628 and "PAR: Parity Type" on page 630.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (THR) or after reading it in the Receive Holding Register (RHR).

25.6.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 25-30.

If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 25-31. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (RHR). It appropriately sets the PARE bit in the Status Register (SR) so that the software can handle the error.

Figure 25-30. T = 0 Protocol without Parity Error

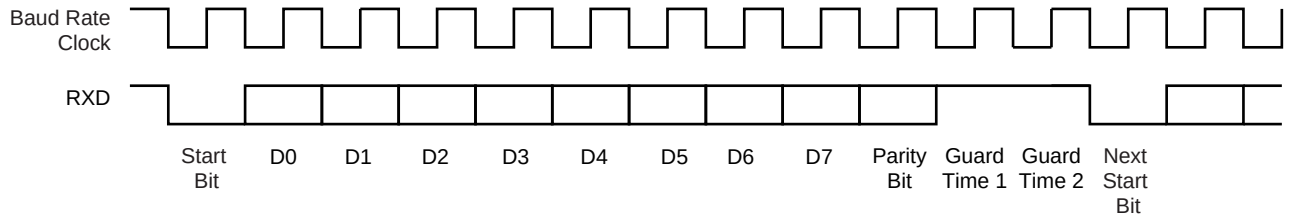
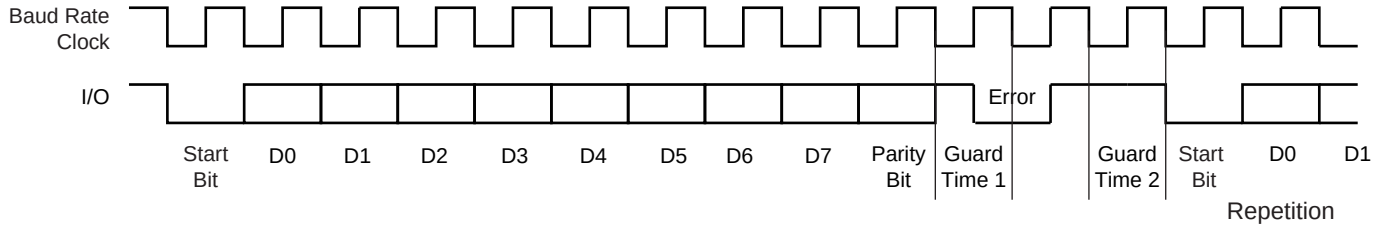


Figure 25-31. T = 0 Protocol with Parity Error



25.6.4.3 Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (NER) register. The NB_ERRORS field can record up to 255 errors. Reading NER automatically clears the NB_ERRORS field.

25.6.4.4 Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (MR). If INACK is at 1, no error signal is driven on the I/O line even if a parity bit is detected.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred. However, the RXRDY bit does raise.

25.6.4.5 Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the Mode Register (MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION, the ITERATION bit is set in the Channel Status Register (CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in CSR can be cleared by writing the Control Register with the RSIT bit at 1.

25.6.4.6 Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (MR). The maximum number of NACK transmitted is programmed in the MAX_ITERATION field. As soon as MAX_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

25.6.4.7 Protocol T = 1

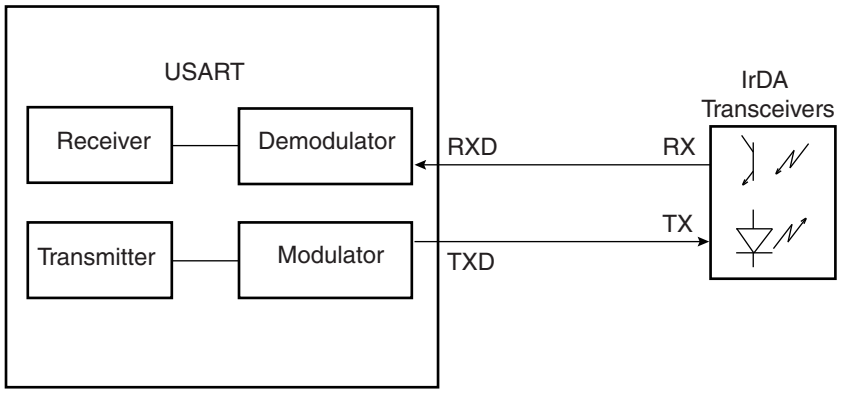
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (CSR).

25.6.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 25-32. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 Kb/s to 115.2 Kb/s.

The USART IrDA mode is enabled by setting the MODE field in the Mode Register (MR) to the value 0x8. The IrDA Filter Register (IFR) allows configuring the demodulator filter. The USART transmitter and receiver operate in a normal asynchronous mode and all parameters are accessible (except those fixed by IrDA specification: one start bit, 8 data bits and one stop bit). Note that the modulator and the demodulator are activated.

Figure 25-32. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

To receive IrDA signals, the following needs to be done:

- Disable TX and Enable RX
- Configure the TXD pin as I/O and set it as an output at 0 (to avoid LED emission). Disable the internal pull-up (better for power consumption).
- Receive data

25.6.5.1 IrDA Modulation

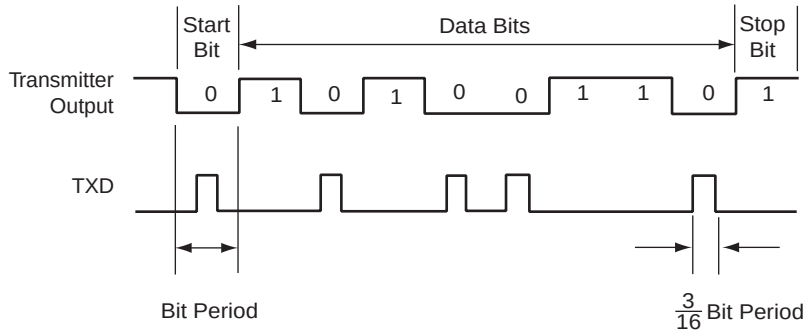
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 25-10.

Table 25-10. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 25-33 shows an example of character transmission.

Figure 25-33. IrDA Modulation



25.6.5.2 IrDA Baud Rate

Table 25-11 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 25-11. IrDA Baud Rate Error

Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88

Table 25-11. IrDA Baud Rate Error (Continued)

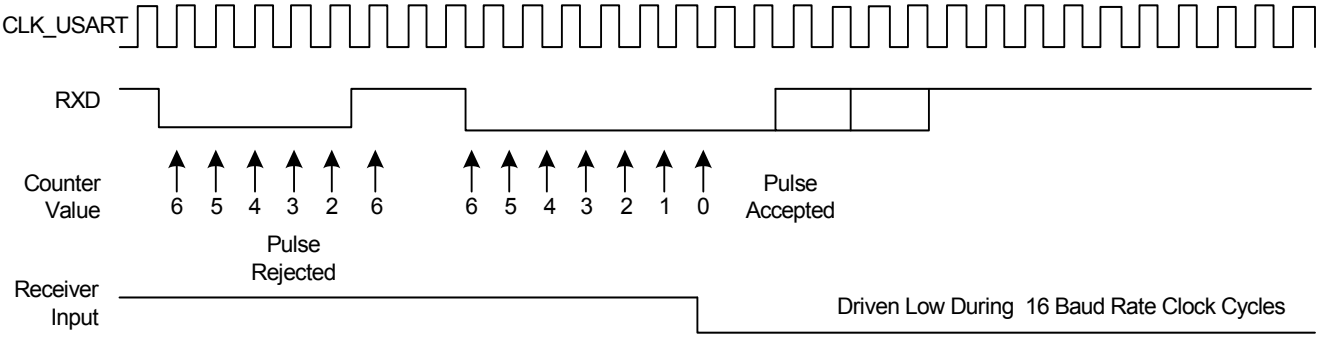
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

25.6.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in IFR. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the CLK_USART speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with IFR. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 25-34 illustrates the operations of the IrDA demodulator.

Figure 25-34. IrDA Demodulator Operations

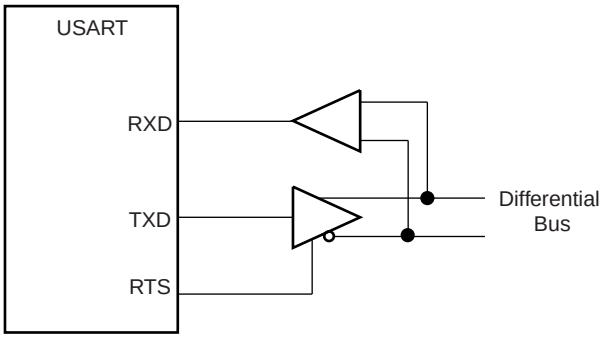


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

25.6.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 25-35.

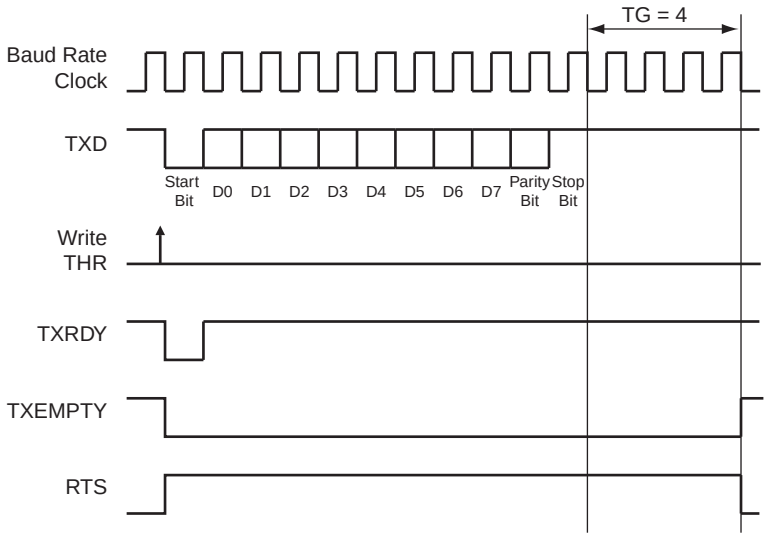
Figure 25-35. Typical Connection to a RS485 Bus



The USART is set in RS485 mode by programming the MODE field in the Mode Register (MR) to the value 0x1.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 25-36 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 25-36. Example of RTS Drive with Timeguard



25.6.7 Modem Mode

The USART features modem mode, which enables control of the signals: DTR (Data Terminal Ready), DSR (Data Set Ready), RTS (Request to Send), CTS (Clear to Send), DCD (Data Carrier Detect) and RI (Ring Indicator). While operating in modem mode, the USART behaves as a DTE (Data Terminal Equipment) as it drives DTR and RTS and can detect level change on DSR, DCD, CTS and RI.

Setting the USART in modem mode is performed by writing the MODE field in the Mode Register (MR) to the value 0x3. While operating in modem mode the USART behaves as though in asynchronous mode and all the parameter configurations are available.

Table 25-12 gives the correspondence of the USART signals with modem connection standards.

Table 25-12. Circuit References

USART Pin	V24	CCITT	Direction
TXD	2	103	From terminal to modem
RTS	4	105	From terminal to modem
DTR	20	108.2	From terminal to modem
RXD	3	104	From modem to terminal
CTS	5	106	From terminal to modem
DSR	6	107	From terminal to modem
DCD	8	109	From terminal to modem
RI	22	125	From terminal to modem

The control of the DTR output pin is performed by writing the Control Register (CR) with the DTRDIS and DTREN bits respectively at 1. The disable command forces the corresponding pin to its inactive level, i.e. high. The enable command forces the corresponding pin to its active level, i.e. low. RTS output pin is automatically controlled in this mode

The level changes are detected on the RI, DSR, DCD and CTS pins. If an input change is detected, the RIIC, DSRIC, DCDIC and CTSIC bits in the Channel Status Register (CSR) are set respectively and can trigger an interrupt. The status is automatically cleared when CSR is read. Furthermore, the CTS automatically disables the transmitter when it is detected at its inactive state. If a character is being transmitted when the CTS rises, the character transmission is completed before the transmitter is actually disabled.

25.6.8 SPI Mode

The Serial Peripheral Interface (SPI) Mode is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master” which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turns being masters and one master may simultaneously shift data into multiple slaves. (Multiple Master Protocol is the opposite of Single Master Protocol, where one CPU is always the master while all of the others are always slaves.) However, only one slave may drive its output to write data back to the master at any given time.

A slave device is selected when its NSS signal is asserted by the master. The USART in SPI Master mode can address only one SPI Slave because it can generate only one NSS signal. Nevertheless the user can use standard I/O lines to access more than one SPI slave.

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input of the slave.
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master.
- Serial Clock (CLK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates. The CLK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows the master to select or deselect the slave.

25.6.8.1 Modes of Operation

The USART can operate in Master Mode or in Slave Mode.

Operation in SPI Master Mode is programmed by writing at 0xE the MODE field in the Mode Register. In this case the SPI lines must be connected as described below:

- the MOSI line is driven by the output pin TXD
- the MISO line drives the input pin RXD
- the CLK line is driven by the output pin CLK
- the NSS line is driven by the output pin RTS

Operation in SPI Slave Mode is programmed by writing at 0xF the MODE field in the Mode Register. In this case the SPI lines must be connected as described below:

- the MOSI line drives the input pin RXD
- the MISO line is driven by the output pin TXD
- the CLK line drives the input pin CLK
- the NSS line drives the input pin CTS

In order to avoid unpredicted behavior, any change of the SPI Mode must be followed by a software reset of the transmitter and of the receiver (except the initial configuration after a hardware reset).

25.6.8.2 Baud Rate

In SPI Mode, the baudrate generator operates in the same way as in USART synchronous mode: See Section “25.6.1.4” on page 570. However, there are some restrictions:

In SPI Master Mode:

- the external clock CLK must not be selected (USCLKS … 0x3), and the bit CLKO must be set to “1” in the Mode Register (MR), in order to generate correctly the serial clock on the CLK pin.
- to obtain correct behavior of the receiver and the transmitter, the value programmed in CD of must be superior or equal to 4.
- if the internal clock divided (CLK_USART/DIV) is selected, the value programmed in CD must be even to ensure a 50:50 mark/space ratio on the CLK pin, this value can be odd if the internal clock is selected (CLK_USART).

In SPI Slave Mode:

- the external clock (CLK) selection is forced regardless of the value of the USCLKS field in the Mode Register (MR). Likewise, the value written in BRGR has no effect, because the clock is provided directly by the signal on the USART CLK pin.
- to obtain correct behavior of the receiver and the transmitter, the external clock (CLK) frequency must be at least 4 times lower than the system clock.

25.6.8.3 Data Transfer

Up to 9 data bits are successively shifted out on the TXD pin at each rising or falling edge (depending of CPOL and CPHA) of the programmed serial clock. There is no Start bit, no Parity bit and no Stop bit.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (MR). The 9 bits are selected by setting the MODE 9 bit regardless of the CHRL field. The MSB data bit is always sent first in SPI Mode (Master or Slave).

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Mode Register. The clock phase is programmed with the CPHA bit. These two parameters determine the edges of the clock signal upon which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

Table 25-13. SPI Bus Protocol Mode

SPI Bus Protocol Mode	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 25-37. SPI Transfer Format (CPHA=1, 8 bits per transfer)

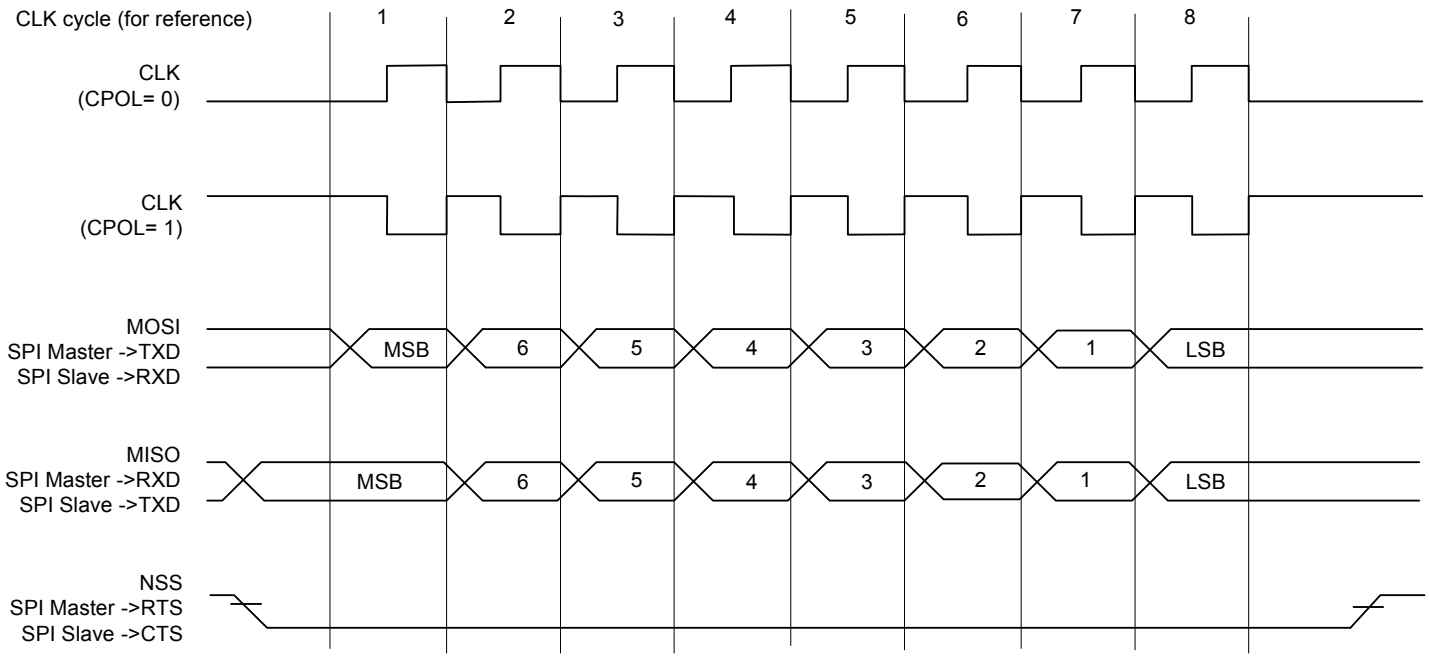
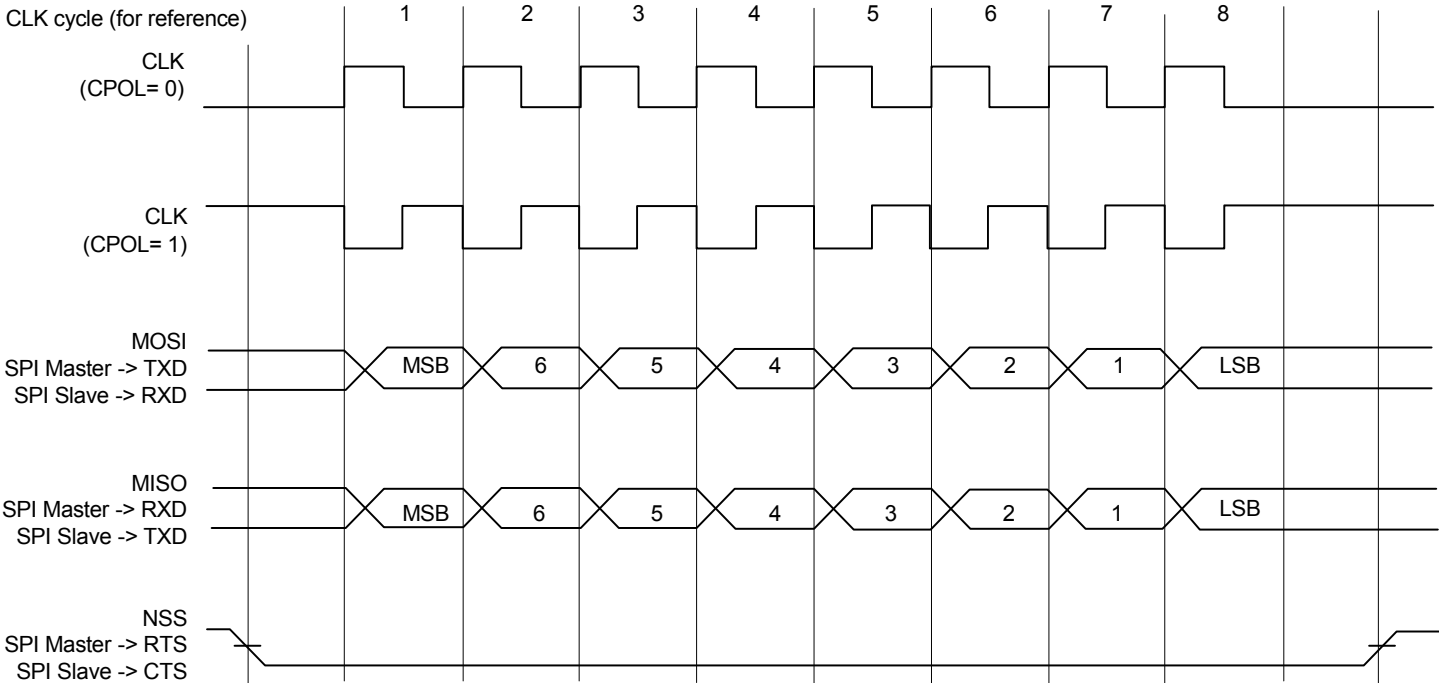


Figure 25-38. SPI Transfer Format (CPHA=0, 8 bits per transfer)



25.6.8.4 Receiver and Transmitter Control

See Section “25.6.2” on page 572.

25.6.8.5 Character Transmission

The characters are sent by writing in the Transmit Holding Register (THR). An additional condition for transmitting a character can be added when the USART is configured in SPI master mode. In the MR register, the value configured on INACK field can prevent any character transmission (even if THR has been written) while the receiver side is not ready (character not read). When INACK equals 0, the character is transmitted whatever the receiver status. If INACK is set to 1, the transmitter waits for the receiver holding register to be read before transmitting the character (RXRDY flag cleared), thus preventing any overflow (character loss) on the receiver side.

The transmitter reports two status bits in the Channel Status Register (CSR): TXRDY (Transmitter Ready), which indicates that THR is empty and TXEMPTY, which indicates that all the characters written in THR have been processed. When the current character processing is completed, the last character written in THR is transferred into the Shift Register of the transmitter and THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in THR while TXRDY is low has no effect and the written character is lost.

If the USART is in SPI Slave Mode and if a character must be sent while the Transmit Holding Register (THR) is empty, the UNRE (Underrun Error) bit is set. The TXD transmission line stays at high level during all this time. The UNRE bit is cleared by writing the Control Register (CR) with the RSTSTA (Reset Status) bit at 1.

In SPI Master Mode, the slave select line (NSS) is asserted at low level 1 Tbit before the transmission of the MSB bit and released at high level 1 Tbit after the transmission of the LSB bit. So, the slave select line (NSS) is always released between each character transmission and a minimum delay of 3 Tbits always inserted. However, in order to address slave devices supporting the CSAAT mode (Chip Select Active After Transfer), the slave select line (NSS) can be forced at low level by writing the Control Register (CR) with the RTSEN bit at 1. The slave select line (NSS) can be released at high level only by writing the Control Register (CR) with the RTSDIS bit at 1 (for example, when all data have been transferred to the slave device).

In SPI Slave Mode, the transmitter does not require a falling edge of the slave select line (NSS) to initiate a character transmission but only a low level. However, this low level must be present on the slave select line (NSS) at least 1 Tbit before the first serial clock cycle corresponding to the MSB bit.

25.6.8.6 Character Reception

When a character reception is completed, it is transferred to the Receive Holding Register (RHR) and the RXRDY bit in the Status Register (CSR) rises. If a character is completed while RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (CR) with the RSTSTA (Reset Status) bit at 1.

To ensure correct behavior of the receiver in SPI Slave Mode, the master device sending the frame must ensure a minimum delay of 1 Tbit between each character transmission. The receiver does not require a falling edge of the slave select line (NSS) to initiate a character reception but only a low level. However, this low level must be present on the slave select line (NSS) at least 1 Tbit before the first serial clock cycle corresponding to the MSB bit.

25.6.8.7 Receiver Time-out

Because the receiver baudrate clock is active only during data transfers in SPI Mode, a receiver time-out is impossible in this mode, whatever the Time-out value is (field TO) in the Time-out Register (RTOR).

25.6.9 LIN Mode

The LIN Mode provides Master node and Slave node connectivity on a LIN bus.

The LIN (Local Interconnect Network) is a serial communication protocol which efficiently supports the control of mechatronic nodes in distributed automotive applications.

The main properties of the LIN bus are:

- Single Master/Multiple Slaves concept
- Low cost silicon implementation based on common UART/SCI interface hardware, an equivalent in software, or as a pure state machine.
- Self synchronization without quartz or ceramic resonator in the slave nodes
- Deterministic signal transmission
- Low cost single-wire implementation
- Speed up to 20 Kbit/s

LIN provides cost efficient bus communication where the bandwidth and versatility of CAN are not required.

The LIN Mode enables processing LIN frames with a minimum of action from the microprocessor.

25.6.9.1 Modes of operation

The USART can act either as a LIN Master node or as a LIN Slave node.

The node configuration is chosen by setting the MODE field in the Mode Register (MR):

- LIN Master Node (MODE=0xA)
- LIN Slave Node (MODE=0xB)

In order to avoid unpredicted behavior, any change of the LIN node configuration must be followed by a software reset of the transmitter and of the receiver (except the initial node configuration after a hardware reset). (See Section 25.6.9.3)

25.6.9.2 Baud Rate Configuration

See Section “25.6.1.1” on page 568.

- LIN Master Node: the baud rate is configured in the Baud Rate Generator Register (BRGR).
- LIN Slave Node: the initial baud rate is configured in BRGR, this configuration is automatically copied in the LIN Baud Rate Register (LINBRR) when writing BRGR. After synchronization procedure, the baud rate is updated in LINBRR.

25.6.9.3 Receiver and Transmitter Control

See Section “25.6.2” on page 572.

25.6.9.4 Character Transmission

See Section “25.6.3.1” on page 573.

25.6.9.5 Character Reception

See Section “25.6.3.7” on page 581.

25.6.9.6 *Header Transmission (Master Node Configuration)*

All the LIN Frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

So in Master node configuration, the frame handling starts with the sending of the header.

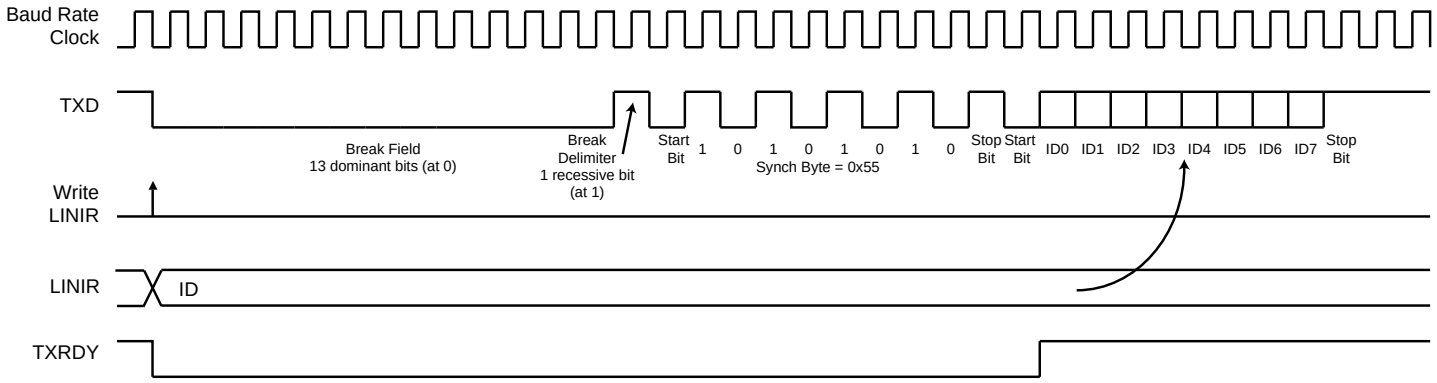
The header is transmitted as soon as the identifier is written in the LIN Identifier register (LINIR). At this moment the flag TXRDY falls.

The Break Field, the Synch Field and the Identifier Field are sent automatically one after the other.

The Break Field consists of 13 dominant bits and 1 recessive bit, the Synch Field is the character 0x55 and the Identifier corresponds to the character written in the LIN Identifier Register (LINIR). The Identifier parity bits can be automatically computed and sent (see Section 25.6.9.9).

The bit TXRDY is set when the identifier character is transferred into the Shift Register of the transmitter.

Figure 25-39. Header Transmission



25.6.9.7 Header Reception (Slave Node Configuration)

All the LIN Frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

In Slave node configuration, the frame handling starts with the reception of the header.

The USART uses a break detection threshold of 11 nominal bit times at the actual baud rate. At any time, if 11 consecutive recessive bits are detected on the bus, the USART detects a Break Field. As long as a Break Field has not been detected, the USART stays idle and the received data are not taken in account.

When a Break Field has been detected, the USART expects the Synch Field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized (see Section 25.6.9.8). If the received Synch character is not 0x55, an Inconsistent Synch Field error is generated (see Section 25.6.10).

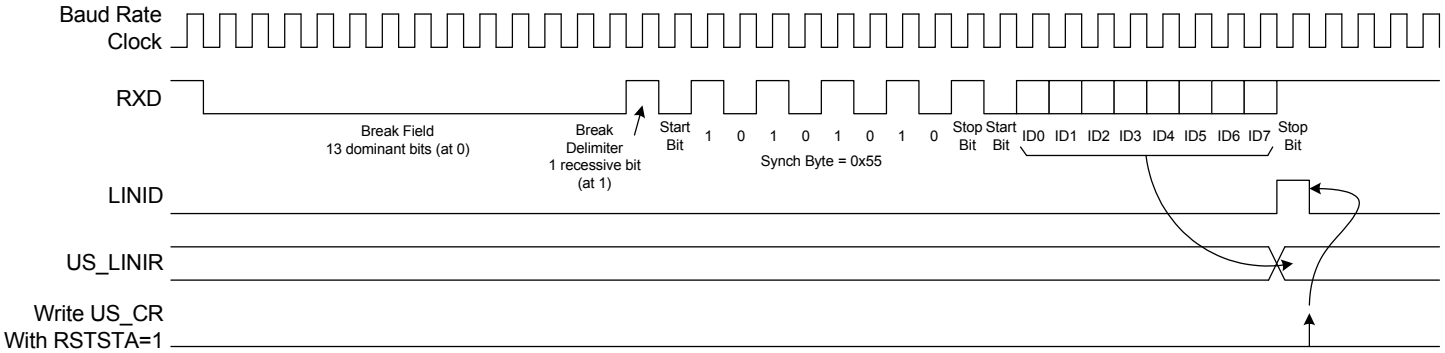
After receiving the Synch Field, the USART expects to receive the Identifier Field.

When the Identifier has been received, the flag LINID is set to “1”. At this moment the field IDCHR in the LIN Identifier register (LINIR) is updated with the received character. The Identifier parity bits can be automatically computed and checked (see Section 25.6.9.9).

If the header is not entirely received within the time given by the maximum length of the header THeader_Maximum, the error bit LINHTE in the Channel Status register (CSR) is set to 1.

The bits LINID, LINBK and LINHTE are reset by writing the bit RSTSTA to 1 in the Control register (CR).

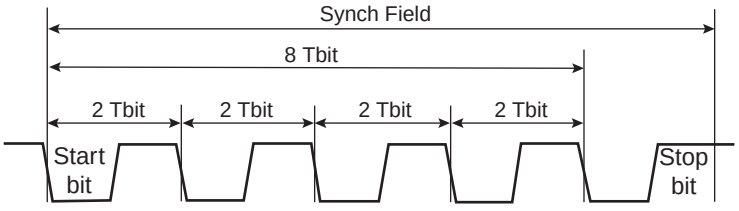
Figure 25-40. Header Reception



25.6.9.8 Slave Node Synchronization

The synchronization is done only in Slave node configuration. The procedure is based on time measurement between falling edges of the Synch Field. The falling edges are available in distances of 2, 4, 6 and 8 bit times.

Figure 25-41. Synch Field



The time measurement is made by a 19-bit counter clocked by the sampling clock (see Section 25.6.1).

When the start bit of the Synch Field is detected the counter is reset. Then during the next 8 Tbits of the Synch Field, the counter is incremented. At the end of these 8 Tbits, the counter is stopped. At this moment, the 16 most significant bits of the counter (value divided by 8) gives the new clock divider (LINCD) and the 3 least significant bits of this value (the remainder) gives the new fractional part (LINFP).

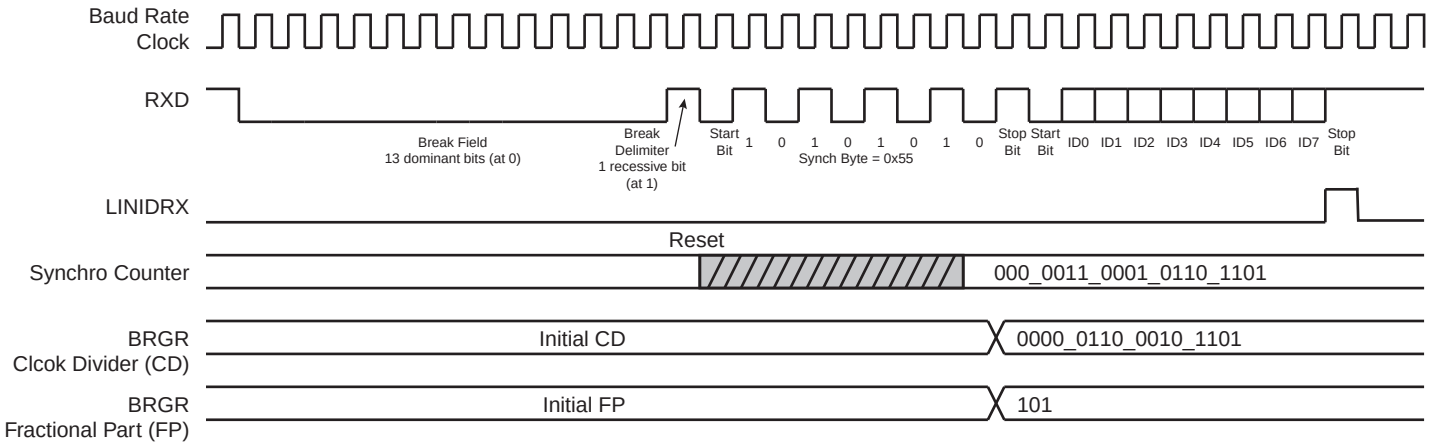
Once the Synch Field has been entirely received, the clock divider (LINCD) and the fractional part (LINFP) are updated in the LIN Baud Rate register (LINBRR) with the computed values, if the synchronization is not disabled by the bit SYNCDIS in the LIN Mode register (LINMR).

If after reception of the Synch Field, it appears that the computed baudrate deviation compared to the initial baud rate is superior to the maximum tolerance FToI_Unsynch (+/- 15%) then the clock divider (LINCD) and the fractional part (LINFP) are not updated and the error bit STE in the Channel Status register CSR is set to 1.

If after reception of the Synch Field, it appears that the sampled Synch character is not equal to 0x55 then the clock divider (LINCD) and the fractional part (LINFP) are not updated, and the error bit ISFE in the Channel Status register (CSR) is set to 1.

The bits LINSTE and LINISFE are reset by writing the bit RSTSTA at 1 in the Control register (CR).

Figure 25-42. Slave Node Synchronization



The accuracy of the synchronization depends on several parameters:

- The nominal clock frequency (F_{Nom}) (the theoretical slave node clock frequency)
- The Baudrate
- The oversampling (Over=0 => 16X or Over=0 => 8X)

The following formula is used to compute the deviation of the slave bit rate relative to the master bit rate after synchronization (F_{SLAVE} is the real slave node clock frequency).

$$\text{Baudrate\_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{Over}) + \beta] \times \text{Baudrate}}{8 \times F_{SLAVE}}\right)\%$$

$$\text{Baudrate\_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{Over}) + \beta] \times \text{Baudrate}}{8 \times \left(\frac{F_{TOL\_UNSYNCH}}{100} \right) \text{x} F_{Nom}} \right) \%$$

$$-0.5 \leq \alpha \leq +0.5 \quad -1 < \beta < +1$$

$F_{TOL\_UNSYNCH}$ is the deviation of the real slave node clock from the nominal clock frequency. The LIN Standard imposes that it must not exceed ±15%. The LIN Standard imposes also that for communication between two nodes, their bit rate must not differ by more than ±2%. This means that the Baudrate_deviation must not exceed ±1%.

It follows from that, a minimum value for the nominal clock frequency:

$$F_{NOM}(\text{min}) = \left(100 \times \frac{[0.5 \times 8 \times (2 - \text{Over}) + 1] \times \text{Baudrate}}{8 \times \left(\frac{-15}{100} + 1 \right) \times 1\%} \right) \text{Hz}$$

Examples:

- Baudrate = 20 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 2.64 MHz
- Baudrate = 20 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 1.47 MHz
- Baudrate = 1 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 132 kHz
- Baudrate = 1 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 74 kHz

If the fractional baud rate is not used, the accuracy of the synchronization becomes much lower. When the counter is stopped, the 16 most significant bits of the counter (value divided by 8) gives the new clock divider (CD). This value is rounded by adding the first insignificant bit. The equation of the Baudrate deviation is the same as given above, but the constants are as follows:

$$-4 \leq \alpha \leq +4 \quad -1 < \beta < +1$$

It follows from that, a minimum value for the nominal clock frequency:

$$F_{NOM}(\text{min}) = \left(100 \times \frac{[4 \times 8 \times (2 - \text{Over}) + 1] \times \text{Baudrate}}{8 \times \left(\frac{-15}{100} + 1 \right) \times 1\%} \right) \text{Hz}$$

Examples:

- Baudrate = 20 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 19.12 MHz
- Baudrate = 20 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 9.71 MHz
- Baudrate = 1 kbit/s, Over=0 (Oversampling 16X) => F_{Nom}(min) = 956 kHz
- Baudrate = 1 kbit/s, Over=1 (Oversampling 8X) => F_{Nom}(min) = 485 kHz

25.6.9.9 Identifier Parity

A protected identifier consists of two sub-fields; the identifier and the identifier parity. Bits 0 to 5 are assigned to the identifier and bits 6 and 7 are assigned to the parity.

The USART interface can generate/check these parity bits, but this feature can also be disabled. The user can choose between two modes by the PARDIS bit of the LIN Mode register (LINMR):

- PARDIS = 0:

During header transmission, the parity bits are computed and sent with the 6 least significant bits of the IDCHR field of the LIN Identifier register (LINIR). The bits 6 and 7 of this register are discarded.

During header reception, the parity bits of the identifier are checked. If the parity bits are wrong, an Identifier Parity error occurs (see Section 25.6.3.8). Only the 6 least significant bits of the IDCHR field are updated with the received Identifier. The bits 6 and 7 are stuck at 0.

- PARDIS = 1:

During header transmission, all the bits of the IDCHR field of the LIN Identifier register (LINIR) are sent on the bus.

During header reception, all the bits of the IDCHR field are updated with the received Identifier.

25.6.9.10 Node Action

In function of the identifier, the node is concerned, or not, by the LIN response. Consequently, after sending or receiving the identifier, the USART must be configured. There are three possible configurations:

- PUBLISH: the node sends the response.
- SUBSCRIBE: the node receives the response.
- IGNORE: the node is not concerned by the response, it does not send and does not receive the response.

This configuration is made by the field, Node Action (NACT), in the LINMR register (see Section 25.7.16).

Example: a LIN cluster that contains a Master and two Slaves:

- Data transfer from the Master to the Slave 1 and to the Slave 2:

  NACT(Master)=PUBLISH

  NACT(Slave1)=SUBSCRIBE

  NACT(Slave2)=SUBSCRIBE

- Data transfer from the Master to the Slave 1 only:

  NACT(Master)=PUBLISH

  NACT(Slave1)=SUBSCRIBE

  NACT(Slave2)=IGNORE

- Data transfer from the Slave 1 to the Master:

  NACT(Master)=SUBSCRIBE

  NACT(Slave1)=PUBLISH

  NACT(Slave2)=IGNORE

- Data transfer from the Slave1 to the Slave2:

  NACT(Master)=IGNORE

  NACT(Slave1)=PUBLISH

  NACT(Slave2)=SUBSCRIBE

- Data transfer from the Slave2 to the Master and to the Slave1:

  NACT(Master)=SUBSCRIBE

  NACT(Slave1)=SUBSCRIBE

  NACT(Slave2)=PUBLISH

25.6.9.11 Response Data Length

The LIN response data length is the number of data fields (bytes) of the response excluding the checksum.

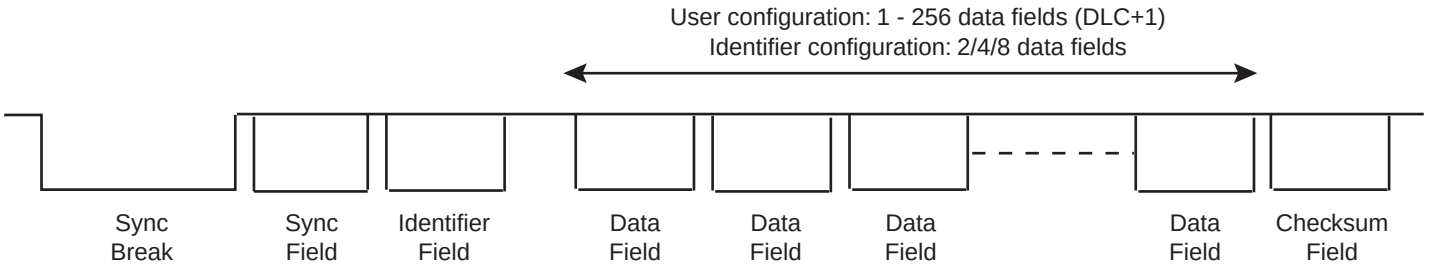
The response data length can either be configured by the user or be defined automatically by bits 4 and 5 of the Identifier (compatibility to LIN Specification 1.1). The user can choose between these two modes by the DLM bit of the LIN Mode register (LINMR):

- DLM = 0: the response data length is configured by the user via the DLC field of the LIN Mode register (LINMR). The response data length is equal to (DLC + 1) bytes. DLC can be programmed from 0 to 255, so the response can contain from 1 data byte up to 256 data bytes.
- DLM = 1: the response data length is defined by the Identifier (IDCHR in LINIR) according to the table below. The DLC field of the LIN Mode register (LINMR) is discarded. The response can contain 2 or 4 or 8 data bytes.

Table 25-14. Response Data Length if DLM = 1

IDCHR[5]	IDCHR[4]	Response Data Length [bytes]
0	0	2
0	1	2
1	0	4
1	1	8

Figure 25-43. Response Data Length



25.6.9.12 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8- bit sum with carry, over all data bytes or all data bytes and the protected identifier. Checksum calculation over the data bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and it is used for communication with LIN 2.0 slaves.

The USART can be configured to:

- Send/Check an Enhanced checksum automatically (CHKDIS = 0 & CHKTYP = 0)
- Send/Check a Classic checksum automatically (CHKDIS = 0 & CHKTYP = 1)
- Not send/check a checksum (CHKDIS = 1)

This configuration is made by the Checksum Type (CHKTYP) and Checksum Disable (CHKDIS) fields of the LIN Mode register (LINMR).

If the checksum feature is disabled, the user can send it manually all the same, by considering the checksum as a normal data byte and by adding 1 to the response data length (see Section 25.6.9.11).

25.6.9.13 *Frame Slot Mode*

This mode is useful only for Master nodes. It respects the following rule: each frame slot shall be longer than or equal to TFrame_Maximum.

If the Frame Slot Mode is enabled (FSDIS = 0) and a frame transfer has been completed, the TXRDY flag is set again only after TFrame_Maximum delay, from the start of frame. So the Master node cannot send a new header if the frame slot duration of the previous frame is inferior to TFrame_Maximum.

If the Frame Slot Mode is disabled (FSDIS = 1) and a frame transfer has been completed, the TXRDY flag is set again immediately.

The TFrame_Maximum is calculated as below:

If the Checksum is sent (CHKDIS = 0):

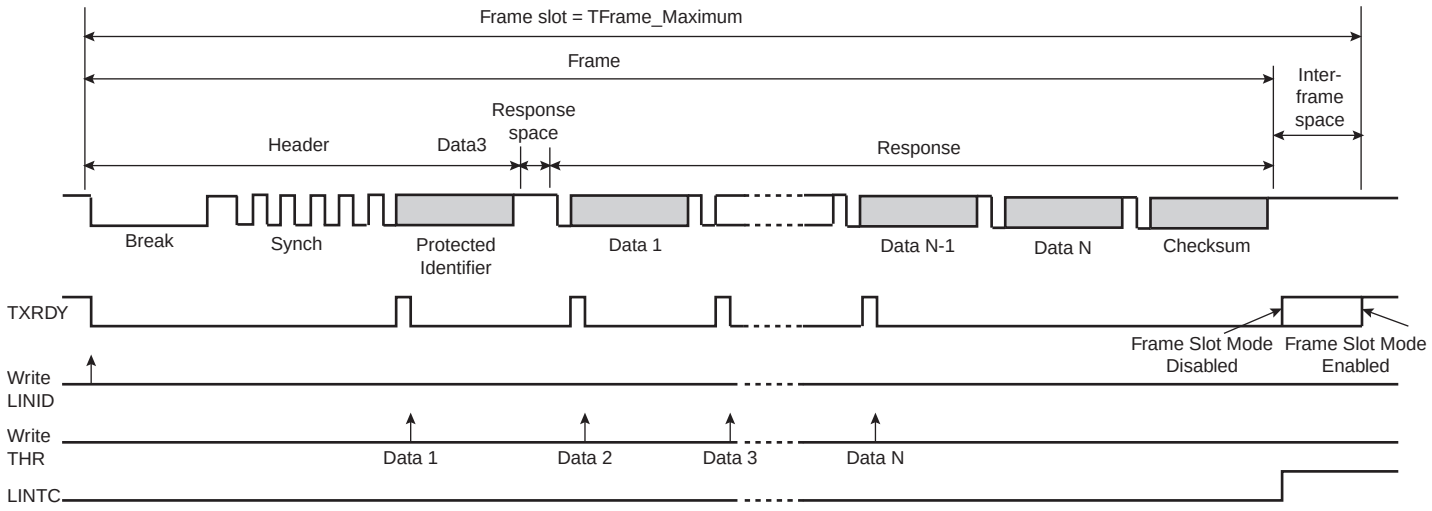
- THeader_Nominal = 34 x TBit
- TResponse_Nominal = 10 x (NData + 1) x TBit
- TFrame_Maximum = 1.4 x (THeader_Nominal + TResponse_Nominal + 1)(Note:)
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1 + 1) + 1) x TBIT
- TFrame_Maximum = (77 + 14 x DLC) x TBIT

If the Checksum is not sent (CHKDIS = 1):

- THeader_Nominal = 34 x TBit
- TResponse_Nominal = 10 x NData x TBit
- TFrame_Maximum = 1.4 x (THeader_Nominal + TResponse_Nominal + 1(Note:))
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1) + 1) x TBIT
- TFrame_Maximum = (63 + 14 x DLC) x TBIT

Note: The term “+1” leads to an integer result for TFrame_Max (LIN Specification 1.3)

Figure 25-44. Frame Slot Mode



25.6.10 LIN Errors

25.6.10.1 *Bit Error*

This error is generated when USART is transmitting and if the transmitted value on the Tx line is different from the value sampled on the Rx line. If a bit error is detected, the transmission is aborted at the next byte border.

This error is reported by LINBE in the Channel Status Register (CSR).

25.6.10.2 *Inconsistent Synch Field Error*

This error is generated if the Synch Field character received is other than 0x55.

This error is reported by CSR.LINISFE

25.6.10.3 *Identifier Parity Error*

This error is generated if the parity of the identifier is wrong. This error can be generated only if the parity feature is enabled (PARDIS = 0).

This error is reported by CSR.LINIPE

25.6.10.4 *Checksum Error*

This error is generated if the received checksum is wrong. Error bit is set to 1 only if the checksum feature is enabled (CHKDIS = 0).

This error is reported by CSR.LINCE

25.6.10.5 *Slave Not Responding Error*

This error is generated when the USART expects a response from another node (NACT = SUBSCRIBE) but no valid message appears on the bus within the time frame given by the maximum length of the message frame, TFrame_Maximum (see Section 25.6.9.13). This error is disabled if the USART does not expect any message (NACT = PUBLISH or NACT = IGNORE).

This error is reported by CSR.LINSNRE

25.6.10.6 *Synch Tolerance Error*

This error is generated if after the clock synchronization procedure it appears that the computed baudrate deviation compared to the initial baudrate is superior to the maximum tolerance FToI_Unsynch (+/- 15%).

This error is reported by CSR.LINSTE

25.6.10.7 *Header Time-out Error*

This error is generated if the Header is not entirely received within the time given by the maximum length of the Header, THeader_Maximum.

This error is reported by CSR.LINHTE

25.6.11 LIN Frame Handling

25.6.11.1 Master Node Configuration

- Write TXEN and RXEN in CR to enable both the transmitter and the receiver.
- Write MODE in MR to select the LIN mode and the Master Node configuration.
- Write CD and FP in BRGR to configure the baud rate.
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM, FSDIS and DLC in LINMR to configure the frame transfer.
- Check that TXRDY in CSR is set to “1”
- Write IDCHR in LINIR to send the header

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the USART sends the response
  - Wait until TXRDY in CSR rises
  - Write TCHR in THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in CSR rises
  - Read RCHR in RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in CSR rises
  - Check the LIN errors
- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in CSR rises
  - Check the LIN errors

Figure 25-45. Master Node Configuration, NACT = PUBLISH

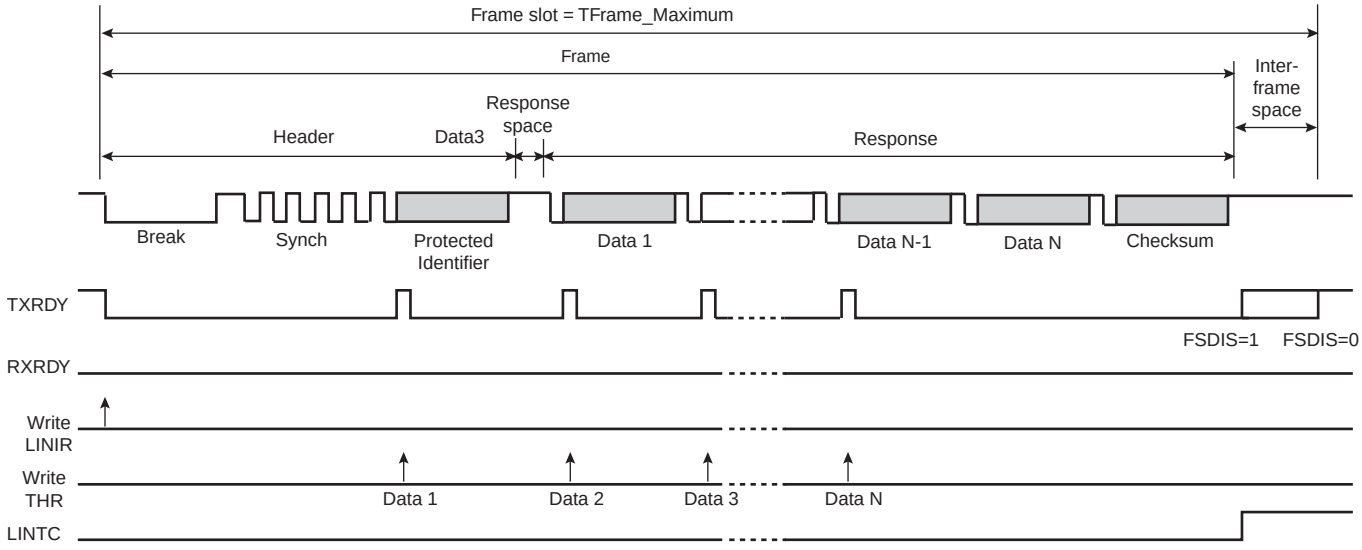


Figure 25-46. Master Node Configuration, NACT=SUBSCRIBE

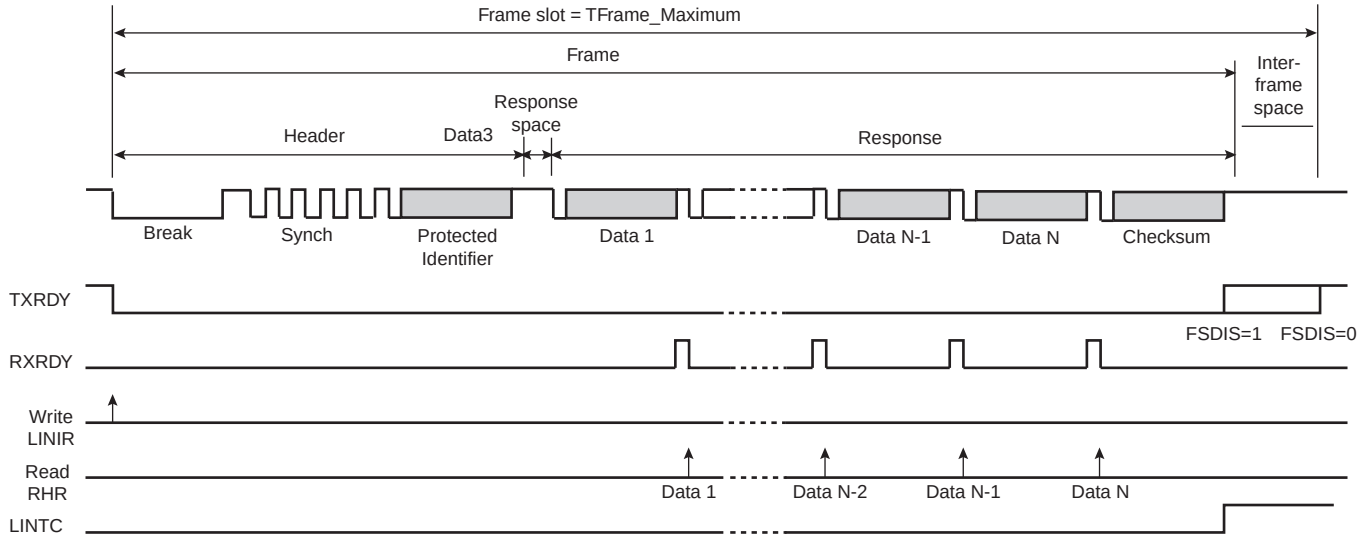
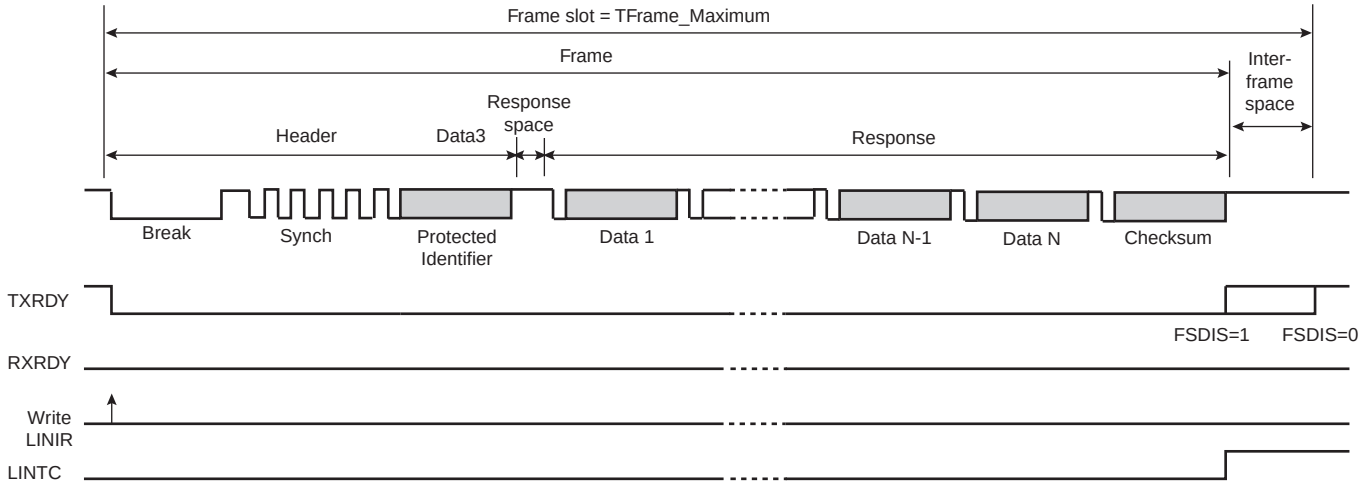


Figure 25-47. Master Node Configuration, NACT=IGNORE



25.6.11.2 Slave Node Configuration

- Write TXEN and RXEN in CR to enable both the transmitter and the receiver.
- Write MODE in MR to select the LIN mode and the Slave Node configuration.
- Write CD and FP in BRGR to configure the baud rate.
- Wait until LINID in CSR rises
- Check LINISFE and LINPE errors
- Read IDCHR in RHR
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM and DLC in LINMR to configure the frame transfer.

IMPORTANT: if the NACT configuration for this frame is PUBLISH, the US_LINMR register, must be write with NACT=PUBLISH even if this field is already correctly configured, that in order to set the TXREADY flag and the corresponding Peripheral DMA Controller write transfer request.

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the USART sends the response
  - Wait until TXRDY in CSR rises
  - Write TCHR in THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in CSR rises
  - Read RCHR in RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in CSR rises
  - Check the LIN errors

- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in CSR rises
  - Check the LIN errors

Figure 25-48. Slave Node Configuration, NACT = PUBLISH

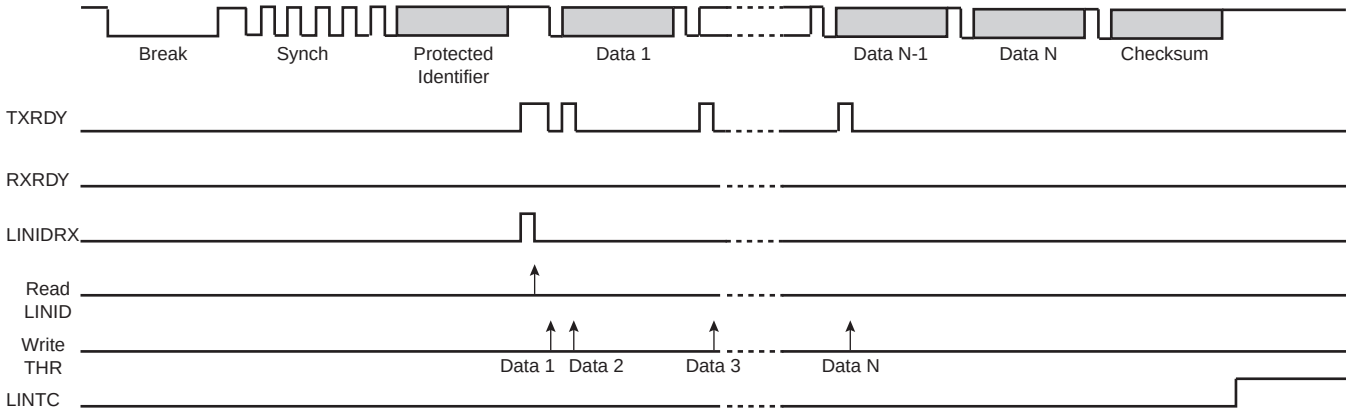


Figure 25-49. Slave Node Configuration, NACT = SUBSCRIBE

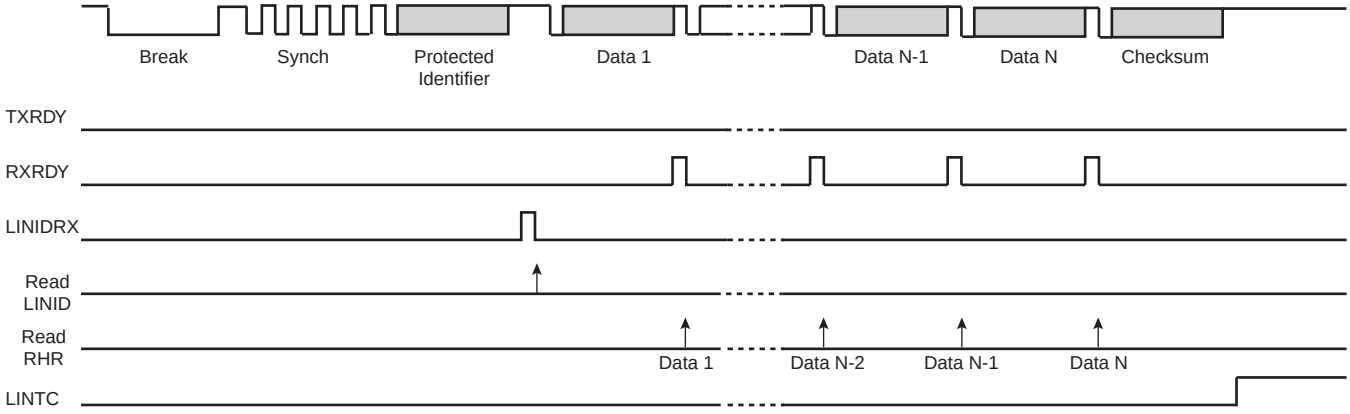
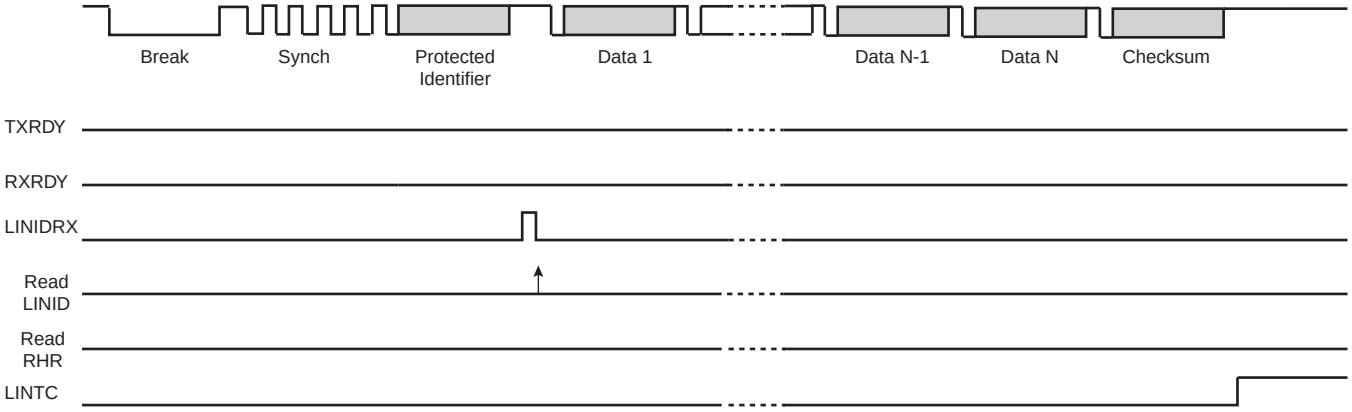


Figure 25-50. Slave Node Configuration, NACT = IGNORE



25.6.12 LIN Frame Handling With The Peripheral DMA Controller

The USART can be used in association with the Peripheral DMA Controller in order to transfer data directly into/from the on- and off-chip memories without any processor intervention.

The Peripheral DMA Controller uses the trigger flags, TXRDY and RXRDY, to write or read into the USART. The Peripheral DMA Controller always writes in the Transmit Holding register (THR) and it always reads in the Receive Holding register (RHR). The size of the data written or read by the Peripheral DMA Controller in the USART is always a byte.

25.6.12.1 Master Node Configuration

The user can choose between two Peripheral DMA Controller modes by the PDCM bit in the LIN Mode register (LINMR):

- PDCM = 1: the LIN configuration is stored in the WRITE buffer and it is written by the Peripheral DMA Controller in the Transmit Holding register THR (instead of the LIN Mode register LINMR). Because the Peripheral DMA Controller transfer size is limited to a byte, the transfer is split into two accesses. During the first access the bits, NACT, PARDIS, CHKDIS, CHKTYP, DLM and FSDIS are written. During the second access the 8-bit DLC field is written.
- PDCM = 0: the LIN configuration is not stored in the WRITE buffer and it must be written by the user in the LIN Mode register (LINMR).

The WRITE buffer also contains the Identifier and the DATA, if the USART sends the response (NACT = PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT = SUBSCRIBE).

Figure 25-51. Master Node with Peripheral DMA Controller (PDCM=1)

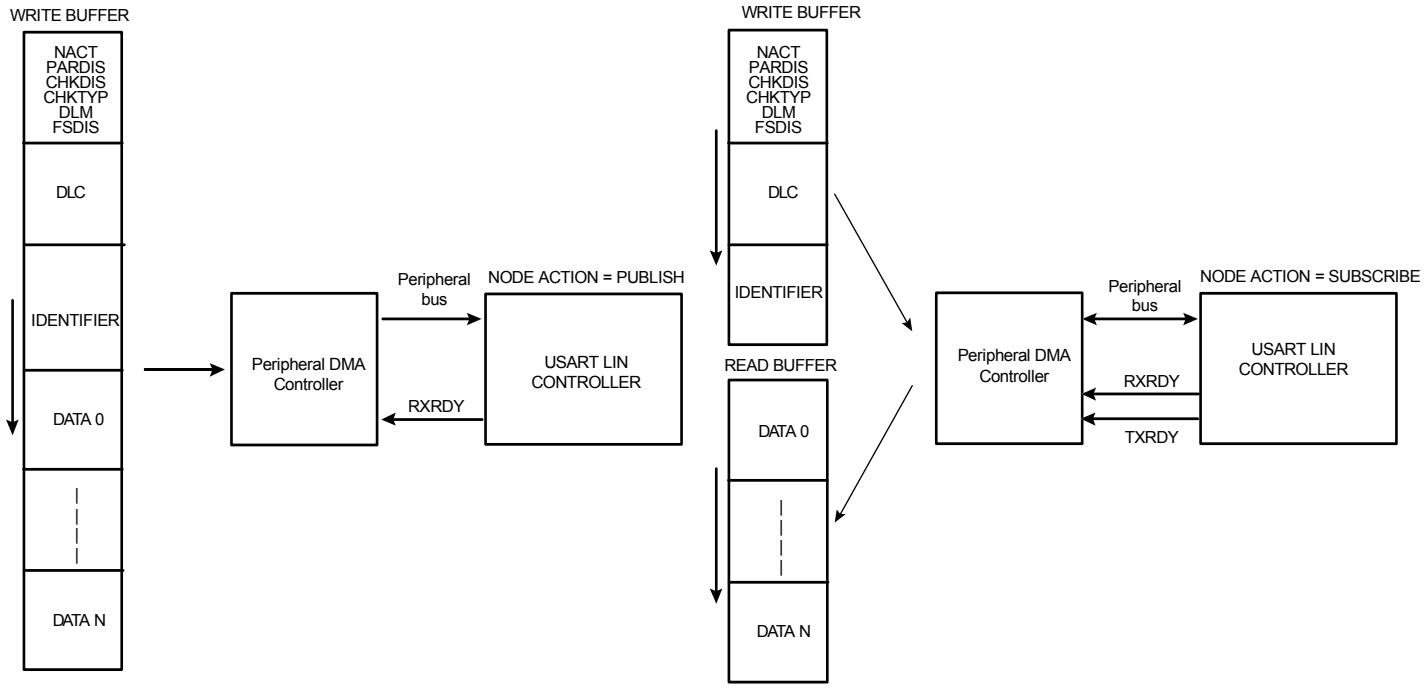
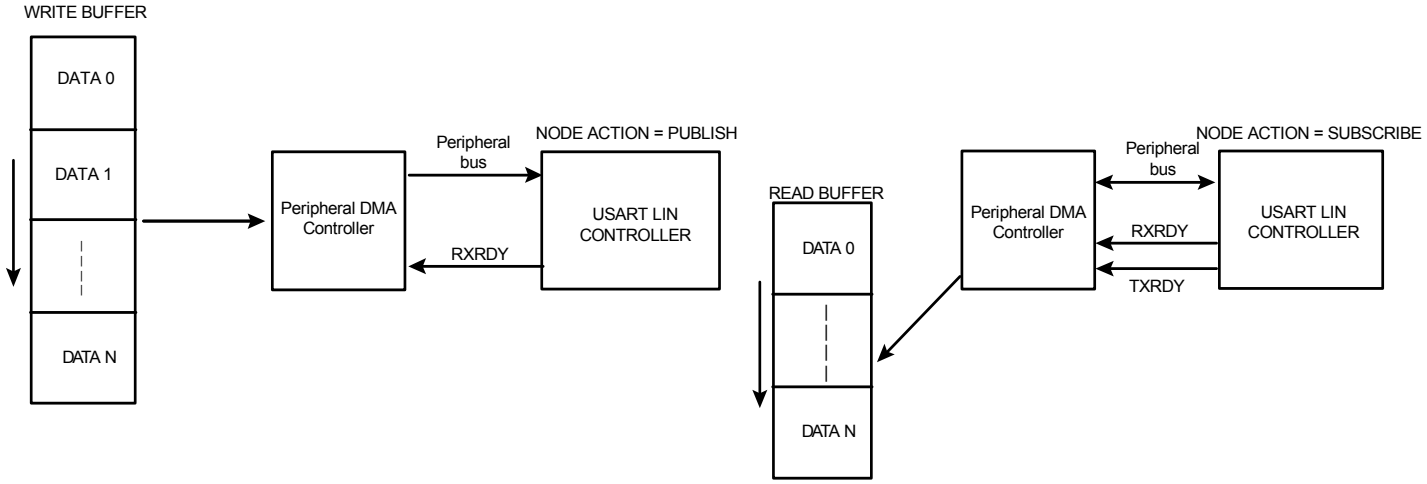


Figure 25-52. Master Node with Peripheral DMA Controller (PDCM=0)



25.6.12.2 Slave Node Configuration

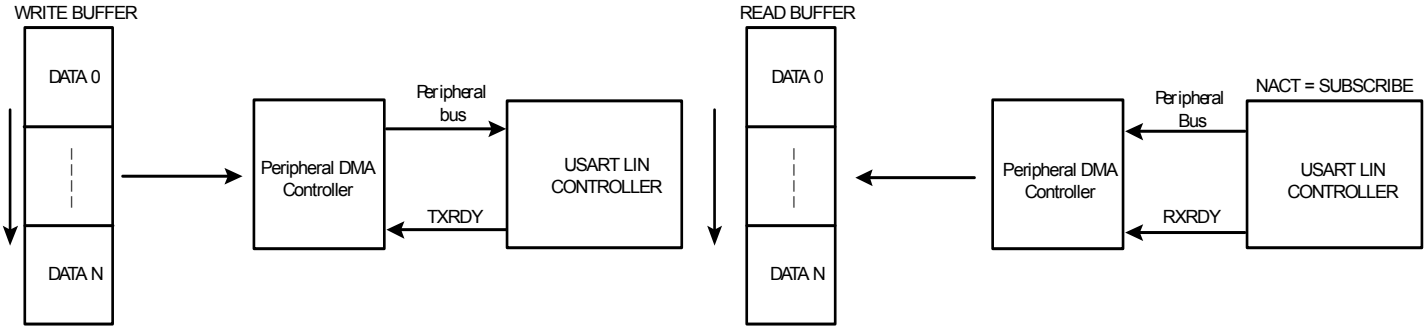
In this configuration, the Peripheral DMA Controller transfers only the DATA. The Identifier must be read by the user in the LIN Identifier register (LINIR). The LIN mode must be written by the user in the LIN Mode register (LINMR).

The WRITE buffer contains the DATA if the USART sends the response (NACT=PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT=SUBSCRIBE).

IMPORTANT: if the NACT configuration for a frame is PUBLISH, the US_LINMR register, must be write with NACT=PUBLISH even if this field is already correctly configured, that in order to set the TXREADY flag and the corresponding Peripheral DMA Controller write transfer request.

Figure 25-53. Slave Node with Peripheral DMA Controller



25.6.13 Wake-up Request

Any node in a sleeping LIN cluster may request a wake-up.

In the LIN 2.0 specification, the wakeup request is issued by forcing the bus to the dominant state from 250 µs to 5 ms. For this, it is necessary to send the character 0xF0 in order to impose 5 successive dominant bits. Whatever the baud rate is, this character respects the specified timings.

- Baud rate min = 1 kbit/s -> Tbit = 1ms -> 5 Tbits = 5 ms
- Baud rate max = 20 kbit/s -> Tbi t= 50 µs -> 5 Tbits = 250 µs

In the LIN 1.3 specification, the wakeup request should be generated with the character 0x80 in order to impose 8 successive dominant bits.

The user can choose by the WKUPTYP bit in the LIN Mode register (LINMR) either to send a LIN 2.0 wakeup request (WKUPTYP=0) or to send a LIN 1.3 wakeup request (WKUPTYP=1).

A wake-up request is transmitted by writing the Control Register (CR) with the LINWKUP bit at 1. Once the transfer is completed, the LINTC flag is asserted in the Status Register (SR). It is cleared by writing the Control Register (CR) with the RSTSTA bit at 1.

25.6.14 Bus Idle Time-out

If the LIN bus is inactive for a certain duration, the slave nodes shall automatically enter in sleep mode. In the LIN 2.0 specification, this time-out is fixed at 4 seconds. In the LIN 1.3 specification, it is fixed at 25000 Tbits.

In Slave Node configuration, the Receiver Time-out detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (CSR) rises and can generate an interrupt, thus indicating to the driver to go into sleep mode.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (RTOR). If the TO field is programmed at 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in CSR remains at 0. Otherwise, the receiver loads a 17-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises.

If STTTO is performed, the counter clock is stopped until a first character is received.

If RETTO is performed, the counter starts counting down immediately from the value TO.

Table 25-15. Receiver Time-out programming

LIN Specification	Baud Rate	Time-out period	TO
2.0	1 000 bit/s	4s	4 000
	2 400 bit/s		9 600
	9 600 bit/s		38 400
	19 200 bit/s		76 800
	20 000 bit/s		80 000
1.3	-	25 000 Tbits	25 000

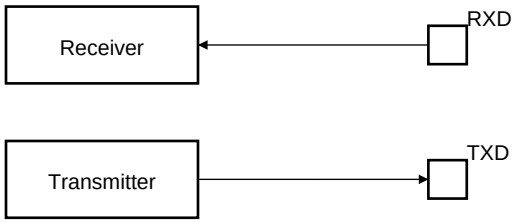
25.6.15 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

25.6.15.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

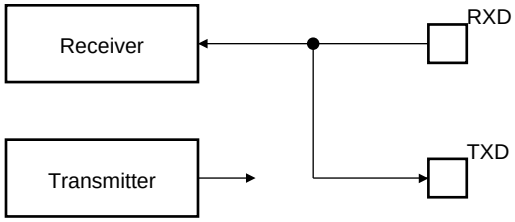
Figure 25-54. Normal Mode Configuration



25.6.15.2 Automatic Echo Mode

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 25-55. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

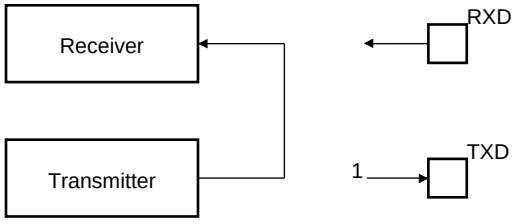
Figure 25-55. Automatic Echo Mode Configuration



25.6.15.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 25-56. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

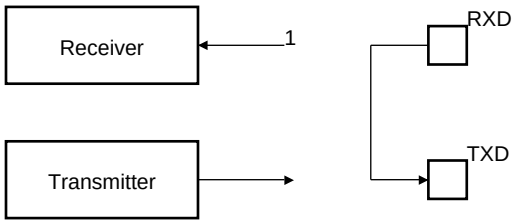
Figure 25-56. Local Loopback Mode Configuration



25.6.15.4 *Remote Loopback Mode*

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 25-57. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 25-57. Remote Loopback Mode Configuration



25.6.16 Write Protection Registers

To prevent any single software error that may corrupt USART behavior, certain address spaces can be write-protected by setting the WPEN bit in the USART Write Protect Mode Register (WPMR).

If a write access to the protected registers is detected, then the WPVS flag in the USART Write Protect Status Register (WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is reset by writing the USART Write Protect Mode Register (WPMR) with the appropriate access key, WPKEY.

The protected registers are:

- ”Mode Register” on page 628
- ”Baud Rate Generator Register” on page 640
- ”Receiver Time-out Register” on page 641
- ”Transmitter Timeguard Register” on page 642
- ”FI DI RATIO Register” on page 643
- ”IrDA FILTER Register” on page 645
- ”Manchester Configuration Register” on page 646

25.7 User Interface

Table 25-16. USART Register Memory Map

Offset	Register	Name	Access	Reset
0x0000	Control Register	CR	Write-only	–
0x0004	Mode Register	MR	Read-write	0x00000000
0x0008	Interrupt Enable Register	IER	Write-only	–
0x000C	Interrupt Disable Register	IDR	Write-only	–
0x0010	Interrupt Mask Register	IMR	Read-only	0x00000000
0x0014	Channel Status Register	CSR	Read-only	0x00000000
0x0018	Receiver Holding Register	RHR	Read-only	0x00000000
0x001C	Transmitter Holding Register	THR	Write-only	–
0x0020	Baud Rate Generator Register	BRGR	Read-write	0x00000000
0x0024	Receiver Time-out Register	RTOR	Read-write	0x00000000
0x0028	Transmitter Timeguard Register	TTGR	Read-write	0x00000000
0x0040	FI DI Ratio Register	FIDI	Read-write	0x00000174
0x0044	Number of Errors Register	NER	Read-only	0x00000000
0x004C	IrDA Filter Register	IFR	Read-write	0x00000000
0x0050	Manchester Encoder Decoder Register	MAN	Read-write	0x30011004
0x0054	LIN Mode Register	LINMR	Read-write	0x00000000
0x0058	LIN Identifier Register	LINIR	Read-write	0x00000000
0x005C	LIN Baud Rate Register	LINBRR	Read-only	0x00000000
0x00E4	Write Protect Mode Register	WPMR	Read-write	0x00000000
0x00E8	Write Protect Status Register	WPSR	Read-only	0x00000000
0x00FC	Version Register	VERSION	Read-only	0x–[1]

Note: 1. Values in the Version Register vary with the version of the IP block implementation.

25.7.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x0

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	LINWKUP	LINABT	RTSDIS/RCS	RTSEN/FCS	DTRDIS	DTREN

15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **LINWKUP: Send LIN Wakeup Signal**
  0: No effect:
  1: Sends a wakeup signal on the LIN bus.
- **LINABT: Abort LIN Transmission**
  0: No effect.
  1: Abort the current LIN transmission.
- **RTSDIS/RCS: Request to Send Disable/Release SPI Chip Select**
  If USART does not operate in SPI Master Mode (MODE … 0xE):
  0: No effect.
  1: Drives the pin RTS to 1.
  If USART operates in SPI Master Mode (MODE = 0xE):
  RCS = 0: No effect.
  RCS = 1: Releases the Slave Select Line NSS (RTS pin).
- **RTSEN/FCS: Request to Send Enable/Force SPI Chip Select**
  If USART does not operate in SPI Master Mode (MODE … 0xE):
  0: No effect.
  1: Drives the pin RTS to 0.
  If USART operates in SPI Master Mode (MODE = 0xE):
  FCS = 0: No effect.
  FCS = 1: Forces the Slave Select Line NSS (RTS pin) to 0, even if USART is no transmitting, in order to address SPI slave devices supporting the CSAAT Mode (Chip Select Active After Transfer).
- **DTRDIS: Data Terminal Ready Disable**
  0: No effect.
  1: Drives the pin DTR to 1.
- **DTREN: Data Terminal Ready Enable**
  0: No effect.
  1: Drives the pin DTR at 0.
- **RETTO: Rearm Time-out**
  0: No effect
  1: Restart Time-out

- **RSTNACK: Reset Non Acknowledge**
  - 0: No effect
  - 1: Resets NACK in CSR.
- **RSTIT: Reset Iterations**
  - 0: No effect.
  - 1: Resets ITERATION in CSR. No effect if the ISO7816 is not enabled.
- **SENDA: Send Address**
  - 0: No effect.
  - 1: In Multidrop Mode only, the next character written to the THR is sent with the address bit set.
- **STTTO: Start Time-out**
  - 0: No effect.
  - 1: Starts waiting for a character before clocking the time-out counter. Resets the status bit TIMEOUT in CSR.
- **STPBRK: Stop Break**
  - 0: No effect.
  - 1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.
- **STTBRK: Start Break**
  - 0: No effect.
  - 1: Starts transmission of a break after the characters present in THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.
- **RSTSTA: Reset Status Bits**
  - 0: No effect.
  - 1: Resets the status bits PARE, FRAME, OVRE, MANERR, LINBE, LINSFE, LINIPE, LINCE, LINSNRE and RXBRK in CSR.
- **TXDIS: Transmitter Disable**
  - 0: No effect.
  - 1: Disables the transmitter.
- **TXEN: Transmitter Enable**
  - 0: No effect.
  - 1: Enables the transmitter if TXDIS is 0.
- **RXDIS: Receiver Disable**
  - 0: No effect.
  - 1: Disables the receiver.
- **RXEN: Receiver Enable**
  - 0: No effect.
  - 1: Enables the receiver, if RXDIS is 0.
- **RSTTX: Reset Transmitter**
  - 0: No effect.
  - 1: Resets the transmitter.
- **RSTRX: Reset Receiver**
  - 0: No effect.
  - 1: Resets the receiver.

25.7.2 Mode Register

Name: MR

Access Type: Read-write

Offset: 0x4

Reset Value: -

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC	MAN	FILTER	–	MAX_ITERATION		
23	22	21	20	19	18	17	16
INVDATA	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF/CPOL
15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC/CPHA
7	6	5	4	3	2	1	0
CHRL		USCLKS		MODE			

This register can only be written if the WPEN bit is cleared in the Write Protect Mode Register (if exists).

- **ONEBIT: Start Frame Delimiter Selector**
  0: Start Frame delimiter is COMMAND or DATA SYNC.
  1: Start Frame delimiter is One Bit.
- **MODSYNC: Manchester Synchronization Mode**
  0:The Manchester Start bit is a 0 to 1 transition
  1: The Manchester Start bit is a 1 to 0 transition.
- **MAN: Manchester Encoder/Decoder Enable**
  0: Manchester Encoder/Decoder are disabled.
  1: Manchester Encoder/Decoder are enabled.
- **FILTER: Infrared Receive Line Filter**
  0: The USART does not filter the receive line.
  1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).
- **MAX_ITERATION**
  Defines the maximum number of iterations in mode ISO7816, protocol T= 0.
- **INVDATA: Inverted Data**
  0: The data field transmitted on TXD line is the same as the one written in THR register or the content read in RHR is the same as RXD line. Normal mode of operation.
  1: The data field transmitted on TXD line is inverted (voltage polarity) compared to the value written in THR register or the content read in RHR is inverted compared to RXD line (or ISO7816 IO line). Inverted Mode of operation, useful for contactless card application. To be used with configuration bit MSBF.
- **VAR_SYNC: Variable Synchronization of Command/Data Sync Start Frame Delimiter**
  0: User defined configuration of command or data sync field depending on SYNC value.
  1: The sync field is updated when a character is written into THR register.
- **DSNACK: Disable Successive NACK**
  0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).
  1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITERATION is asserted.

- **INACK: Inhibit Non Acknowledge**
  0: The NACK is generated.
  1: The NACK is not generated.
  Note: in SPI master mode, if INACK = 0 the character transmission starts as soon as character is written into THR register (assuming TXRDY was set). When INACK = 1, an additional condition must be met. The character transmission starts when a character is written and only if RXRDY bit is cleared (RHR has been read).
- **OVER: Oversampling Mode**
  0: 16x Oversampling.
  1: 8x Oversampling.
- **CLKO: Clock Output Select**
  0: The USART does not drive the CLK pin.
  1: The USART drives the CLK pin if USCLKS does not select the external clock CLK.
- **MODE9: 9-bit Character Length**
  0: CHRL defines character length.
  1: 9-bit character length.
- **MSBF/CPOL: Bit Order or SPI Clock Polarity**
  If USART does not operate in SPI Mode (MODE … 0xE and 0xF):
  MSBF = 0: Least Significant Bit is sent/received first.
  MSBF = 1: Most Significant Bit is sent/received first.
  If USART operates in SPI Mode (Slave or Master, MODE = 0xE or 0xF):
  CPOL = 0: The inactive state value of SPCK is logic level zero.
  CPOL = 1: The inactive state value of SPCK is logic level one.
  CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with CPHA to produce the required clock/data relationship between master and slave devices.
- **CHMODE: Channel Mode**

Table 25-17.

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo. Receiver input is connected to the TXD pin.
1	0	Local Loopback. Transmitter output is connected to the Receiver Input.
1	1	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **NBSTOP: Number of Stop Bits**

Table 25-18.

NBSTOP		Asynchronous (SYNC = 0)	Synchronous (SYNC = 1)
0	0	1 stop bit	1 stop bit
0	1	1.5 stop bits	Reserved
1	0	2 stop bits	2 stop bits
1	1	Reserved	Reserved

- **PAR: Parity Type**

Table 25-19.

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Parity forced to 0 (Space)
0	1	1	Parity forced to 1 (Mark)
1	0	x	No parity
1	1	x	Multidrop mode

- **SYNC/CPHA: Synchronous Mode Select or SPI Clock Phase**

  If USART does not operate in SPI Mode (MODE is … 0xE and 0xF):
  SYNC = 0: USART operates in Asynchronous Mode.
  SYNC = 1: USART operates in Synchronous Mode.
  If USART operates in SPI Mode (MODE = 0xE or 0xF):
  CPHA = 0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.
  CPHA = 1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.
  CPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CHRL: Character Length.**

Table 25-20.

CHRL		Character Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

- **USCLKS: Clock Selection**

Table 25-21.

USCLKS		Selected Clock
0	0	CLK_USART
0	1	CLK_USART/DIV(1)
1	0	Reserved
1	1	CLK

Note: 1. The value of DIV is device dependent. Please refer to the Module Configuration section at the end of this chapter.

- **MODE**

Table 25-22.

MODE				Mode of the USART
0	0	0	0	Normal
0	0	0	1	RS485
0	0	1	0	Hardware Handshaking
0	0	1	1	Modem
0	1	0	0	IS07816 Protocol: T = 0
0	1	1	0	IS07816 Protocol: T = 1
1	0	0	0	IrDA
1	0	1	0	LIN Master
1	0	1	1	LIN Slave
1	1	1	0	SPI Master
1	1	1	1	SPI Slave
Others				Reserved

25.7.3 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x8

Reset Value: -

31	30	29	28	27	26	25	24
LINHTE	LINSTE	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANEA
23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
LINTC	LINID	NACK/LINBK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

For backward compatibility the MANE bit has been duplicated to the MANEA bit position. Writing either one or the other has the same effect.

25.7.4 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0xC

Reset Value: -

31	30	29	28	27	26	25	24
LINHTE	LINSTE	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANEA
23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
LINTC	LINID	NACK/LINBK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

For backward compatibility the MANE bit has been duplicated to the MANEA bit position. Writing either one or the other has the same effect.

25.7.5 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x10

Reset Value: -

31	30	29	28	27	26	25	24
LINHTE	LINSTE	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANEA
23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
LINTC	LINID	NACK/LINBK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared when the corresponding bit in IDR is written to one.

A bit in this register is set when the corresponding bit in IER is written to one.

For backward compatibility the MANE bit has been duplicated to the MANEA bit position. Reading either one or the other has the same effect.

25.7.6 Channel Status Register

Name: CSR

Access Type: Read-only

Offset: 0x14

Reset Value: -

31	30	29	28	27	26	25	24
LINHTE	LINSTE	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANERR
23	22	21	20	19	18	17	16
CTS/LINBLS	DCD	DSR	RI	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
LINTC	LINID	NACK/LINBK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

- **LINHTE: LIN Header Time-out Error**
  - 0: No LIN Header Time-out error has been detected since the last RSTSTA.
  - 1: A LIN Header Time-out error has been detected since the last RSTSTA.
- **LINSTE: LIN Synch Tolerance Error**
  - 0: No LIN Synch Tolerance error has been detected since the last RSTSTA.
  - 1: A LIN Synch Tolerance error has been detected since the last RSTSTA.
- **LINSNRE: LIN Slave Not Responding Error**
  - 0: No LIN Slave Not Responding Error has been detected since the last RSTSTA.
  - 1: A LIN Slave Not Responding Error has been detected since the last RSTSTA.
- **LINCE: LIN Checksum Error**
  - 0: No LIN Checksum Error has been detected since the last RSTSTA.
  - 1: A LIN Checksum Error has been detected since the last RSTSTA.
- **LINIPE: LIN Identifier Parity Error**
  - 0: No LIN Identifier Parity Error has been detected since the last RSTSTA.
  - 1: A LIN Identifier Parity Error has been detected since the last RSTSTA.
- **LINISFE: LIN Inconsistent Synch Field Error**
  - 0: No LIN Inconsistent Synch Field Error has been detected since the last RSTSTA
  - 1: The USART is configured as a Slave node and a LIN Inconsistent Synch Field Error has been detected since the last RSTSTA.
- **LINBE: LIN Bit Error**
  - 0: No Bit Error has been detected since the last RSTSTA.
  - 1: A Bit Error has been detected since the last RSTSTA.
- **MANERR: Manchester Error**
  - 0: No Manchester error has been detected since the last RSTSTA.
  - 1: At least one Manchester error has been detected since the last RSTSTA.
- **CTS/LINBLS: Image of CTS Input or LIN Bus Line Status**
  - **– If USART does not operate in LIN Mode (Master or Slave):**
    - 0: CTS is at 0.
    - 1: CTS is at 1.
  - **– If USART operates in LIN Mode (Master or Slave):**

0: LIN Bus Line is at 0.
1: LIN Bus Line is at 1.

- **DCD: Image of DCD Input**
  0: DCD is at 0.
  1: DCD is at 1.
- **DSR: Image of DSR Input**
  0: DSR is at 0.
  1: DSR is at 1.
- **RI: Image of RI Input**
  0: RI is at 0.
  1: RI is at 1.
- **CTSIC: Clear to Send Input Change Flag**
  0: No input change has been detected on the CTS pin since the last read of CSR.
  1: At least one input change has been detected on the CTS pin since the last read of CSR.
- **DCDIC: Data Carrier Detect Input Change Flag**
  0: No input change has been detected on the DCD pin since the last read of CSR.
  1: At least one input change has been detected on the DCD pin since the last read of CSR.
- **DSRIC: Data Set Ready Input Change Flag**
  0: No input change has been detected on the DSR pin since the last read of CSR.
  1: At least one input change has been detected on the DSR pin since the last read of CSR.
- **RIIC: Ring Indicator Input Change Flag**
  0: No input change has been detected on the RI pin since the last read of CSR.
  1: At least one input change has been detected on the RI pin since the last read of CSR.
- **LINTC: LIN Transfer Completed**
  0: The USART is idle or a LIN transfer is ongoing.
  1: A LIN transfer has been completed since the last RSTSTA.
- **LINID: LIN Identifier**
  0: No LIN Identifier received or sent
  1: The USART is configured as a Slave node and a LIN Identifier has been received or the USART is configured as a Master node and a LIN Identifier has been sent since the last RSTSTA.
- **NACK: Non Acknowledge**
  0: No Non Acknowledge has not been detected since the last RSTNACK.
  1: At least one Non Acknowledge has been detected since the last RSTNACK.
- **RXBUFF: Reception Buffer Full**
  0: The signal Buffer Full from the Receive Peripheral DMA Controller channel is inactive.
  1: The signal Buffer Full from the Receive Peripheral DMA Controller channel is active.
- **ITER/UNRE: Max number of Repetitions Reached or SPI Underrun Error**
  If USART does not operate in SPI Slave Mode (MODE … 0xF):
  ITER = 0: Maximum number of repetitions has not been reached since the last RSTSTA.
  ITER = 1: Maximum number of repetitions has been reached since the last RSTSTA.
  If USART operates in SPI Slave Mode (MODE = 0xF):
  UNRE = 0: No SPI underrun error has occurred since the last RSTSTA.
  UNRE = 1: At least one SPI underrun error has occurred since the last RSTSTA.
- **TXEMPTY: Transmitter Empty**
  0: There are characters in either THR or the Transmit Shift Register, or the transmitter is disabled.
  1: There are no characters in THR, nor in the Transmit Shift Register.
- **TIMEOUT: Receiver Time-out**
  0: There has not been a time-out since the last Start Time-out command (STTTO in CR) or the Time-out Register is 0.
  1: There has been a time-out since the last Start Time-out command (STTTO in CR).
- **PARE: Parity Error**

0: No parity error has been detected since the last RSTSTA.
1: At least one parity error has been detected since the last RSTSTA.

- **FRAME: Framing Error**
  0: No stop bit has been detected low since the last RSTSTA.
  1: At least one stop bit has been detected low since the last RSTSTA.
- **OVRE: Overrun Error**
  0: No overrun error has occurred since the last RSTSTA.
  1: At least one overrun error has occurred since the last RSTSTA.
- **RXBRK: Break Received/End of Break**
  0: No Break received or End of Break detected since the last RSTSTA.
  1: Break Received or End of Break detected since the last RSTSTA.
- **TXRDY: Transmitter Ready**
  0: A character is in the THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.
  1: There is no character in the THR.
- **RXRDY: Receiver Ready**
  0: No complete character has been received since the last read of RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.
  1: At least one complete character has been received and RHR has not yet been read.

25.7.7 Receive Holding Register

Name: RHR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RXSYNH	–	–	–	–	–	–	RXCHR

7	6	5	4	3	2	1	0
RXCHR							

- **RXSYNH: Received Sync**
  0: Last Character received is a Data.
  1: Last Character received is a Command.
- **RXCHR: Received Character**
  Last character received if RXRDY is set.

25.7.8 USART Transmit Holding Register

Name: THR

Access Type: Write-only

Offset: 0x1C

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TXSYNH	–	–	–	–	–	–	TXCHR

7	6	5	4	3	2	1	0
TXCHR							

- **TXSYNH: Sync Field to be transmitted**

  0: The next character sent is encoded as a data. Start Frame Delimiter is DATA SYNC.

  1: The next character sent is encoded as a command. Start Frame Delimiter is COMMAND SYNC.

- **TXCHR: Character to be Transmitted**

  Next character to be transmitted after the current character if TXRDY is not set.

25.7.9 Baud Rate Generator Register

Name: BRGR

Access Type: Read-write

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	FP		

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

This register can only be written if the WPEN bit is cleared in the Write Protect Mode Register.

- **FP: Fractional Part**
  0: Fractional divider is disabled.
  1 - 7: Baudrate resolution, defined by FP x 1/8.
- **CD: Clock Divider**

Table 25-23.

	MODE ≠ ISO7816			
	SYNC = 0		**SYNC = 1 or MODE = SPI (Master or Slave)**	
CD	**OVER = 0**	**OVER = 1**		**MODE = ISO7816**
0	Baud Rate Clock Disabled			
1 to 65535	Baud Rate = Selected Clock/16/CD	Baud Rate = Selected Clock/8/CD	Baud Rate = Selected Clock /CD	Baud Rate = Selected Clock/CD/FI_DI_RATIO

25.7.10 Receiver Time-out Register

Name: RTOR

Access Type: Read-write

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	TO

15	14	13	12	11	10	9	8
TO							

7	6	5	4	3	2	1	0
TO							

This register can only be written if the WPEN bit is cleared in the Write Protect Mode Register.

- **TO: Time-out Value**

  0: The Receiver Time-out is disabled.

  1 - 131071: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

  Note that the size of the TO counter can change depending of implementation. See the Module Configuration section.

25.7.11 Transmitter Timeguard Register

Name: TTGR

Access Type: Read-write

Offset: 0x28

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TG							

This register can only be written if the WPEN bit is cleared in the Write Protect Mode Register.

- **TG: Timeguard Value**

  0: The Transmitter Timeguard is disabled.

  1 - 255: The Transmitter timeguard is enabled and the timeguard delay is TG x Bit Period.

25.7.12 FI DI RATIO Register

Name: FIDI

Access Type: Read-write

Offset: 0x40

Reset Value: 0x00000174

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	FI_DI_RATIO		

7	6	5	4	3	2	1	0
FI_DI_RATIO							

This register can only be written if the WPEN bit is cleared in the Write Protect Mode Register.

- **FI_DI_RATIO: FI Over DI Ratio Value**
  0: If ISO7816 mode is selected, the Baud Rate Generator generates no signal.
  1 - 2047: If ISO7816 mode is selected, the Baud Rate is the clock provided on CLK divided by FI_DI_RATIO.

25.7.13 Number of Errors Register

Name: NER

Access Type: Read-only

Offset: 0x44

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
NB_ERRORS							

- **NB_ERRORS: Number of Errors**

  Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

25.7.14 IrDA FILTER Register

Name: IFR

Access Type: Read-write

Offset: 0x4C

Reset Value: 0x00000000

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

This register can only be written if the WPEN bit is cleared in the Write Protect Mode Register (if exists).

IRDA_FILTER: IrDA Filter

Sets the filter of the IrDA demodulator.

25.7.15 Manchester Configuration Register

Name: MAN

Access Type: Read-write

Offset: 0x50

Reset Value: 0x30011004

31	30	29	28	27	26	25	24
–	DRIFT	1	RX_MPOL	–	–	RX_PP	
23	22	21	20	19	18	17	16
–	–	–	–	RX_PL			
15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	
7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

This register can only be written if the WPEN bit is cleared in the Write Protect Mode Register (if exists).

- **DRIFT: Drift compensation**
  0: The USART can not recover from an important clock drift
  1: The USART can recover from clock drift. The 16X clock mode must be enabled.
- **RX_MPOL: Receiver Manchester Polarity**
  0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.
  1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.
- **RX_PP: Receiver Preamble Pattern detected**

Table 25-24.

RX_PP		Preamble Pattern default polarity assumed (RX_MPOL field not set)
0	0	ALL_ONE
0	1	ALL_ZERO
1	0	ZERO_ONE
1	1	ONE_ZERO

- **RX_PL: Receiver Preamble Length**
  0: The receiver preamble pattern detection is disabled
  1 - 15: The detected preamble length is RX_PL x Bit Period
- **TX_MPOL: Transmitter Manchester Polarity**
  0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.
  1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.

- **TX_PP: Transmitter Preamble Pattern**

Table 25-25.

TX_PP		Preamble Pattern default polarity assumed (TX_MPOL field not set)
0	0	ALL_ONE
0	1	ALL_ZERO
1	0	ZERO_ONE
1	1	ONE_ZERO

- **TX_PL: Transmitter Preamble Length**

  0: The Transmitter Preamble pattern generation is disabled

  1 - 15: The Preamble Length is TX_PL x Bit Period

25.7.16 LIN Mode Register

Name: LINMR

Access Type: Read-write

Offset: 0x54

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	SYNCDIS	PDCM

15	14	13	12	11	10	9	8
DLC							

7	6	5	4	3	2	1	0
WKUPTYP	FSDIS	DLM	CHKTYP	CHKDIS	PARDIS	NACT	

- **SYNCDIS: Synchronization Disable**
  - 0: The Synchronization procedure is performed in LIN Slave node configuration.
  - 1: The Synchronization procedure is not performed.
- **PDCM: Peripheral DMA Controller Mode**
  - 0: The LIN mode register LINMR is not written by the Peripheral DMA Controller.
  - 1: The LIN mode register LINMR (excepting that bit) is written by the Peripheral DMA Controller.
- **DLC: Data Length Control**
  - 0 - 255: Defines the response data length if DLM=0,in that case the response data length is equal to DLC+1 bytes.
- **WKUPTYP: Wakeup Signal Type**
  - 0: setting the bit LINWKUP in the control register sends a LIN 2.0 wakeup signal.
  - 1: setting the bit LINWKUP in the control register sends a LIN 1.3 wakeup signal.
- **FSDIS: Frame Slot Mode Disable**
  - 0: The Frame Slot Mode is enabled.
  - 1: The Frame Slot Mode is disabled.
- **DLM: Data Length Mode**
  - 0: The response data length is defined by the field DLC of this register.
  - 1: The response data length is defined by the bits 4 and 5 of the Identifier (IDCHR in LINIR).
- **CHKTYP: Checksum Type**
  - 0: LIN 2.0 “Enhanced” Checksum
  - 1: LIN 1.3 “Classic” Checksum
- **CHKDIS: Checksum Disable**
  - 0: In Master node configuration, the checksum is computed and sent automatically. In Slave node configuration, the checksum is checked automatically.
  - 1: Whatever the node configuration is, the checksum is not computed/sent and it is not checked.
- **PARDIS: Parity Disable**
  - 0: In Master node configuration, the Identifier Parity is computed and sent automatically. In Master node and Slave node configuration, the parity is checked automatically.
  - 1:Whatever the node configuration is, the Identifier parity is not computed/sent and it is not checked.

- **NACT: LIN Node Action**

Table 1.

NACT		Mode Description
0	0	PUBLISH: The USART transmits the response.
0	1	SUBSCRIBE: The USART receives the response.
1	0	IGNORE: The USART does not transmit and does not receive the response.
1	1	Reserved

25.7.17 LIN Identifier Register

Name: LINIR

Access Type: Read-write or Read-only

Offset: 0x58

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
IDCHR							

- **IDCHR: Identifier Character**

  If MODE=0xA (Master node configuration):
  IDCHR is Read-write and its value is the Identifier character to be transmitted.
  if MODE=0xB (Slave node configuration):
  IDCHR is Read-only and its value is the last Identifier character that has been received.

25.7.18 LIN Baud Rate Register

Name: LINBRR

Access Type: Read-only

Offset: 0x5C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	LINFP		

15	14	13	12	11	10	9	8
LINCD							

7	6	5	4	3	2	1	0
LINCD							

- **LINFP: LIN Fractional Part after Synchronization**
- **LINCD: LIN Clock Divider after Synchronization**

25.7.19 Write Protect Mode Register

Register Name: WPMR

Access Type: Read-write

Offset: 0xE4

Reset Value: See Table 25-16

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPEN

- **WPKEY: Write Protect KEY**

  Should be written at value 0x555341 ("USA" in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

- **WPEN: Write Protect Enable**

  0 = Disables the Write Protect if WPKEY corresponds to 0x555341 ("USA" in ASCII).

  1 = Enables the Write Protect if WPKEY corresponds to 0x555341 ("USA" in ASCII).

Protects the registers:

- "Mode Register" on page 628
- "Baud Rate Generator Register" on page 640
- "Receiver Time-out Register" on page 641
- "Transmitter Timeguard Register" on page 642
- "FI DI RATIO Register" on page 643
- "IrDA FILTER Register" on page 645
- "Manchester Configuration Register" on page 646

25.7.20 Write Protect Status Register

Register Name: WPSR

Access Type: Read-only

Offset: 0xE8

Reset Value: See Table 25-16

31	30	29	28	27	26	25	24
—	—	—	—	—	—	—	—

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPVS

- **WPVSRC: Write Protect Violation Source**

  When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

- **WPVS: Write Protect Violation Status**

  0 = No Write Protect Violation has occurred since the last read of the WPSR register.

  1 = A Write Protect Violation has occurred since the last read of the WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

Note: Reading WPSR automatically clears all fields.

25.7.21 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	MFN			

15	14	13	12	11	10	9	8
–	–	–	–	VERSION			

7	6	5	4	3	2	1	0
VERSION							

- **MFN**

  Reserved. No functionality associated.

- **VERSION**

  Version of the module. No functionality associated.

25.8 Module Configuration

The specific configuration for each USART instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks according to the table in the System Bus Clock Connections section.

Table 25-26. Module Configuration

Feature	USART0 USART2 USART3 USART4	USART1
ISO7816	Not Implemented	Implemented
IRDA Logic	Not Implemented	Implemented
RS485 Logic	Not Implemented	Implemented
Modem Logic	Not Implemented	Implemented
SPI Logic	Implemented	Implemented
LIN Logic	Implemented	Implemented
Manchester Logic	Not Implemented	Implemented
Fractional Baudrate	Implemented	Implemented
DIV value for divided CLK_USART	8	8
Receiver Time-out Counter Size (Size of the RTOR.TO field)	17-bits	17-bits

Table 25-27. Module Clock Name

Module name	Clock name	Description
USART0	CLK_USART0	Peripheral Bus clock from the PBA clock domain
USART1	CLK_USART1	Peripheral Bus clock from the PBC clock domain
USART2	CLK_USART2	Peripheral Bus clock from the PBA clock domain
USART3	CLK_USART3	Peripheral Bus clock from the PBA clock domain
USART4	CLK_USART4	Peripheral Bus clock from the PBC clock domain

25.8.1 Clock Connections

Each USART can be connected to an internally divided clock:

Table 25-28. USART Clock Connections

USART	Source	Name	Connection
0	Internal	CLK_DIV	PBA Clock / 8 (CLK_PBA_USART_DIV)
1			PBC Clock / 8 (CLK_PBC_USART_DIV)
2			PBA Clock / 8 (CLK_PBA_USART_DIV)
3			PBA Clock / 8 (CLK_PBA_USART_DIV)
4			PBC Clock / 8 (CLK_PBC_USART_DIV)

25.8.2 Register Reset Values

Table 25-29. Register Reset Values

Register	Reset Value
VERSION	0x00000602

26. Serial Peripheral Interface (SPI)

Rev: 2.1.1.3

26.1 Features

- **Compatible with an embedded 32-bit microcontroller**
- **Supports communication with serial external devices**
  - **Four chip selects with external decoder support allow communication with up to 15 peripherals**
  - **Serial memories, such as DataFlash and 3-wire EEPROMs**
  - **Serial peripherals, such as ADCs, DACs, LCD controllers, CAN controllers and Sensors**
  - **External co-processors**
- **Master or Slave Serial Peripheral Bus Interface**
  - **4 - to 16-bit programmable data length per chip select**
  - **Programmable phase and polarity per chip select**
  - **Programmable transfer delays between consecutive transfers and between clock and data per chip select**
  - **Programmable delay between consecutive transfers**
  - **Selectable mode fault detection**
- **Connection to Peripheral DMA Controller channel capabilities optimizes data transfers**
  - **One channel for the receiver, one channel for the transmitter**
  - **Next buffer support**
  - **Four character FIFO in reception**

26.2 Overview

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (Multiple Master Protocol opposite to Single Master Protocol where one CPU is always the master while all of the others are always slaves) and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

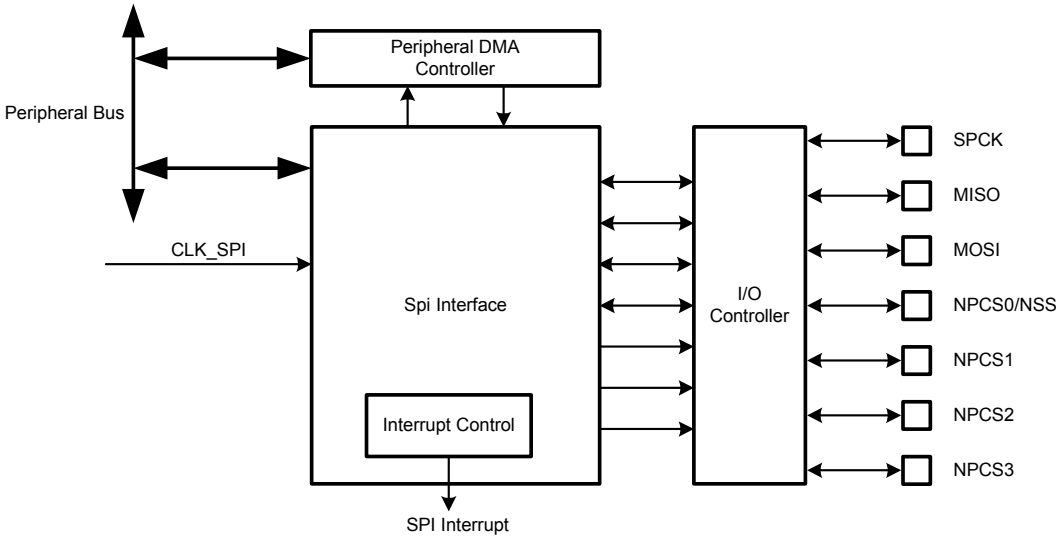
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): this data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): this data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): this control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): this control line allows slaves to be turned on and off by hardware.

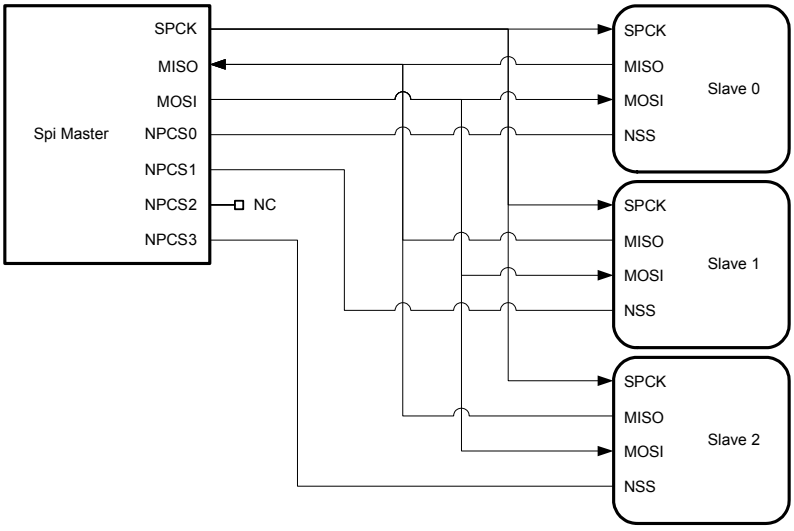
26.3 Block Diagram

Figure 26-1. SPI Block Diagram



26.4 Application Block Diagram

Figure 26-2. Application Block Diagram: Single Master/Multiple Slave Implementation



26.5 I/O Lines Description

Table 26-1. I/O Lines Description

Pin Name	Pin Description	Type	
		Master	Slave
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1-NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

26.6 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

26.6.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with I/O lines. The user must first configure the I/O Controller to assign the SPI pins to their peripheral functions.

26.6.2 Clocks

The clock for the SPI bus interface (CLK_SPI) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the SPI before disabling the clock, to avoid freezing the SPI in an undefined state.

26.6.3 Interrupts

The SPI interrupt request line is connected to the interrupt controller. Using the SPI interrupt requires the interrupt controller to be programmed first.

26.7 Functional Description

26.7.1 Modes of Operation

The SPI operates in master mode or in slave mode.

Operation in master mode is configured by writing a one to the Master/Slave Mode bit in the Mode Register (MR.MSTR). The pins NPCS0 to NPCS3 are all configured as outputs, the SPCK pin is driven, the MISO line is wired on the receiver input and the MOSI line driven as an output by the transmitter.

If the MR.MSTR bit is written to zero, the SPI operates in slave mode. The MISO line is driven by the transmitter output, the MOSI line is wired on the receiver input, the SPCK pin is driven by the transmitter to synchronize the receiver. The NPCS0 pin becomes an input, and is used as a Slave Select signal (NSS). The pins NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operations. The baud rate generator is activated only in master mode.

26.7.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is configured with the Clock Polarity bit in the Chip Select Registers (CSRn.CPOL). The clock phase is configured with the Clock Phase bit in the CSRn registers (CSRn.NCPHA). These two bits determine the edges of the clock signal on which data is driven and sampled. Each of the two bits has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

Table 26-2 on page 660 shows the four modes and corresponding parameter settings.

Table 26-2. SPI modes

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 26-3 on page 660 and Figure 26-4 on page 661 show examples of data transfers.

Figure 26-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)

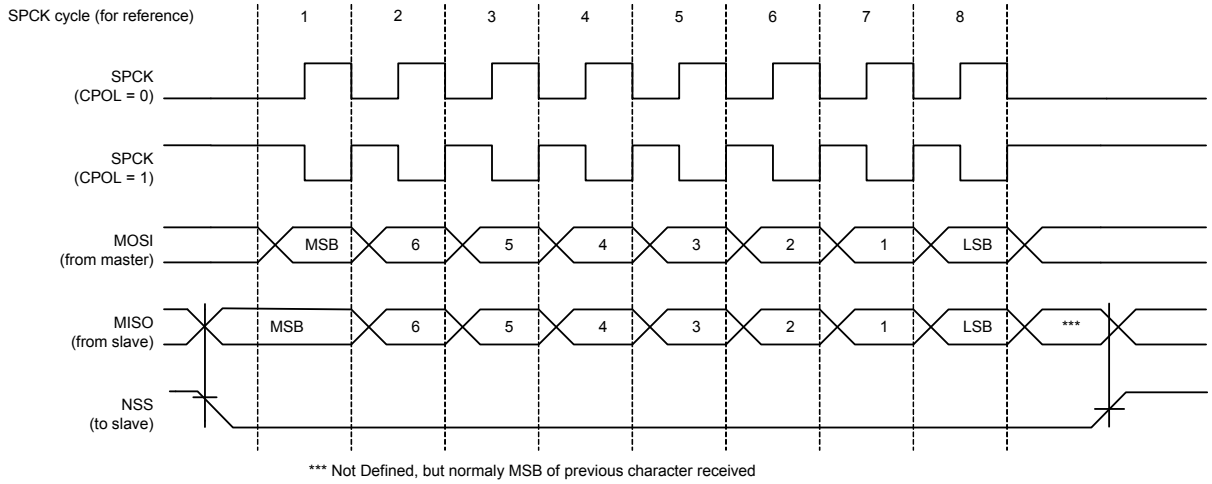
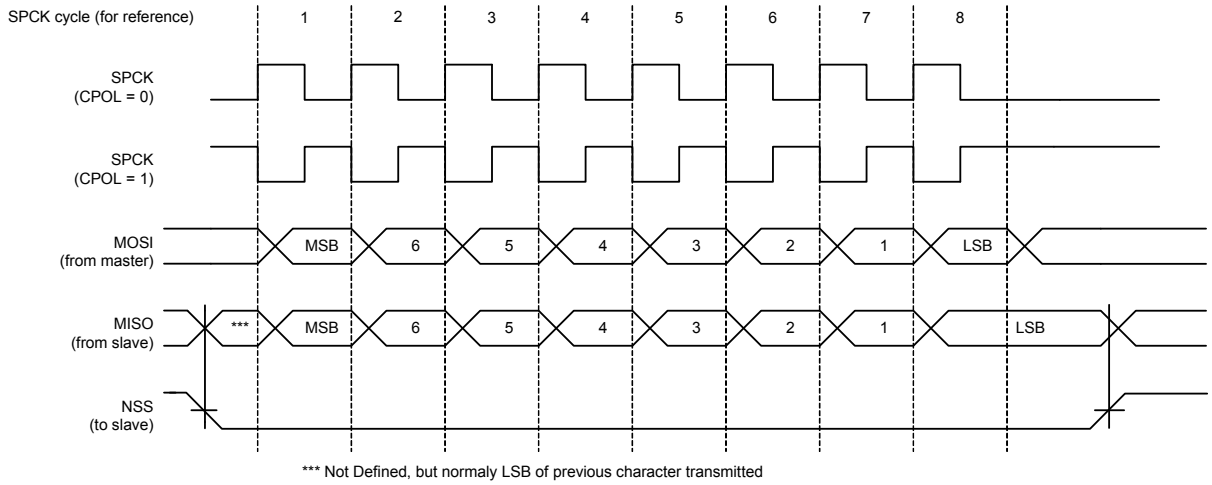


Figure 26-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



26.7.3 Master Mode Operations

When configured in master mode, the SPI uses the internal programmable baud rate generator as clock source. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register (TDR) and the Receive Data Register (RDR), and a single Shift Register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer begins when the processor writes to the TDR register. The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Transmission cannot occur without reception.

Before writing to the TDR, the Peripheral Chip Select field in TDR (TDR.PCS) must be written in order to select a slave.

If new data is written to TDR during the transfer, it stays in it until the current transfer is completed. Then, the received data is transferred from the Shift Register to RDR, the data in TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in TDR in the Shift Register is indicated by the Transmit Data Register Empty bit in the Status Register (SR.TDRE). When new data is written in TDR, this bit is cleared. The SR.TDRE bit is used to trigger the Transmit Peripheral DMA Controller channel.

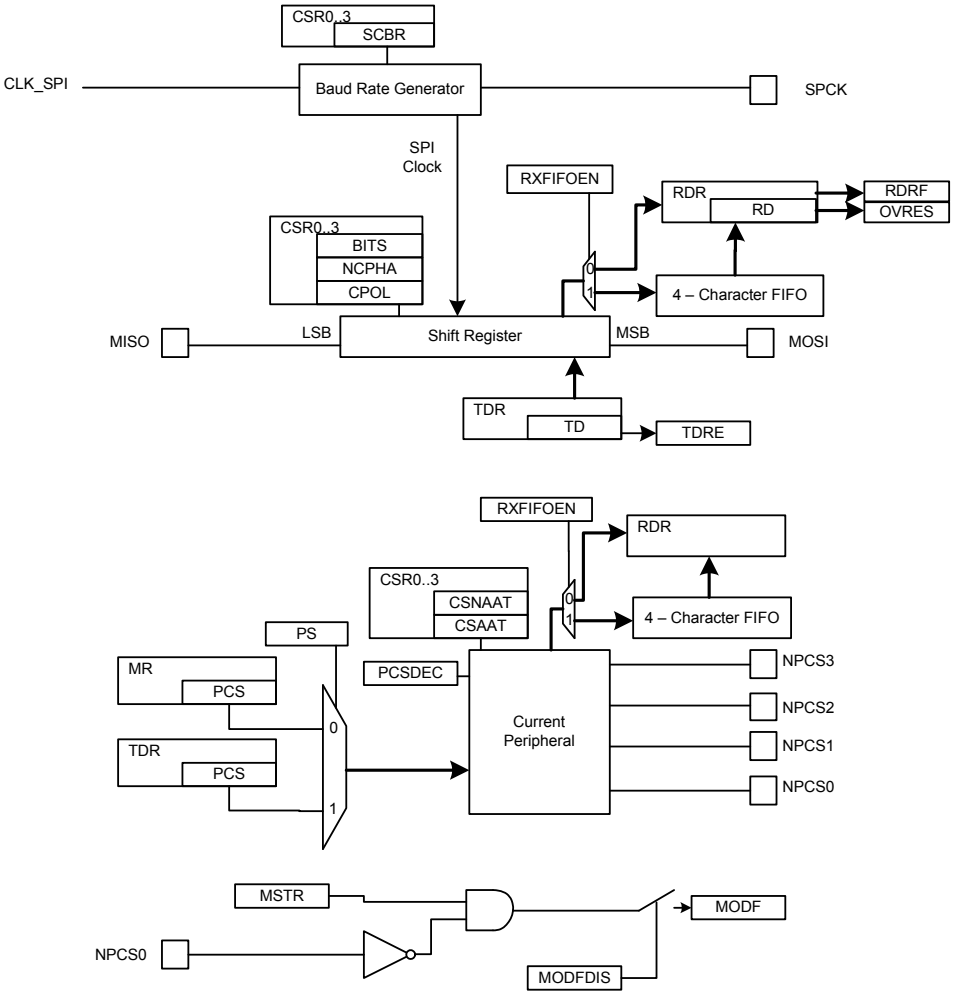
The end of transfer is indicated by the Transmission Registers Empty bit in the SR register (SR.TXEMPTY). If a transfer delay (CSRn.DLYBCT) is greater than zero for the last transfer, SR.TXEMPTY is set after the completion of said delay. The CLK_SPI can be switched off at this time.

During reception, received data are transferred from the Shift Register to the reception FIFO. The FIFO can contain up to 4 characters (both Receive Data and Peripheral Chip Select fields). While a character of the FIFO is unread, the Receive Data Register Full bit in SR remains high (SR.RDRF). Characters are read through the RDR register. If the four characters stored in the FIFO are not read and if a new character is stored, this sets the Overrun Error Status bit in the SR register (SR.OVRES). The procedure to follow in such a case is described in Section 26.7.3.8.

Figure 26-5 on page 662shows a block diagram of the SPI when operating in master mode. Figure 26-6 on page 663 shows a flow chart describing how transfers are handled.

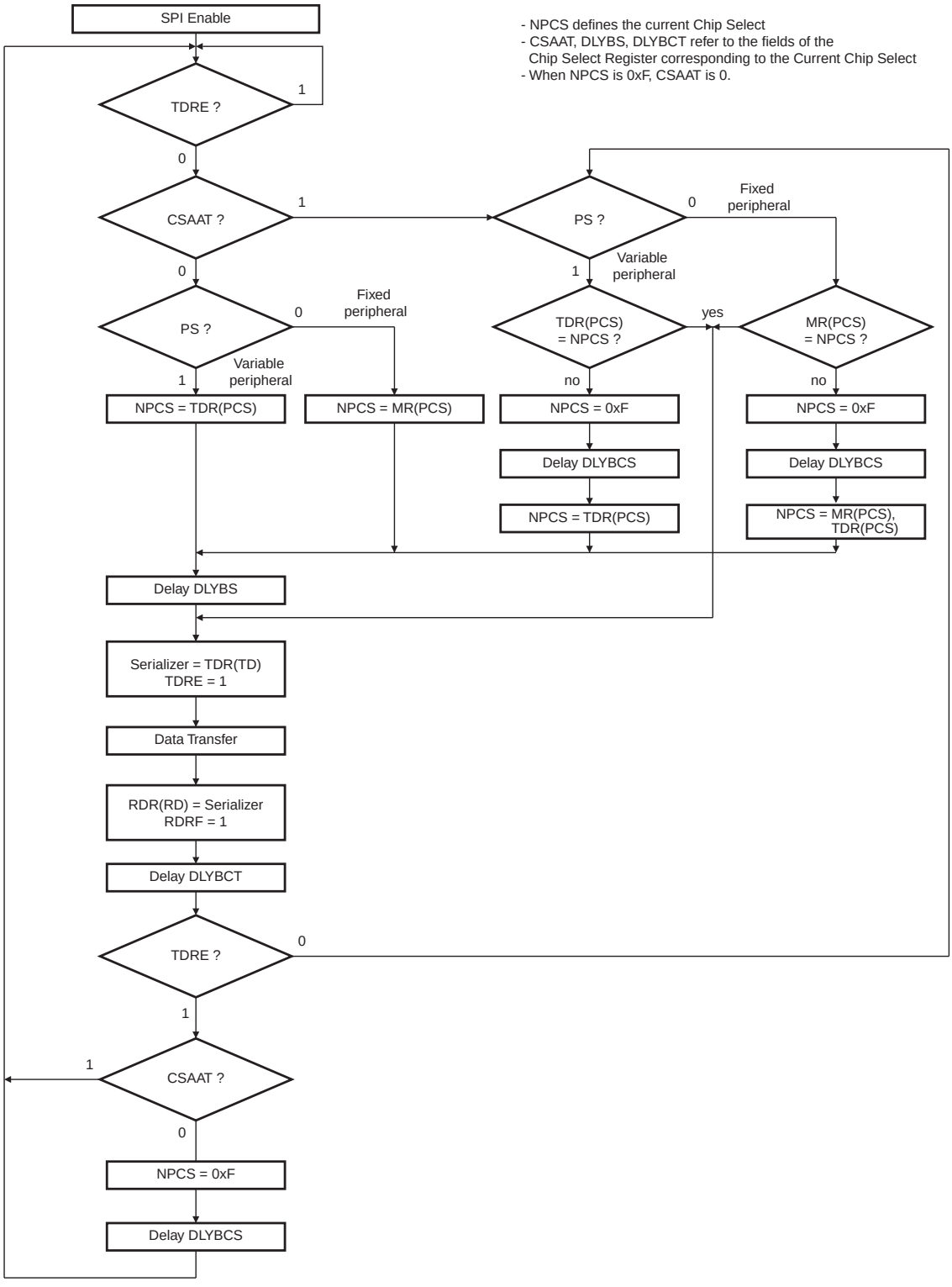
26.7.3.1 Master mode block diagram

Figure 26-5. Master Mode Block Diagram



26.7.3.2 Master mode flow diagram

Figure 26-6. Master Mode Flow Diagram



26.7.3.3 Clock generation

The SPI Baud rate clock is generated by dividing the CLK_SPI , by a value between 1 and 255.

This allows a maximum operating baud rate at up to CLK_SPI and a minimum operating baud rate of CLK_SPI divided by 255.

Writing the Serial Clock Baud Rate field in the CSRn registers (CSRn.SCBR) to zero is forbidden. Triggering a transfer while CSRn.SCBR is zero can lead to unpredictable results.

At reset, CSRn.SCBR is zero and the user has to configure it at a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be configured in the CSRn.SCBR field. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

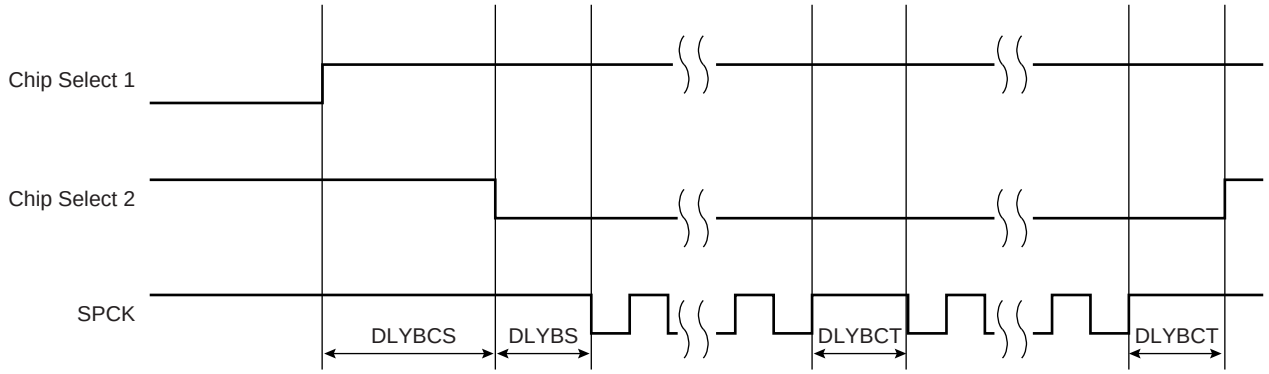
26.7.3.4 Transfer delays

Figure 26-7 on page 664 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be configured to modify the transfer waveforms:

- The delay between chip selects, programmable only once for all the chip selects by writing to the Delay Between Chip Selects field in the MR register (MR.DLYBCS). Allows insertion of a delay between release of one chip select and before assertion of a new one.
- The delay before SPCK, independently programmable for each chip select by writing the Delay Before SPCK field in the CSRn registers (CSRn.DLYBS). Allows the start of SPCK to be delayed after the chip select has been asserted.
- The delay between consecutive transfers, independently programmable for each chip select by writing the Delay Between Consecutive Transfers field in the CSRn registers (CSRn.DLYBCT). Allows insertion of a delay between two transfers occurring on the same chip select

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 26-7. Programmable Delays



26.7.3.5 Peripheral selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all the NPCS signals are high before and after each transfer.

The peripheral selection can be performed in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral
- Variable Peripheral Select: Data can be exchanged with more than one peripheral

Fixed Peripheral Select is activated by writing a zero to the Peripheral Select bit in MR (MR.PS). In this case, the current peripheral is defined by the MR.PCS field and the TDR.PCS field has no effect.

Variable Peripheral Select is activated by writing a one to the MR.PS bit . The TDR.PCS field is used to select the current peripheral. This means that the peripheral selection can be defined for each new data.

The Fixed Peripheral Selection allows buffer transfers with a single peripheral. Using the Peripheral DMA Controller is an optimal means, as the size of the data transfer between the memory and the SPI is either 4 bits or 16 bits. However, changing the peripheral selection requires the Mode Register to be reprogrammed.

The Variable Peripheral Selection allows buffer transfers with multiple peripherals without reprogramming the MR register. Data written to TDR is 32-bits wide and defines the real data to be transmitted and the peripheral it is destined to. Using the Peripheral DMA Controller in this mode requires 32-bit wide buffers, with the data in the LSBs and the PCS and LASTXFER fields in the MSBs, however the SPI still controls the number of bits (8 to16) to be transferred through MISO and MOSI lines with the CSRn registers. This is not the optimal means in term of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

26.7.3.6 Peripheral chip select decoding

The user can configure the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with an external logic. This can be enabled by writing a one to the Chip Select Decode bit in the MR register (MR.PCSDEC).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e. driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field of either the MR register or the TDR register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at one) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, the CRS0 register defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14.

26.7.3.7 Peripheral deselection

When operating normally, as soon as the transfer of the last data written in TDR is completed, the NPCS lines all rise. This might lead to runtime error if the processor is too long in responding

to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring the chip select line to remain active during a full set of transfers.

To facilitate interfacing with such devices, the CSRn registers can be configured with the Chip Select Active After Transfer bit written to one (CSRn.CSAAT) . This allows the chip select lines to remain in their current state (low = active) until transfer to another peripheral is required.

When the CSRn.CSAAT bit is written to qero, the NPCS does not rise in all cases between two transfers on the same peripheral. During a transfer on a Chip Select, the SR.TDRE bit rises as soon as the content of the TDR is transferred into the internal shifter. When this bit is detected the TDR can be reloaded. If this reload occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the Chip Select is not de-asserted between the two transfers. This might lead to difficulties for interfacing with some serial peripherals requiring the chip select to be de-asserted after each transfer. To facilitate interfacing with such devices, the CSRn registers can be configured with the Chip Select Not Active After Transfer bit (CSRn.CSNAAT) written to one. This allows to de-assert systematically the chip select lines during a time DLYBCS. (The value of the CSRn.CSNAAT bit is taken into account only if the CSRn.CSAAT bit is written to zero for the same Chip Select).

Figure 26-8 on page 667 shows different peripheral deselection cases and the effect of the CSRn.CSAAT and CSRn.CSNAAT bits.

26.7.3.8 FIFO management

A FIFO has been implemented in Reception FIFO (both in master and in slave mode), in order to be able to store up to 4 characters without causing an overrun error. If an attempt is made to store a fifth character, an overrun error rises. If such an event occurs, the FIFO must be flushed. There are two ways to Flush the FIFO:

- By performing four read accesses of the RDR (the data read must be ignored)
- By writing a one to the Flush Fifo Command bit in the CR register (CR.FLUSHFIFO).

After that, the SPI is able to receive new data.

Figure 26-8. Peripheral Deselection

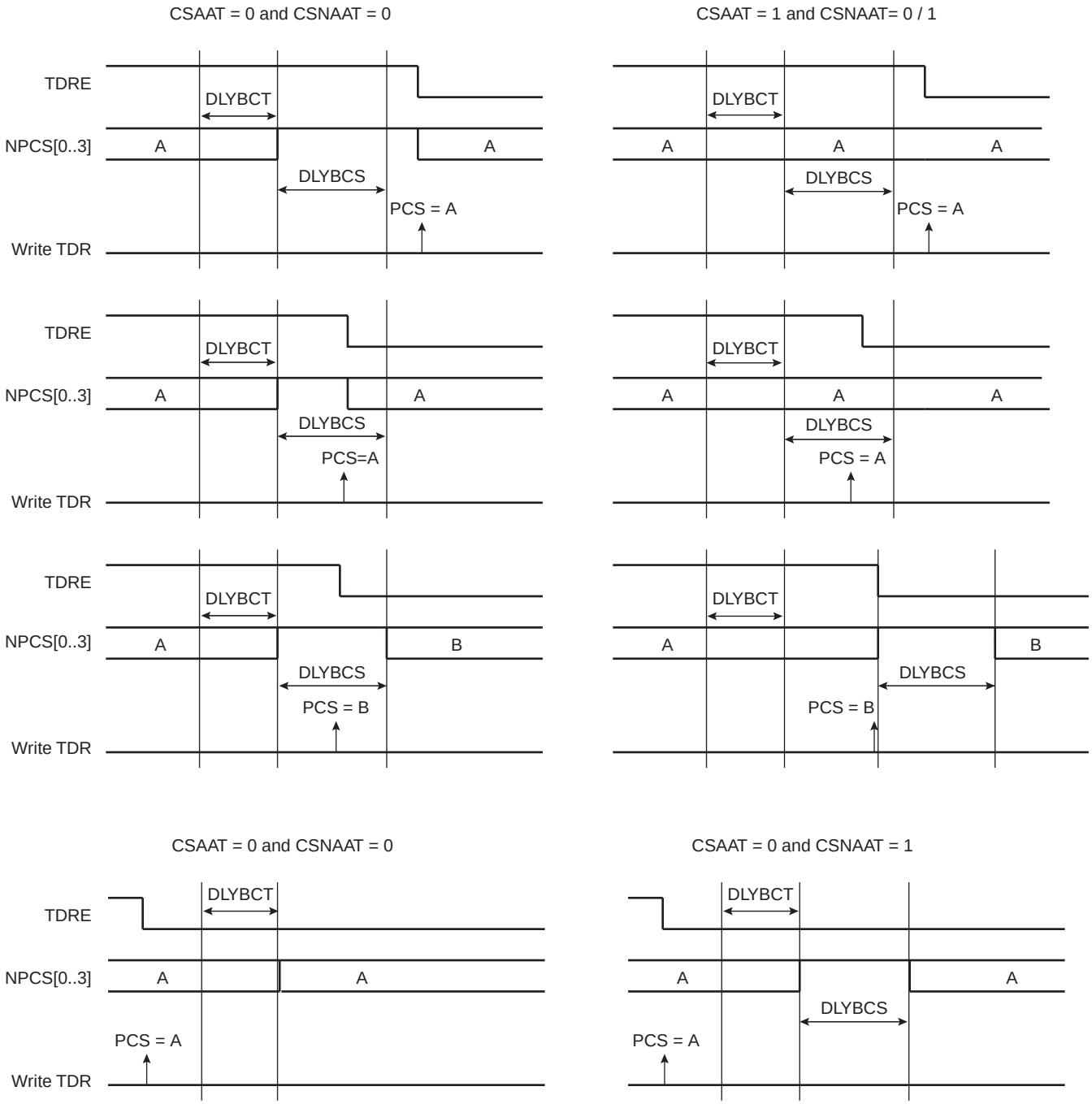


Figure 26-8 on page 667 shows different peripheral deselection cases and the effect of the CSRn.CSAAT and CSRn.CSNAAT bits.

26.7.3.9 Mode fault detection

The SPI is capable of detecting a mode fault when it is configured in master mode and NPCS0, MOSI, MISO, and SPCK are configured as open drain through the I/O Controller with either internal or external pullup resistors. If the I/O Controller does not have open-drain capability, mode fault detection **must** be disabled by writing a one to the Mode Fault Detection bit in the MR

register (MR.MODFDIS). In systems with open-drain I/O lines, a mode fault is detected when a low level is driven by an external master on the NPCS0/NSS signal.

When a mode fault is detected, the Mode Fault Error bit in the SR (SR.MODF) is set until the SR is read and the SPI is automatically disabled until re-enabled by writing a one to the SPI Enable bit in the CR register (CR.SPIEN).

By default, the mode fault detection circuitry is enabled. The user can disable mode fault detection by writing a one to the Mode Fault Detection bit in the MR register (MR.MODFDIS).

26.7.4 SPI Slave Mode

When operating in slave mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits for NSS to go active before receiving the serial clock from an external master. When NSS falls, the clock is validated on the serializer, which processes the number of bits defined by the Bits Per Transfer field of the Chip Select Register 0 (CSR0.BITS). These bits are processed following a phase and a polarity defined respectively by the CSR0.NCPHA and CSR0.CPOL bits. Note that the BITS, CPOL, and NCPHA bits of the other Chip Select Registers have no effect when the SPI is configured in Slave Mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

When all the bits are processed, the received data is transferred in the Receive Data Register and the SR.RDRF bit rises. If the RDR register has not been read before new data is received, the SR.OVRES bit is set. Data is loaded in RDR even if this flag is set. The user has to read the SR register to clear the SR.OVRES bit.

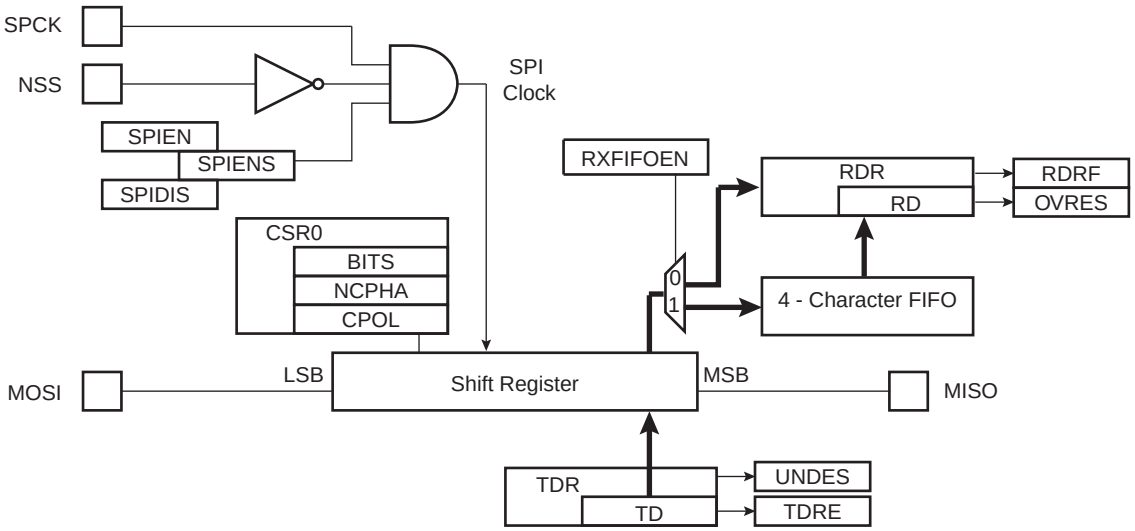
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the TDR register, the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets to zero.

When a first data is written in TDR, it is transferred immediately in the Shift Register and the SR.TDRE bit rises. If new data is written, it remains in TDR until a transfer occurs, i.e. NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in TDR is transferred in the Shift Register and the SR.TDRE bit rises. This enables frequent updates of critical variables with single transfers.

Then, a new data is loaded in the Shift Register from the TDR. In case no character is ready to be transmitted, i.e. no character has been written in TDR since the last load from TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted. In this case the Underrun Error Status bit is set in SR (SR.UNDES).

Figure 26-9 on page 669 shows a block diagram of the SPI when operating in slave mode.

Figure 26-9. Slave Mode Functional Block Diagram



26.8 User Interface

Table 26-3. SPI Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	0x00000000
0x04	Mode Register	MR	Read/Write	0x00000000
0x08	Receive Data Register	RDR	Read-only	0x00000000
0x0C	Transmit Data Register	TDR	Write-only	0x00000000
0x10	Status Register	SR	Read-only	0x00000000
0x14	Interrupt Enable Register	IER	Write-only	0x00000000
0x18	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x30	Chip Select Register 0	CSR0	Read/Write	0x00000000
0x34	Chip Select Register 1	CSR1	Read/Write	0x00000000
0x38	Chip Select Register 2	CSR2	Read/Write	0x00000000
0x3C	Chip Select Register 3	CSR3	Read/Write	0x00000000
0x E4	Write Protection Control Register	WPCR	Read/Write	0X00000000
0xE8	Write Protection Status Register	WPSR	Read-only	0x00000000
0xF8	Features Register	FEATURES	Read-only	- (1)
0xFC	Version Register	VERSION	Read-only	- (1)

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

26.8.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	LASTXFER
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	FLUSHFIFO
7	6	5	4	3	2	1	0
SWRST	-	-	-	-	-	SPIDIS	SPIEN

- **LASTXFER: Last Transfer**
  1: The current NPCS will be deasserted after the character written in TD has been transferred. When CSRn.CSAAT is one, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.
  0: Writing a zero to this bit has no effect.
- **FLUSHFIFO: Flush Fifo Command**
  1: If The FIFO Mode is enabled (MR.FIFOEN written to one) and if an overrun error has been detected, this command allows to empty the FIFO.
  0: Writing a zero to this bit has no effect.
- **SWRST: SPI Software Reset**
  1: Writing a one to this bit will reset the SPI. A software-triggered hardware reset of the SPI interface is performed. The SPI is in slave mode after software reset. Peripheral DMA Controller channels are not affected by software reset.
  0: Writing a zero to this bit has no effect.
- **SPIDIS: SPI Disable**
  1: Writing a one to this bit will disable the SPI. As soon as SPIDIS is written to one, the SPI finishes its transfer, all pins are set in input mode and no data is received or transmitted. If a transfer is in progress, the transfer is finished before the SPI is disabled. If both SPIEN and SPIDIS are equal to one when the CR register is written, the SPI is disabled.
  0: Writing a zero to this bit has no effect.
- **SPIEN: SPI Enable**
  1: Writing a one to this bit will enable the SPI to transfer and receive data.
  0: Writing a zero to this bit has no effect.

26.8.2 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCS							

23	22	21	20	19	18	17	16
-	-	-	-	PCS			

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
LLB	RXFIFOEN	-	MODFDIS	-	PCSDEC	PS	MSTR

- **DLYBCS: Delay Between Chip Selects**
  This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.
  If DLYBCS is less than or equal to six, six CLK_SPI periods will be inserted by default.
  Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{CLKSPI}$$

- **PCS: Peripheral Chip Select**
  This field is only used if Fixed Peripheral Select is active (PS = 0).
  If PCSDEC = 0:
  PCS = xxx0NPCS[3:0] = 1110
  PCS = xx01NPCS[3:0] = 1101
  PCS = x011NPCS[3:0] = 1011
  PCS = 0111NPCS[3:0] = 0111
  PCS = 1111forbidden (no peripheral is selected)
  (x = don't care)
  If PCSDEC = 1:
  NPCS[3:0] output signals = PCS.
- **LLB: Local Loopback Enable**
  1: Local loopback path enabled. LLB controls the local loopback on the data serializer for testing in master mode only (MISO is internally connected on MOSI).
  0: Local loopback path disabled.
- **RXFIFOEN: FIFO in Reception Enable**
  1: The FIFO is used in reception (four characters can be stored in the SPI).

0: The FIFO is not used in reception (only one character can be stored in the SPI).

- **MODFDIS: Mode Fault Detection**
  1: Mode fault detection is disabled. If the I/O controller does not have open-drain capability, mode fault detection **must** be disabled for proper operation of the SPI.
  0: Mode fault detection is enabled.
- **PCSDEC: Chip Select Decode**
  0: The chip selects are directly connected to a peripheral device.
  1: The four chip select lines are connected to a 4- to 16-bit decoder.
  When PCSDEC equals one, up to 15 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder. The CSRn registers define the characteristics of the 15 chip selects according to the following rules:
  CSR0 defines peripheral chip select signals 0 to 3.
  CSR1 defines peripheral chip select signals 4 to 7.
  CSR2 defines peripheral chip select signals 8 to 11.
  CSR3 defines peripheral chip select signals 12 to 14.
- **PS: Peripheral Select**
  1: Variable Peripheral Select.
  0: Fixed Peripheral Select.
- **MSTR: Master/Slave Mode**
  1: SPI is in master mode.
  0: SPI is in slave mode.

26.8.3 Receive Data Register

Name: RDR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
RD[15:8]							

7	6	5	4	3	2	1	0
RD[7:0]							

- **RD: Receive Data**
  Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

26.8.4 Transmit Data Register

Name: TDR

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	LASTXFER

23	22	21	20	19	18	17	16
-	-	-	-	PCS			

15	14	13	12	11	10	9	8
TD[15:8]							

7	6	5	4	3	2	1	0
TD[7:0]							

- **LASTXFER: Last Transfer**

  1: The current NPCS will be deasserted after the character written in TD has been transferred. When CSRn.CSAAT is one, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.
  0: Writing a zero to this bit has no effect.

  This field is only used if Variable Peripheral Select is active (MR.PS = 1).

- **PCS: Peripheral Chip Select**

  If PCSDEC = 0:
  PCS = xxx0NPCS[3:0] = 1110
  PCS = xx01NPCS[3:0] = 1101
  PCS = x011NPCS[3:0] = 1011
  PCS = 0111NPCS[3:0] = 0111
  PCS = 1111forbidden (no peripheral is selected)
  (x = don’t care)
  If PCSDEC = 1:
  NPCS[3:0] output signals = PCS

  This field is only used if Variable Peripheral Select is active (MR.PS = 1).

- **TD: Transmit Data**

  Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the TDR register in a right-justified format.

26.8.5 Status Register

Name: SR

Access Type: Read-only

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	SPIENS

15	14	13	12	11	10	9	8
-	-	-	-	-	UNDES	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
-	-	-	-	OVRES	MODF	TDRE	RDRF

- **SPIENS: SPI Enable Status**
  1: This bit is set when the SPI is enabled.
  0: This bit is cleared when the SPI is disabled.
- **UNDES: Underrun Error Status (Slave Mode Only)**
  1: This bit is set when a transfer begins whereas no data has been loaded in the TDR register.
  0: This bit is cleared when the SR register is read.
- **TXEMPTY: Transmission Registers Empty**
  1: This bit is set when TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.
  0: This bit is cleared as soon as data is written in TDR.
- **NSSR: NSS Rising**
  1: A rising edge occurred on NSS pin since last read.
  0: This bit is cleared when the SR register is read.
- **OVRES: Overrun Error Status**
  1: This bit is set when an overrun has occurred. An overrun occurs when RDR is loaded at least twice from the serializer since the last read of the RDR.
  0: This bit is cleared when the SR register is read.
- **MODF: Mode Fault Error**
  1: This bit is set when a Mode Fault occurred.
  0: This bit is cleared when the SR register is read.
- **TDRE: Transmit Data Register Empty**
  1: This bit is set when the last data written in the TDR register has been transferred to the serializer.
  0: This bit is cleared when data has been written to TDR and not yet transferred to the serializer.
  TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.
- **RDRF: Receive Data Register Full**
  1: Data has been received and the received data has been transferred from the serializer to RDR since the last read of RDR.
  0: No data has been received since the last read of RDR

26.8.6 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
-	-	-	-	OVRES	MODF	TDRE	RDRF

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

26.8.7 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
-	-	-	-	OVRES	MODF	TDRE	RDRF

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

26.8.8 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
-	-	-	-	OVRES	MODF	TDRE	RDRF

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

26.8.9 Chip Select Register 0

Name: CSR0

Access Type: Read/Write

Offset: 0x30

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

- **DLYBCT: Delay Between Consecutive Transfers**

  This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

  When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

  Otherwise, the following equation determines the delay:

  $$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{CLKSPI}$$

- **DLYBS: Delay Before SPCK**

  This field defines the delay from NPCS valid to the first valid SPCK transition.

  When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

  Otherwise, the following equations determine the delay:

  $$\text{Delay Before SPCK} = \frac{DLYBS}{CLKSPI}$$

- **SCBR: Serial Clock Baud Rate**

  In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the CLK_SPI. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

  $$\text{SPCK Baudrate} = \frac{CLKSPI}{SCBR}$$

  Writing the SCBR field to zero is forbidden. Triggering a transfer while SCBR is zero can lead to unpredictable results.

  At reset, SCBR is zero and the user has to write it to a valid value before performing the first transfer.

  If a clock divider (SCBRn) field is set to one and the other SCBR fields differ from one, access on CSn is correct but no correct access will be possible on other CS.

- **BITS: Bits Per Transfer**
  The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  1: The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured after the leading (inactive-to-active) edge of SPCK and changed on the trailing (active-to-inactive) edge of SPCK.
  0: Data is changed on the leading (inactive-to-active) edge of SPCK and captured after the trailing (active-to-inactive) edge of SPCK.
  NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

26.8.10 Chip Select Register 1

Name: CSR1

Access Type: Read/Write

Offset: 0x34

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

- **DLYBCT: Delay Between Consecutive Transfers**

  This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

  When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

  Otherwise, the following equation determines the delay:

  $$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{CLKSPI}$$

- **DLYBS: Delay Before SPCK**

  This field defines the delay from NPCS valid to the first valid SPCK transition.

  When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

  Otherwise, the following equations determine the delay:

  $$\text{Delay Before SPCK} = \frac{DLYBS}{CLKSPI}$$

- **SCBR: Serial Clock Baud Rate**

  In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the CLK_SPI. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

  $$\text{SPCK Baudrate} = \frac{CLKSPI}{SCBR}$$

  Writing the SCBR field to zero is forbidden. Triggering a transfer while SCBR is zero can lead to unpredictable results.

  At reset, SCBR is zero and the user has to write it to a valid value before performing the first transfer.

  If a clock divider (SCBRn) field is set to one and the other SCBR fields differ from one, access on CSn is correct but no correct access will be possible on other CS.

- **BITS: Bits Per Transfer**
  The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  1: The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured after the leading (inactive-to-active) edge of SPCK and changed on the trailing (active-to-inactive) edge of SPCK.
  0: Data is changed on the leading (inactive-to-active) edge of SPCK and captured after the trailing (active-to-inactive) edge of SPCK.
  NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

26.8.11 Chip Select Register 2

Name: CSR2

Access Type: Read/Write

Offset: 0x38

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

- **DLYBCT: Delay Between Consecutive Transfers**

  This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

  When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

  Otherwise, the following equation determines the delay:

  $$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{CLKSPI}$$

- **DLYBS: Delay Before SPCK**

  This field defines the delay from NPCS valid to the first valid SPCK transition.

  When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

  Otherwise, the following equations determine the delay:

  $$\text{Delay Before SPCK} = \frac{DLYBS}{CLKSPI}$$

- **SCBR: Serial Clock Baud Rate**

  In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the CLK_SPI. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

  $$\text{SPCK Baudrate} = \frac{CLKSPI}{SCBR}$$

  Writing the SCBR field to zero is forbidden. Triggering a transfer while SCBR is zero can lead to unpredictable results.

  At reset, SCBR is zero and the user has to write it to a valid value before performing the first transfer.

  If a clock divider (SCBRn) field is set to one and the other SCBR fields differ from one, access on CSn is correct but no correct access will be possible on other CS.

- **BITS: Bits Per Transfer**
  The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  1: The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured after the leading (inactive-to-active) edge of SPCK and changed on the trailing (active-to-inactive) edge of SPCK.
  0: Data is changed on the leading (inactive-to-active) edge of SPCK and captured after the trailing (active-to-inactive) edge of SPCK.
  NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

26.8.12 Chip Select Register 3

Name: CSR3

Access Type: Read/Write

Offset: 0x3C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

- **DLYBCT: Delay Between Consecutive Transfers**

  This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

  When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

  Otherwise, the following equation determines the delay:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{CLKSPI}$$

- **DLYBS: Delay Before SPCK**

  This field defines the delay from NPCS valid to the first valid SPCK transition.

  When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

  Otherwise, the following equations determine the delay:

$$\text{Delay Before SPCK} = \frac{DLYBS}{CLKSPI}$$

- **SCBR: Serial Clock Baud Rate**

  In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the CLK_SPI. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

$$\text{SPCK Baudrate} = \frac{CLKSPI}{SCBR}$$

  Writing the SCBR field to zero is forbidden. Triggering a transfer while SCBR is zero can lead to unpredictable results.

  At reset, SCBR is zero and the user has to write it to a valid value before performing the first transfer.

  If a clock divider (SCBRn) field is set to one and the other SCBR fields differ from one, access on CSn is correct but no correct access will be possible on other CS.

- **BITS: Bits Per Transfer**
  The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  1: The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured after the leading (inactive-to-active) edge of SPCK and changed on the trailing (active-to-inactive) edge of SPCK.
  0: Data is changed on the leading (inactive-to-active) edge of SPCK and captured after the trailing (active-to-inactive) edge of SPCK.
  NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

26.8.13 Write Protection Control Register

Register Name: WPCR

Access Type: Read-write

Offset: 0xE4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SPIWPKEY[23:16]							

23	22	21	20	19	18	17	16
SPIWPKEY[15:8]							

15	14	13	12	11	10	9	8
SPIWPKEY[7:0]							

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	SPIWPEN

- **SPIWPKEY: SPI Write Protection Key Password**

  If a value is written in SPIWPEN, the value is taken into account only if SPIWPKEY is written with “SPI” (SPI written in ASCII Code, i.e. 0x535049 in hexadecimal).
- **SPIWPEN: SPI Write Protection Enable**

  1: The Write Protection is Enabled

  0: The Write Protection is Disabled

26.8.14 Write Protection Status Register

Register Name: WPSR

Access Type: Read-only

Offset: 0xE8

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
SPIWPVSRC							

7	6	5	4	3	2	1	0
-	-	-	-	-	SPIWPVS		

- **SPIWPVSRC: SPI Write Protection Violation Source**
  This Field indicates the Peripheral Bus Offset of the register concerned by the violation (MR or CSRx)
- **SPIWPVS: SPI Write Protection Violation Status**

SPIWPVS value	Violation Type
1	The Write Protection has blocked a Write access to a protected register (since the last read).
2	Software Reset has been performed while Write Protection was enabled (since the last read or since the last write access on MR, IER, IDR or CSRx).
3	Both Write Protection violation and software reset with Write Protection enabled have occurred since the last read.
4	Write accesses have been detected on MR (while a chip select was active) or on CSRi (while the Chip Select “i” was active) since the last read.
5	The Write Protection has blocked a Write access to a protected register and write accesses have been detected on MR (while a chip select was active) or on CSRi (while the Chip Select “i” was active) since the last read.
6	Software Reset has been performed while Write Protection was enabled (since the last read or since the last write access on MR, IER, IDR or CSRx) and some write accesses have been detected on MR (while a chip select was active) or on CSRi (while the Chip Select “i” was active) since the last read.
7	- The Write Protection has blocked a Write access to a protected register. and - Software Reset has been performed while Write Protection was enabled. and - Write accesses have been detected on MR (while a chip select was active) or on CSRi (while the Chip Select “i” was active) since the last read.

26.8.15 Features Register

Register Name: FEATURES

Access Type: Read-only

Offset: 0xF8

Reset Value: –

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	SWIMPL	FIFORIMPL	BRPBHSB	CSNAATIMPL	EXTDEC

15	14	13	12	11	10	9	8
LENNCONF							LENCONF

7	6	5	4	3	2	1	0
PHZNCONF	PHCONF	PPNCONF	PCONF	NCS			

- **SWIMPL: Spurious Write Protection Implemented**
  0: Spurious write protection is not implemented.
  1: Spurious write protection is implemented.
- **FIFORIMPL: FIFO in Reception Implemented**
  0: FIFO in reception is not implemented.
  1: FIFO in reception is implemented.
- **BRPBHSB: Bridge Type is PB to HSB**
  0: Bridge type is not PB to HSB.
  1: Bridge type is PB to HSB.
- **CSNAATIMPL: CSNAAT Features Implemented**
  0: CSNAAT (Chip select not active after transfer) features are not implemented.
  1: CSNAAT features are implemented.
- **EXTDEC: External Decoder True**
  0: External decoder capability is not implemented.
  1: External decoder capability is implemented.
- **LENNCONF: Character Length if not Configurable**
  If the character length is not configurable, this field specifies the fixed character length.
- **LENCONF: Character Length Configurable**
  0: The character length is not configurable.
  1: The character length is configurable.
- **PHZNCONF: Phase is Zero if Phase not Configurable**
  0: If phase is not configurable, phase is non-zero.
  1: If phase is not configurable, phase is zero.
- **PHCONF: Phase Configurable**
  0: Phase is not configurable.
  1: Phase is configurable.

- **PPNCONF: Polarity Positive if Polarity not Configurable**
  0: If polarity is not configurable, polarity is negative.
  1: If polarity is not configurable, polarity is positive.
- **PCONF: Polarity Configurable**
  0: Polarity is not configurable.
  1: Polarity is configurable.
- **NCS: Number of Chip Selects**
  This field indicates the number of chip selects implemented.

26.8.16 Version Register

Register Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: –

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	MFN			

15	14	13	12	11	10	9	8
				VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **MFN**

  Reserved. No functionality associated.

- **VERSION**

  Version number of the module. No functionality associated.

26.9 Module Configuration

The specific configuration for each SPI instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 26-4. SPI Clock Name

Module Name	Clock Name	Description
SPI0	CLK_SPI0	Peripheral Bus clock from the PBC clock domain
SPI1	CLK_SPI1	Peripheral Bus clock from the PBA clock domain

Table 26-5. Register Reset Values

Register	Reset Value
FEATURES	0x001F0154
VERSION	0x00000211

27. Two-Wire Master Interface (TWIM)

Rev: 1.1.0.1

27.1 Features

- **Compatible with I²C standard**
  - **Multi-master support**
  - **100 and 400 kbit/sin transfer speed**
  - **7- and 10-bit and General Call addressing**
- **Compatible with SMBus standard**
  - **Hardware Packet Error Checking (CRC) generation and verification with ACK control**
  - **SMBus ALERT interface**
  - **25 ms clock low timeout delay**
  - **10 ms master cumulative clock low extend time**
  - **25 ms slave cumulative clock low extend time**
- **Compatible with PMBus**
- **Compatible with Atmel Two-Wire Interface Serial Memories**
- **DMA interface for reducing CPU load**
- **Arbitrary transfer lengths, including 0 data bytes**
- **Optional clock stretching if transmit or receive buffers not ready for data transfer**

27.2 Overview

The Atmel Two-wire Interface Master (TWIM) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 kbit/s, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus serial EEPROM and I²C compatible device such as a real rime clock (RTC), dot matrix/graphic LCD controller and temperature sensor, to name a few. TWIM is always a bus master and can transfer sequential or single bytes. Multiple master capability is supported. Arbitration of the bus is performed internally and relinquishes the bus automatically if the bus arbitration is lost.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.Table 27-1 on page 698 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I²C compatible device.

Table 27-1. Atmel TWIM Compatibility with I²C Standard

I²C Standard	Atmel TWIM
Standard-mode (100 kbit/s)	Supported
Fast-mode (400 kbit/s)	Supported
Fast-mode Plus (1 Mbit/s)	Supported
7- or 10-bits Slave Addressing	Supported
START BYTE(1)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Slope Control and Input Filtering (Fast mode)	Supported
Clock Stretching	Supported

Note: 1. START + b000000001 + Ack + Sr

Table 27-2 on page 699 lists the compatibility level of the Atmel Two-wire Master Interface and a full SMBus compatible master.

Table 27-2. Atmel TWIM Compatibility with SMBus Standard

SMBus Standard	Atmel TWIM
Bus Timeouts	Supported
Address Resolution Protocol	Supported
Alert	Supported
Host Functionality	Supported
Packet Error Checking	Supported

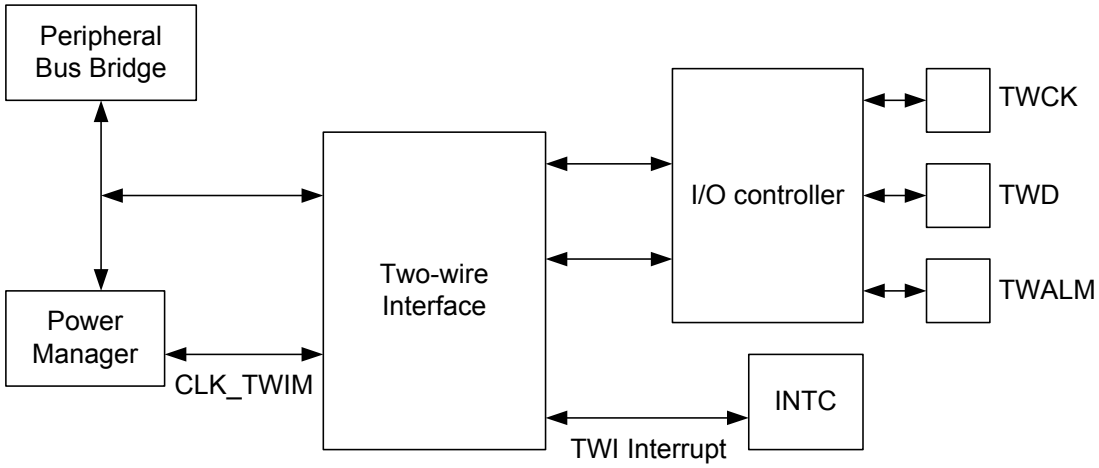
27.3 List of Abbreviations

Table 27-3. Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

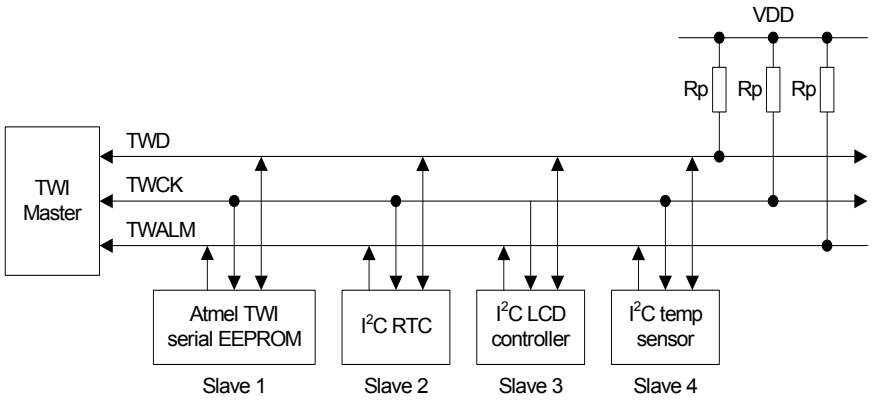
27.4 Block Diagram

Figure 27-1. Block Diagram



27.5 Application Block Diagram2

Figure 27-2. Application Block Diagram



Rp: pull-up value as given by the I2C Standard

27.6 I/O Lines Description

Table 27-4. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output
TWALM	SMBus SMBALERT	Input/Output

27.7 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

27.7.1 I/O Lines

TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 27-4 on page 702). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWALM is used to implement the optional SMBus SMBALERT signal.

The TWALM, TWD, and TWCK pins may be multiplexed with I/O Controller lines. To enable the TWIM, the programmer must perform the following steps:

- Program the I/O Controller to:
  - Dedicate TWD, TWCK and optionally TWALM as peripheral lines.
  - Define TWD, TWCK and optionally TWALM as open-drain.

27.7.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the TWIM, the TWIM will stop functioning and resume operation after the system wakes up from sleep mode.

27.7.3 Clocks

The clock for the TWIM bus interface (CLK_TWIM) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the TWIM before disabling the clock, to avoid freezing the TWIM in an undefined state.

27.7.4 DMA

The TWIM DMA handshake interface is connected to the Peripheral DMA Controller. Using the TWIM DMA functionality requires the Peripheral DMA Controller to be programmed after setting up the TWIM.

27.7.5 Interrupts

The TWIM interrupt request lines are connected to the interrupt controller. Using the TWIM interrupts requires the interrupt controller to be programmed first.

27.7.6 Debug Operation

When an external debugger forces the CPU into debug mode, the TWIM continues normal operation. If the TWIM is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

27.8 Functional Description

27.8.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 27-4 on page 702).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 27-4 on page 702).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

Figure 27-3. START and STOP Conditions

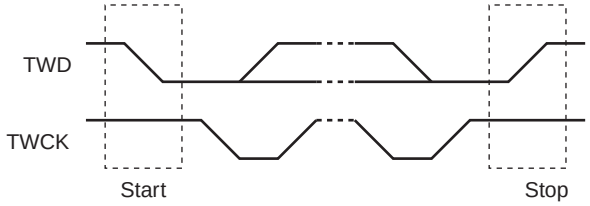
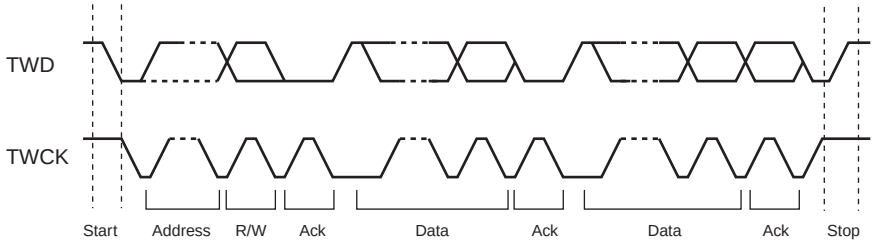


Figure 27-4. Transfer Format



27.8.2 Operation

The TWIM has two modes of operation:

- Master transmitter mode
- Master receiver mode

The master is the device which starts and stops a transfer and generates the TWCK clock. These modes are described in the following chapters.

27.8.2.1 Clock Generation

The The Clock Waveform Generator Register (CWGR) is used to control the waveform of the TWCK clock. CWGR must be programmed so that the desired TWI bus timings are generated. CWGR describes bus timings as a function of cycles of a prescaled clock. The clock prescaling can be selected through the EXP field in CWGR.

$$f_{prescaled} = \frac{f_{clkpb}}{2^{(EXP+1))}}$$

CWGR has the following fields:

LOW: Prescaled clock cycles in clock low count. Used to time T_{LOW}. and T_{BUF}.

HIGH: Prescaled clock cycles in clock high count. Used to time T_{HIGH}.

STASTO: Prescaled clock cycles in clock high count. Used to time T_{HD_STA}, T_{SU_STA}, T_{SU_STO}.

DATA: Prescaled clock cycles for data setup and hold count. Used to time T_{HD_DAT}, T_{SU_DAT}.

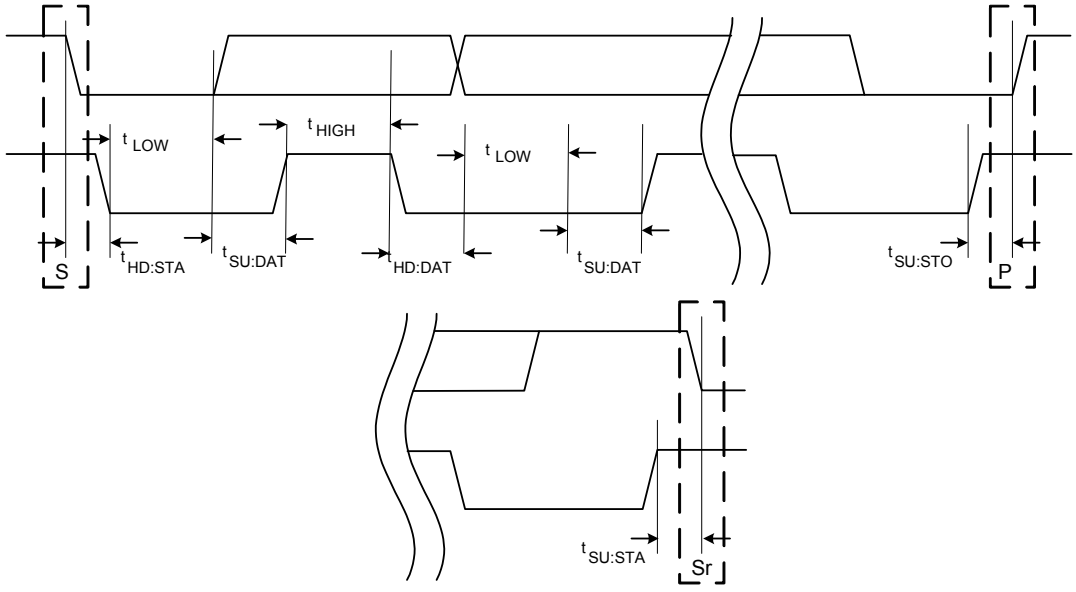
EXP: Specifies the clock prescaler setting.

Note that the total clock low time generated is the sum of T_{HD_DAT} + T_{SU_DAT} + T_{LOW}.

Any slave or other bus master taking part in the transfer may extend the TWCK low period at any time.

The TWIM hardware monitors the state of the TWCK line as required by the I²C specification. The clock generation counters are started when a high/low level is detected on the TWCK line, not when the TWIM hardware releases/drives the TWCK line. This means that the CWGR settings alone do not determine the TWCK frequency. The CWGR settings determine the clock low time and the clock high time, but the TWCK rise and fall times are determined by the external circuitry (capacitive load, etc.).

Figure 27-5. Bus Timing Diagram



27.8.2.2 Setting up and Performing a Transfer

Operation of TWIM is mainly controlled by the Control Register (CR) and the Command Register (CMDR). The following list presents the main steps in a typical communication:

1. Before any transfers can be performed, bus timings must be configured by programming the Clock Waveform Generator Register (CWGR). If operating in SMBus mode, the SMBus Timing Register (SMBTR) register must also be configured.
2. If a DMA controller is to be used for the transfers, it must be set up.
3. CMDR or NCMDR must be programmed with a value describing the transfer to be performed.

The interrupt system can be set up to give interrupt request on specific events or error conditions, for example when the transfer is complete or if arbitration is lost.

The controller will refuse to start a new transfer while ANAK, DNAK or ARBLST is set in the Status Register (SR). This is necessary to avoid a race when the software issues a continuation of the current transfer at the same time as one of these errors happen. Also, if ANAK or DNAK occur, a STOP condition is sent automatically. The programmer will have to restart the transmission by clearing the errors bit in SR after resolving the cause for the NACK.

After a data or address NACK from the slave, a STOP will be transmitted automatically. Note that the VALID bit in CMDR is NOT cleared in this case. If this transfer is to be discarded, the VALID bit can be cleared manually allowing any command in NCMDR to be copied into CMDR.

When a data or address NACK is returned by the slave while the master is transmitting, it is possible that new data has already been written to the THR register. This data will be transferred out as the first data byte of the next transfer. If this behavior is to be avoided, the safest approach is to perform a software reset of the TWIM.

27.8.3 Master Transmitter Mode

A START condition is transmitted and master transmitter mode is initiated when the bus is free and CMDR has been written with START=1 and READ=0. START and SADR+W will then be transmitted. During the address acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to acknowledge the address. The master polls the data line during this clock pulse and sets the Address Not Acknowledged bit (ANAK**)** in the Status Register if no slave acknowledges the address.

After the address phase, the following is repeated:

while (NBYTES>0)

1. Wait until THR contains a valid data byte, stretching low period of TWCK. SR.TXRDY indicates the state of THR. Software or a DMA controller must write the data byte to THR.
2. Transmit this data byte
3. Decrement NBYTES
4. If (NBYTES==0) and STOP=1, transmit STOP condition

Programming CMDR with START=STOP=1 and NBYTES=0 will generate a transmission with no data bytes, ie START, SADR+W, STOP.

TWI transfers require the slave to acknowledge each received data byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the Data Acknowledge bit (DNACK) in the Status Register if the slave does not

acknowledge the data byte. As with the other status bits, an interrupt can be generated if enabled in the Interrupt Enable Register (TWIM_IER).

TXRDY is used as Transmit Ready for the Peripheral DMA Controller transmit channel.

The end of a command is marked by setting the SR.CCOMP bit to one. See Figure 27-6 on page 705 and Figure 27-7 on page 705.

Figure 27-6. Master Write with One Data Byte

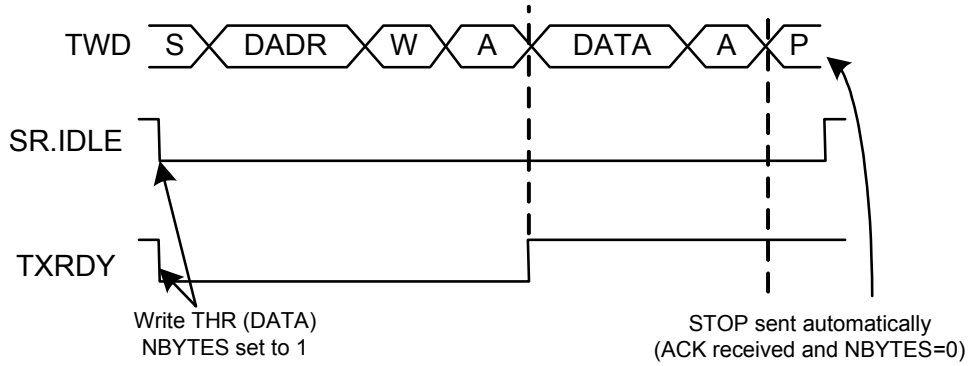
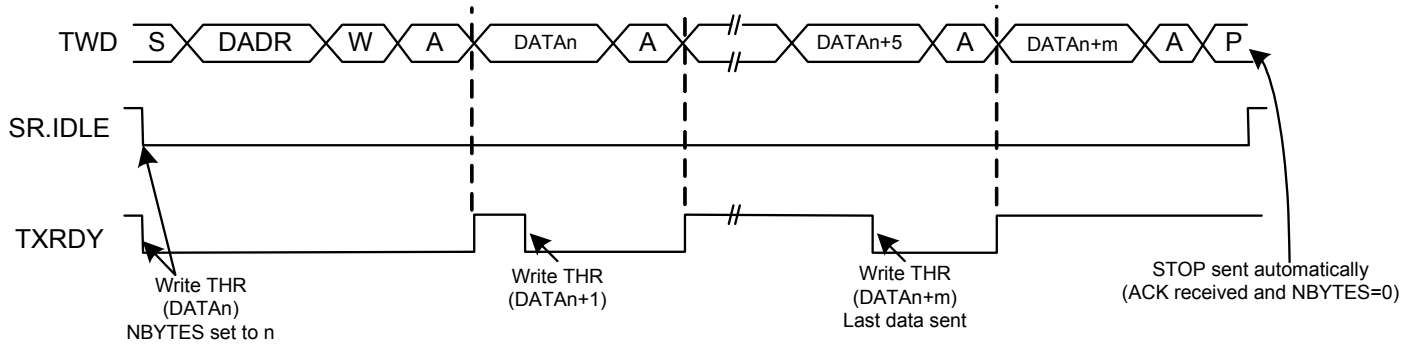


Figure 27-7. Master Write with Multiple Data Bytes



27.8.4 Master Receiver Mode

A START condition is transmitted and master receiver mode is initiated when the bus is free and CMDR has been written with START=1 and READ=1. START and SADR+R will then be transmitted. During the address acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to acknowledge the address. The master polls the data line during this clock pulse and sets the Address Not Acknowledged bit (ANAK) in the Status Register if no slave acknowledges the address.

After the address phase, the following is repeated:

while (NBYTES>0)

1. Wait until RHR is empty, stretching low period of TWCK. SR.RXRDY indicates the state of RHR. Software or a DMA controller must read any data byte present in RHR.
2. Release TWCK generating a clock that the slave uses to transmit a data byte.
3. Place the received data byte in RHR, set RXRDY.
4. If NBYTES=0, generate a NAK after the data byte, otherwise generate an ACK.
5. Decrement NBYTES

6. If (NBYTES==0) and STOP=1, transmit STOP condition.

Programming CMDR with START=STOP=1 and NBYTES=0 will generate a transmission with no data bytes, ie START, DADR+R, STOP

The TWI transfers require the master to acknowledge each received data byte. During the acknowledge clock pulse (9th pulse), the slave releases the data line (HIGH), enabling the master to pull it down in order to generate the acknowledge. All data bytes except the last are acknowledged by the master. Not acknowledging the last byte informs the slave that the transfer is finished.

RXRDY is used as Receive Ready for the Peripheral DMA Controller receive channel.

Figure 27-8. Master Read with One Data Byte

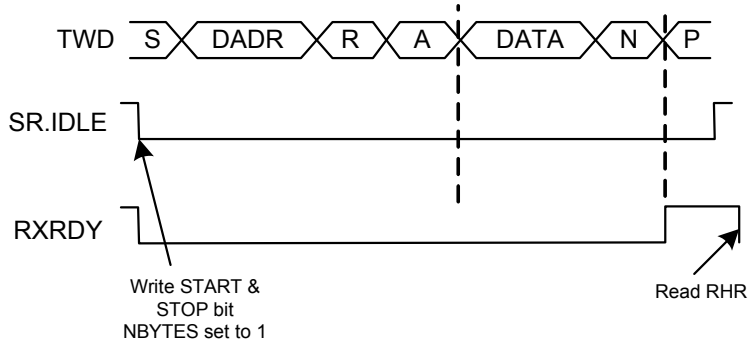
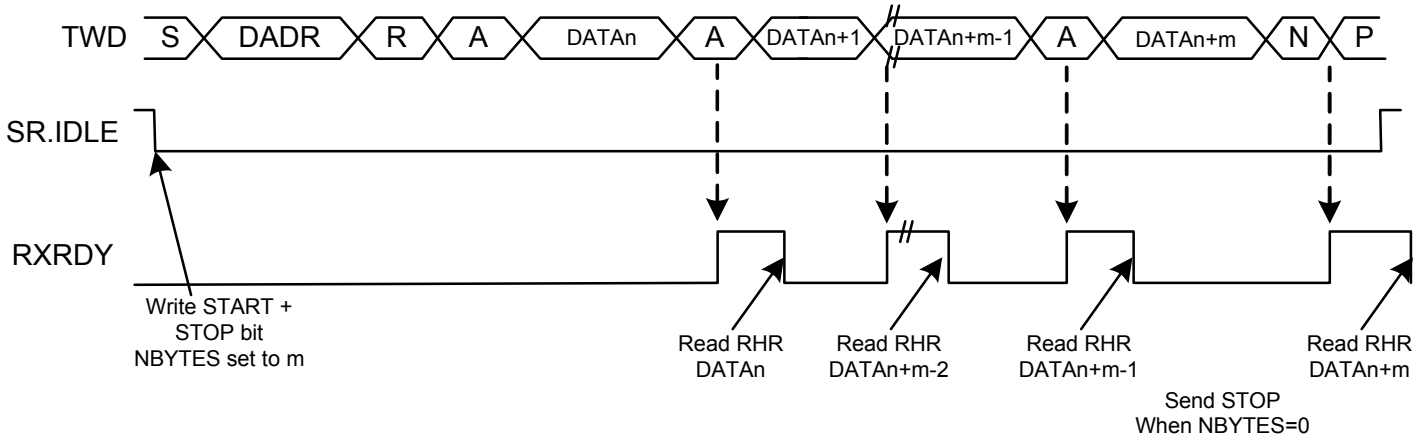


Figure 27-9. Master Read with Multiple Data Bytes



27.8.5 Using the Peripheral DMA Controller

The use of the Peripheral DMA Controller significantly reduces the CPU load. The programmer can set up ring buffers for the DMA controller, containing data to transmit or free buffer space to place received data.

To assure correct behavior, respect the following programming sequences:

27.8.5.1 *Data Transmit with the Peripheral DMA Controller*

1. Initialize the transmit Peripheral DMA Controller (memory pointers, size, etc.).
2. Configure the TWIM (ADR, NBYTES, etc.).
3. Start the transfer by setting the Peripheral DMA Controller TXTEN bit.
4. Wait for the Peripheral DMA Controller end TX flag.
5. Disable the Peripheral DMA Controller by setting the Peripheral DMA Controller TXDIS bit.

27.8.5.2 *Data Receive with the Peripheral DMA Controller*

1. Initialize the receive Peripheral DMA Controller (memory pointers, size, etc.).
2. Configure the TWIM (ADR, NBYTES, etc.).
3. Start the transfer by setting the Peripheral DMA Controller RXTEN bit.
4. Wait for the Peripheral DMA Controller end RX flag.
5. Disable the Peripheral DMA Controller by setting the Peripheral DMA Controller RXDIS bit.

27.8.6 Multi-master Mode

More than one master may access the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as arbitration is lost by a master, it stops sending data and listens to the bus in order to detect a STOP. The SR.ARBLST flag will be set. When the STOP is detected, the master who lost arbitration may reinitiate the data transfer.

Arbitration is illustrated in Figure 27-11 on page 708.

If the user starts a transfer and if the bus is busy, TWIM automatically waits for a STOP condition on the bus before initiating the transfer (see Figure 27-10 on page 707).

Note: The state of the bus (busy or free) is not indicated in the user interface.

Figure 27-10. Programmer Sends Data While the Bus is Busy

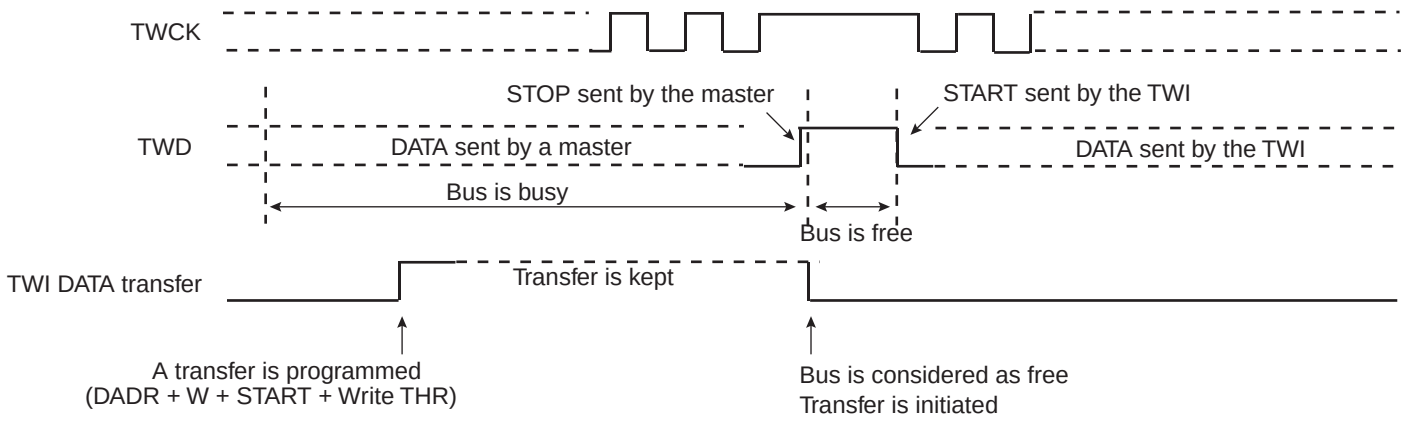
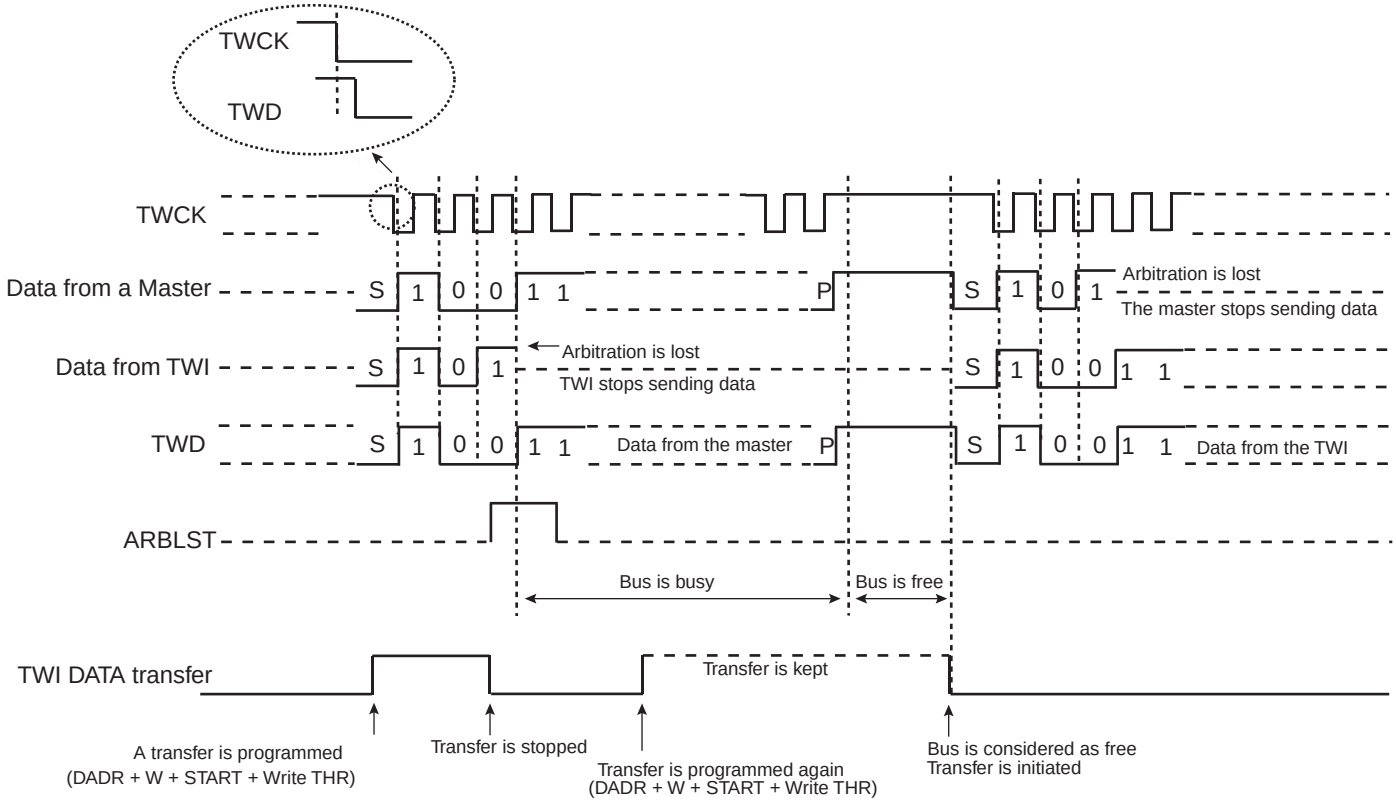


Figure 27-11. Arbitration Cases



27.8.7 Combined Transfers

CMDR and NCMDR may be used to generate longer sequences of connected transfers, since generation of START and/or STOP conditions is programmable on a per-command basis.

Programming NCMDR with START=1 when the previous transfer was programmed with STOP=0 will cause a REPEATED START on the bus. The ability to generate such connected transfers allows arbitrary transfer lengths, since it is legal to program CMDR with both START=0 and STOP=0. If this is done in master receiver mode, the CMDR.ACKLAST bit must also be controlled.

As for single data transfers, the TXRDY and RXRDY bits in the Status Register indicates when data to transmit can be written to the THR, or when received data can be read from RHR. Transfer of data to THR and from RHR can also be done automatically by DMA, see ”Using the Peripheral DMA Controller” on page 706

27.8.7.1 Write Followed by Write

Consider the following transfer:

START, DADR+W, DATA+A, DATA+A, REPSTART, DADR+W, DATA+A, DATA+A, STOP.

To generate this transfer:

1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=0.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=0.
3. Wait until SR.TXRDY==1, then write first data byte to transfer to THR.
4. Wait until SR.TXRDY==1, then write second data byte to transfer to THR.

5. Wait until SR.TXRDY==1, then write third data byte to transfer to THR.
6. Wait until SR.TXRDY==1, then write fourth data byte to transfer to THR.

27.8.7.2 *Read Followed by Read*

Consider the following transfer:

START, DADR+R, DATA+A, DATA+NA, REPSTART, DADR+R, DATA+A, DATA+NA, STOP.

To generate this transfer:

1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=1.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=1.
3. Wait until SR.RXRDY==1, then read first data byte received from RHR.
4. Wait until SR.RXRDY==1, then read second data byte received from RHR.
5. Wait until SR.RXRDY==1, then read third data byte received from RHR.
6. Wait until SR.RXRDY==1, then read fourth data byte received from RHR.

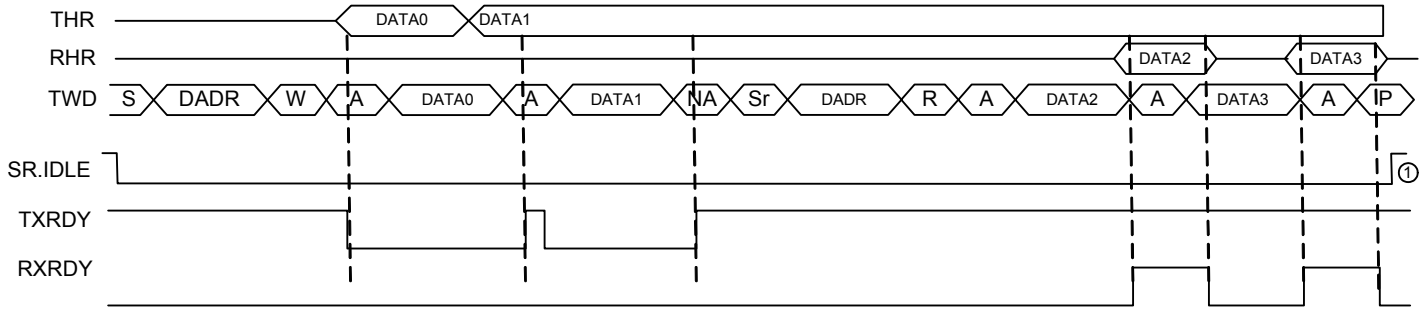
If combining several transfers, without any STOP or REPEATED START between them, remember to set the ACKLAST bit in CMDR to keep from ending each of the partial transfers with a NACK.

27.8.7.3 *Write Followed by Read*

Consider the following transfer:

START, DADR+W, DATA+A, DATA+A, REPSTART, DADR+R, DATA+A, DATA+NA, STOP.

Figure 27-12. Combining a Write and Read Transfer



To generate this transfer:

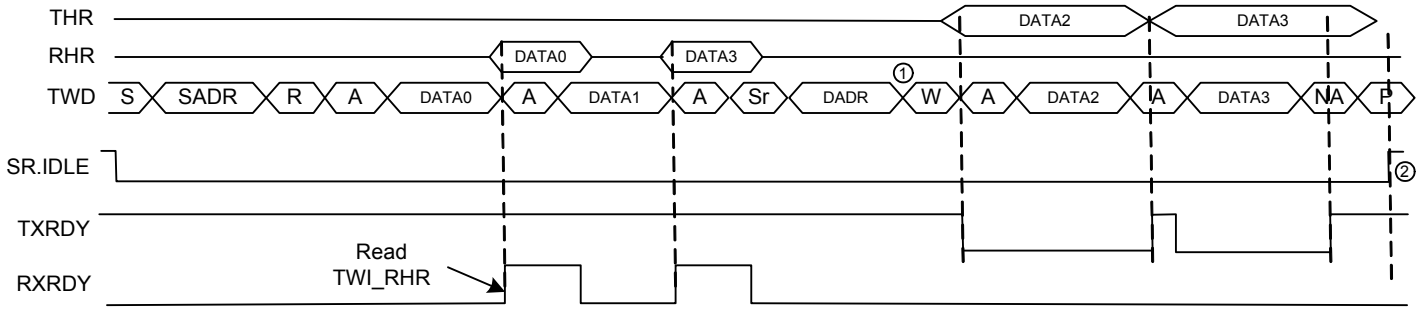
1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=0.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=1.
3. Wait until SR.TXRDY==1, then write first data byte to transfer to THR.
4. Wait until SR.TXRDY==1, then write second data byte to transfer to THR.
5. Wait until SR.RXRDY==1, then read first data byte received from RHR.
6. Wait until SR.RXRDY==1, then read second data byte received from RHR.

27.8.7.4 *Read Followed by Write*

Consider the following transfer:

START, DADR+R, DATA+A, DATA+NA, REPSTART, DADR+W, DATA+A, DATA+A, STOP.

Figure 27-13. Combining a Read and Write Transfer



To generate this transfer:

1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=1.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=0.
3. Wait until SR.RXRDY==1, then read first data byte received from RHR.
4. Wait until SR.RXRDY==1, then read second data byte received from RHR.
5. Wait until SR.TXRDY==1, then write first data byte to transfer to THR.
6. Wait until SR.TXRDY==1, then write second data byte to transfer to THR.

27.8.8 Ten Bit Addressing

Setting CMDR.TENBIT enables 10-bit addressing in hardware. Performing transfers with 10-bit addressing is similar to transfers with 7-bit addresses, except that bits 10:7 of CMDR.ADR must be set appropriately.

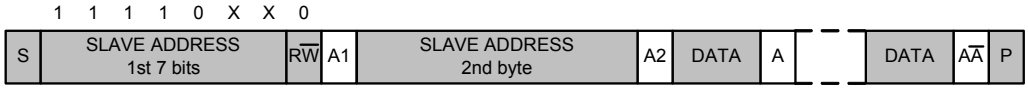
In Figure 27-14 on page 710 and Figure 27-15 on page 711, the grey boxes represent signals driven by the master, the white boxes are driven by the slave.

27.8.8.1 Master Transmitter

To perform a master transmitter transfer,

1. Program CMDR with TENBIT=1, REPSAME=0, READ=0, START=1, STOP=1 and the desired address and NBYTES value.

Figure 27-14. A Write Transfer with 10-bit Addressing



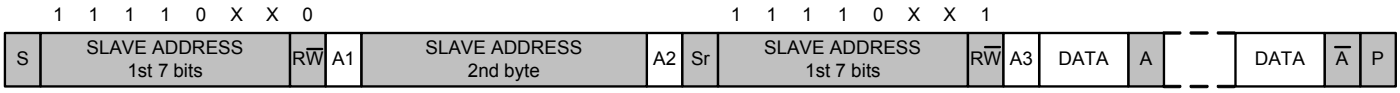
27.8.8.2 Master Receiver

When using master receiver mode with 10-bit addressing, CMDR.REPSAME must also be controlled. CMDR.REPSAME must be written to one when the address phase of the transfer should consist of only 1 address byte (the 11110xx byte) and not 2 address bytes. The I²C standard specifies that such addressing is required when addressing a slave for reads using 10-bit addressing.

To perform a master receiver transfer,

1. Program CMDR with TENBIT=1, REPSAME=0, READ=0, START=1, STOP=0, NBYTES=0 and the desired address.
2. Program NCMDR with TENBIT=1, REPSAME=1, READ=1, START=1, STOP=1 and the desired address and NBYTES value.

Figure 27-15. A Read Transfer with 10-bit Addressing



27.8.9 SMBus Mode

SMBus mode is enabled and disabled by the SMEN and SMDIS bits in CR. SMBus mode operation is similar to I²C operation with the following exceptions:

- Only 7-bit addressing can be used.
- The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into SMBTR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A dedicated bus line, SMBALERT, allows a slave to get a master's attention.
- A set of addresses have been reserved for protocol handling, such as Alert Response Address (ARA) and Host Header (HH) Address.

27.8.9.1 Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing CMDR.PECEN to one enables automatic PEC handling in the current transfer. Transfers with and without PEC can freely be intermixed in the same system, since some slaves may not support PEC. The PEC LFSR is always updated on every bit transmitted or received, so that PEC handling on combined transfers will be correct.

In master transmitter mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave will compare it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave will return an ACK to the master. If the PEC values differ, data was corrupted, and the slave will return a NACK value. The DNAK bit in SR reflects the state of the last received ACK/NACK value. Some slaves may not be able to check the received PEC in time to return a NACK if an error occurred. In this case, the slave should always return an ACK after the PEC byte, and some other mechanism must be implemented to verify that the transmission was received correctly.

In master receiver mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master will compare it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the PECERR bit in SR is set. In master receiver mode, the PEC byte is always followed by a NACK transmitted by the master, since it is the last byte in the transfer.

The PEC byte is automatically inserted in a master transmitter transmission if PEC is enabled when NBYTES reaches zero. The PEC byte is identified in a master receiver transmission if PEC is enabled when NBYTES reaches zero. NBYTES must therefore be set to the total number of data bytes in the transmission, including the PEC byte.

In combined transfers, the PECEN bit should only be set in the last of the combined transfers. Consider the following transfer:

S, ADR+W, COMMAND_BYTE, ACK, SR, ADR+R, DATA_BYTE, ACK, PEC_BYTE, NACK, P

This transfer is generated by writing two commands to the command registers. The first command is a write with NBYTES=1 and PECEN=0, and the second is a read with NBYTES=2 and PECEN=1.

Writing a one to the STOP bit in CR will place a STOP condition on the bus after the current byte. No PEC byte will be sent in this case.

27.8.9.2 Timeouts

The TLOWS and TLOWM fields in SMBTR configure the SMBus timeout values. If a timeout occurs, the master will transmit a STOP condition and leave the bus. The SR.TOUT bit is also set.

27.8.9.3 SMBus ALERT Signal

A slave can get the master's attention by pulling the TWALM line low. SR.SMBAL will then be set. This can be set up to trigger an interrupt, and software can then take the appropriate action, as defined in the SMBus standard.

27.8.10 Identifying Bus Events

This chapter lists the different bus events, and how these affects bits in the TWIM registers. This is intended to help writing drivers for the TWIM.

Table 27-5. Bus Events

Event	Effect
Master transmitter has sent a data byte	SR.THR is cleared.
Master receiver has received a data byte	SR.RHR is set.
Start+Sadr sent, no ack received from slave	SR.ANAK is set. SR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
Data byte sent to slave, no ack received from slave	SR.DNAK is set. SR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
Arbitration lost	SR.ARBLST is set. SR.CCOMP not set. CMDR.VALID remains set. TWCK and TWD immediately released to a pulled-up state.
SMBus Alert received	SR.SMBAL is set.
SMBus timeout received	SR.SMBTOUT is set. SR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.

Table 27-5. Bus Events

Event	Effect
Master transmitter receives SMBus PEC Error	SR.DNAK is set. SR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
Master receiver discovers SMBus PEC Error	SR.PECERR is set. SR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
CR.STOP is written by user	SR.STOP is set. SR.CCOMP set. CMDR.VALID remains set. STOP transmitted on bus after current byte transfer has finished.

27.9 User Interface

Table 27-6. TWIM Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control	CR	Write-only	0x00000000
0x04	Clock Waveform Generator	CWGR	Read/Write	0x00000000
0x08	SMBus Timing	SMBTR	Read/Write	0x00000000
0x0C	Command	CMDR	Read/Write	0x00000000
0x10	Next Command	NCMDR	Read/Write	0x00000000
0x14	Receive Holding	RHR	Read-only	0x00000000
0x18	Transmit Holding	THR	Write-only	0x00000000
0x1C	Status	SR	Read-only	0x00000002
0x20	Interrupt Enable Register	IER	Write-only	0x00000000
0x24	Interrupt Disable Register	IDR	Write-only	0x00000000
0x28	Interrupt Mask Register	IMR	Read-only	0x00000000
0x2C	Status Clear Register	SCR	Write-only	0x00000000
0x30	Parameter Register	PR	Read-only	(1)
0x34	Version Register	VR	Read-only	(1)

Note: 1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

27.9.1 Control Register (CR)

Name: CR

Access Type: Write-only

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	STOP

7	6	5	4	3	2	1	0
SWRST	-	SMDIS	SMEN	-	-	MDIS	MEN

- **STOP: Stop the current transfer**
  Writing a one to this bit terminates the current transfer, sending a STOP condition after the shifter has become idle. If there are additional pending transfers, they will have to be explicitly restarted by software after the STOP condition has been successfully sent.
  Writing a zero to this bit has no effect.
- **SWRST: Software Reset**
  If the TWIM master interface is enabled, writing a one to this bit resets the TWIM. All transfers are halted immediately, possibly violating the bus semantics.
  If the TWIM master interface is not enabled, it must first be enabled before writing a one to this bit.
  Writing a zero to this bit has no effect.
- **SMDIS: SMBus Disable**
  Writing a one to this bit disables SMBus mode.
  Writing a zero to this bit has no effect.
- **SMEN: SMBus Enable**
  Writing a one to this bit enables SMBus mode.
  Writing a zero to this bit has no effect.
- **MDIS: Master Disable**
  Writing a one to this bit disables the master interface.
  Writing a zero to this bit has no effect.
- **MEN: Master enable**
  Writing a one to this bit enables the master interface.
  Writing a zero to this bit has no effect.

27.9.2 Clock Waveform Generator Register (CWGR)

Name: CWGR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	EXP			DATA			

23	22	21	20	19	18	17	16
STASTO							

15	14	13	12	11	10	9	8
HIGH							

7	6	5	4	3	2	1	0
LOW							

- **EXP: Clock Prescaler**
  Used to specify how to prescale the TWCK clock. Counters are prescaled according to the following formula

$$f_{prescaled} = \frac{f_{clkpb}}{2^{(EXP+1)}}$$

- **DATA: Data Setup and Hold Cycles**
  Clock cycles for data setup and hold count. Prescaled by CWGR.EXP. Used to time T_{HD_DAT}, T_{SU_DAT}.
- **STASTO: START and STOP Cycles**
  Clock cycles in clock high count. Prescaled by CWGR.EXP. Used to time T_{HD_STA}, T_{SU_STA}, T_{SU_STO}
- **HIGH: Clock High Cycles**
  Clock cycles in clock high count. Prescaled by CWGR.EXP. Used to time T_{HIGH}.
- **LOW: Clock Low Cycles**
  Clock cycles in clock low count. Prescaled by CWGR.EXP. Used to time T_{LOW}, T_{BUF}.

27.9.3 SMBus Timing Register (SMBTR)

Name: SMBTR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
EXP				-	-	-	-

23	22	21	20	19	18	17	16
THMAX							

15	14	13	12	11	10	9	8
TLOWM							

7	6	5	4	3	2	1	0
TLOWS							

- **EXP: SMBus Timeout Clock prescaler**
  Used to specify how to prescale the TIM and TLOWM counters in SMBTR. Counters are prescaled according to the following formula

$$f_{prescaled,SMBus} = \frac{f_{clkpb}}{2^{(EXP+1)}}$$

- **THMAX: Clock High maximum cycles**
  Clock cycles in clock high maximum count. Prescaled by SMBTR.EXP. Used for bus free detection. Used to time $T_{HIGH:MAX}$.
  NOTE: Uses the prescaler specified by CWGR, NOT the prescaler specified by SMBTR.
- **TLOWM: Master Clock stretch maximum cycles**
  Clock cycles in master maximum clock stretch count. Prescaled by SMBTR.EXP. Used to time $T_{LOW:MEXT}$
- **TLOWS: Slave Clock stretch maximum cycles**
  Clock cycles in slave maximum clock stretch count. Prescaled by SMBTR.EXP. Used to time $T_{LOW:SEXT}$.

27.9.4 Command Register (CMDR)

Name: CMDR

Access Type: Read/Write

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	--			-	--	ACKLAST	PECEN

23	22	21	20	19	18	17	16
NBYTES							

15	14	13	12	11	10	9	8
VALID	STOP	START	REPSAME	TENBIT	SADR[9:7]		

7	6	5	4	3	2	1	0
SADR[6:0]							READ

- **ACKLAST: ACK Last Master RX Byte**
  Writing this bit to zero causes the last byte in master receive mode (when NBYTES has reached 0) to be NACKed. This is the standard way of ending a master receiver transfer.
  Writing this bit to one causes the last byte in master receive mode (when NBYTES has reached 0) to be ACKed. Used for performing linked transfers in master receiver mode with no STOP or REPEATED START between the subtransfers. This is needed when more than 255 bytes are to be received in one single transmission.
- **PECEN: Packet Error Checking Enable**
  Writing this bit to zero causes the transfer not to use PEC byte verification. The PEC LFSR is still updated for every bit transmitted or received. Must be used if SMBus mode is disabled.
  Writing this bit to one causes the transfer to use PEC. PEC byte generation (if master transmitter) or PEC byte verification (if master receiver) will be performed.
- **NBYTES: Number of data bytes in transfer**
  The number of data bytes in the transfer. After the specified number of bytes have been transferred, a STOP condition is transmitted if CMDR.STOP is set. In SMBus mode, if PEC is used, NBYTES includes the PEC byte, ie there are NBYTES-1 data bytes and a PEC byte.
- **VALID: CMDR Valid**
  Writing this to zero indicates that CMDR does not contain a valid command.
  Writing this to one indicates that CMDR contains a valid command. This bit is cleared when the command is finished.
- **STOP: Send STOP condition**
  Write this bit to zero to not transmit a STOP condition after the data bytes have been transmitted.
  Write this bit to one to transmit a STOP condition after the data bytes have been transmitted.
- **START: Send START condition**
  Write this bit to zero if the transfer in CMDR should not commence with a START or REPEATED START condition.
  Write this bit to one if the transfer in CMDR should commence with a START or REPEATED START condition. If the bus is free when the command is executed, a START condition is used, if the bus is busy, a REPEATED START is used.
- **REPSAME: Transfer is to same address as previous address**
  Only used in 10-bit addressing mode, always write to 0 in 7-bit addressing mode.

Write this bit to one if the command in CMDR performs a repeated start to the same slave address as addressed in the previous transfer in order to enter master receiver mode.

Write this bit to zero otherwise.

- **TENBIT: Ten Bit Addressing Mode**

  Write this bit to zero to use 7-bit addressing mode.

  Write this bit to one to use 10-bit addressing mode. Must not be used when TWIM is in SMBus mode.

- **SADR: Slave Address**

  Address of the slave involved in the transfer. Bits 9-7 are don't care if 7-bit addressing is used.

- **READ: Transfer Direction**

  Write this bit to zero to let the master transmit data.

  Write this bit to one to let the master receive data.

27.9.5 Next Command Register (NCMDR)

Name: NCMDR

Access Type: Read/Write

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	---			-	---	ACKLAST	PECEN

23	22	21	20	19	18	17	16
NBYTES							

15	14	13	12	11	10	9	8
VALID	STOP	START	REPSAME	TENBIT	SADR[9:7]		

7	6	5	4	3	2	1	0
SADR[6:0]							READ

This register is identical to CMDR. When the VALID bit in CMDR becomes 0, the contents of NCMDR is copied into CMDR, clearing the VALID bit in NCMDR. If the VALID bit in CMDR is cleared when NCMDR is written, the contents are copied immediately.

27.9.6 Receive Holding Register (RHR)

Name: RHR

Access Type: Read-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: Received Data**
  When the RXRDY bit in the Status Register (SR) is set, this field contains a byte received from the TWI bus.

27.9.7 Transmit Holding Register (THR)

Name: THR

Access Type: Write-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
TXDATA							

- **TXDATA: Data to Transmit**
  Write data to be transferred on the TWI bus here.

27.9.8 Status Register (SR)

Name: SR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000002

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	MENB

15	14	13	12	11	10	9	8
-	STOP	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	BUSFREE	IDLE	CCOMP	CRDY	TXRDY	RXRDY

- **MENB: Master Interface Enable**
  0: Master interface is disabled.
  1: Master interface is enabled.
- **STOP: Stop Request Accepted**
  This bit is set when STOP request caused by setting CR STOP has been accepted, and transfer has stopped.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **PECERR: PEC Error**
  This bit is set when a SMBus PEC error occurred.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **TOUT: Timeout**
  This bit is set when a SMBus timeout occurred.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **SMBALERT: SMBus Alert**
  This bit is set when an SMBus Alert was received.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **ARBLST: Arbitration Lost**
  This bit is set when the actual state of the SDA line did not correspond to the data driven onto it, indicating a higher-priority transmission in progress by a different master.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **DNAK: NAK in Data Phase Received**
  This bit is set when no ACK was received form slave during data transmission.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **ANAK: NAK in Address Phase Received**
  This bit is set when no ACK was received from slave during address phase
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **BUSFREE: Two-wire Bus is Free**
  This bit is set when activity has completed on the two-wire bus.

Otherwise, this bit is cleared.

- **IDLE: Master Interface is Idle**
  This bit is set when no command is in progress, and no command waiting to be issued.
  Otherwise, this bit is cleared.
- **CCOMP: Command Complete**
  This bit is set when the current command has completed successfully.
  Not set if the command failed due to conditions such as a NAK receved from slave.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **CRDY: Ready for More Commands**
  This bit is set when CMDR and/or NCMDR is ready to receive one or more commands.
  This bit is cleared when this is no longer true.
- **TXRDY: THR Data Ready**
  This bit is set when THR is ready for one or more data bytes.
  This bit is cleared when this is no longer true (i.e. THR is full or transmission has stopped).
- **RXRDY: RHR Data Ready**
  This bit is set when RX data are ready to be read from RHR.
  This bit is cleared when this is no longer true.

27.9.9 Interrupt Enable Register (IER)

Name: IER

Access Type: Write-only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	STOP	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	BUSFREE	IDLE	CCOMP	CRDY	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR

27.9.10 Interrupt Disable Register (IDR)

Name: IDR

Access Type: Write-only

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	--	-

15	14	13	12	11	10	9	8
-	STOP	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	BUSFREE	IDLE	CCOMP	CRDY	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR

27.9.11 Interrupt Mask Register (IMR)

Name: IMR

Access Type: Read-only

Offset: 0x28

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	--	-

15	14	13	12	11	10	9	8
-	STOP	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	BUSFREE	IDLE	CCOMP	CRDY	TXRDY	RXRDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
This bit is cleared when the corresponding bit in IDR is written to one.
This bit is set when the corresponding bit in IER is written to one.

27.9.12 Status Clear Register (SCR)

Name: SCR

Access Type : Write-only

Offset: 0x2C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	--	-

15	14	13	12	11	10	9	8
-	STOP	PECERR	TOUT	SMBALERT	ARBLST	DNAK	ANAK

7	6	5	4	3	2	1	0
-	-	-	-	CCOMP	-	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

27.9.13 Parameter Register (PR)

Name: PR

Access Type: Read-only

Offset: 0x30

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

•

27.9.14 Version Register (VR)

Name: VR

Access Type: Read-only

Offset: 0x34

Reset Value: Device-specific

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION [11:8]			

7	6	5	4	3	2	1	0
VERSION [7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

•

•

27.10 Module Configuration

The specific configuration for each TWIM instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 27-7. Module Configuration

Feature	TWIM0	TWIM1	TWIM2
SMBus ALERT interface	Implemented	Implemented	Not Implemented

Table 27-8. Module Clock Name

Module Name	Clock Name	Description
TWIM0	CLK_TWIM0	Peripheral Bus clock from the PBA clock domain
TWIM1	CLK_TWIM1	Peripheral Bus clock from the PBA clock domain
TWIM2	CLK_TWIM2	Peripheral Bus clock from the PBC clock domain

Table 27-9. Register Reset Values

Register	Reset Value
VR	0x0000 0101
PR	0x0000 0000

28. Two-Wire Slave Interface (TWIS)

Rev: 1.2.0.1

28.1 Features

- **Compatible with I²C standard**
  - **100 and 400 kbit/sin transfer speeds**
  - **7 and 10-bit and General Call addressing**
- **Compatible with SMBus standard**
  - **Hardware Packet Error Checking (CRC) generation and verification with ACK response**
  - **SMBALERT interface**
  - **25 ms clock low timeout delay**
  - **25 ms slave cumulative clock low extend time**
- **Compatible with PMBus**
- **DMA interface for reducing CPU load**
- **Arbitrary transfer lengths, including 0 data bytes**
- **Optional clock stretching if transmit or receive buffers not ready for data transfer**
- **32-bit Peripheral Bus interface for configuration of the interface**

28.2 Overview

The Atmel Two-wire Interface Slave (TWIS) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 kbit/s, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus I²C or SMBus compatible master. TWIS is always a bus slave and can transfer sequential or single bytes.

Below, Table 28-1 on page 732 lists the compatibility level of the Atmel Two-wire Slave Interface and a full I²C compatible device.

Table 28-1. Atmel TWIS Compatibility with I²C Standard

I²C Standard	Atmel TWIS
Standard-mode (100 kbit/s)	Supported
Fast-mode (400 kbit/s)	Supported
7 or 10 bits Slave Addressing	Supported
START BYTE(1)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NAK Management	Supported
Slope control and input filtering (Fast mode)	Supported
Clock stretching	Supported

Note: 1. START + b000000001 + Ack + Sr

Below, Table 28-2 on page 733 lists the compatibility level of the Atmel Two-wire Slave Interface and a full SMBus compatible device.

Table 28-2. Atmel TWIS Compatibility with SMBus Standard

SMBus Standard	Atmel TWIS
Bus Timeouts	Supported
Address Resolution Protocol	Supported
Alert	Supported
Packet Error Checking	Supported

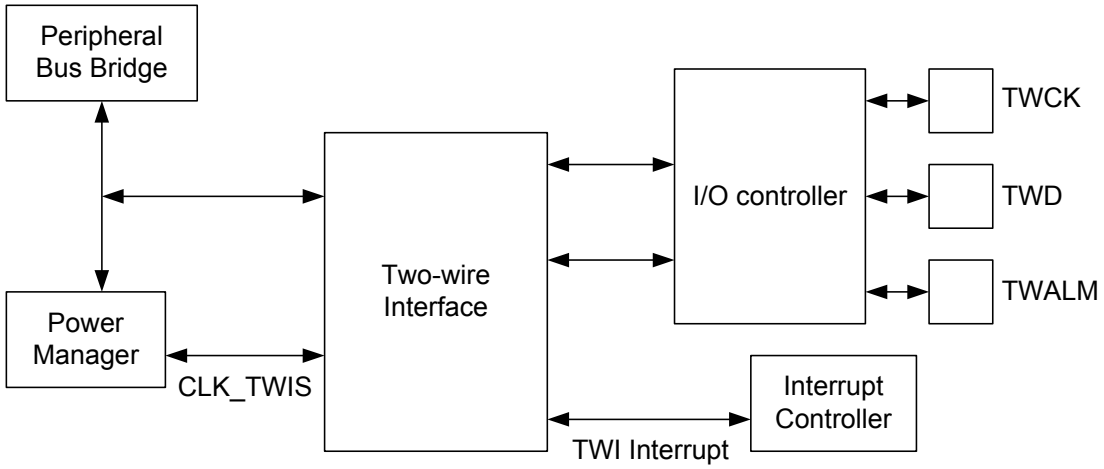
28.3 List of Abbreviations

Table 28-3. Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

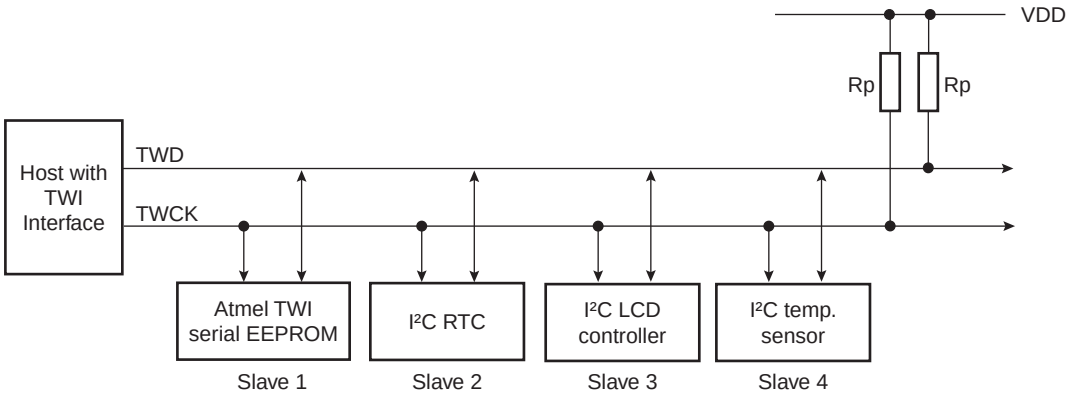
28.4 Block Diagram

Figure 28-1. Block Diagram



28.5 Application Block Diagram

Figure 28-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

28.6 I/O Lines Description

Table 28-4. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output
TWALM	SMBus SMBALERT	Input/Output

28.7 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

28.7.1 I/O Lines

TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 28-5 on page 736). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWALM is used to implement the optional SMBus SMBALERT signal.

TWALM, TWD, and TWCK pins may be multiplexed with I/O Controller lines. To enable the TWIS, the programmer must perform the following steps:

- Program the I/O Controller to:
  - Dedicate TWD, TWCK and optionally TWALM as peripheral lines.
  - Define TWD, TWCK and optionally TWALM as open-drain.

28.7.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the TWIS, the TWIS will stop functioning and resume operation after the system wakes up from sleep mode. TWIS is able to wake the system from sleep mode upon address match, see Section 28.8.8 on page 743.

28.7.3 Clocks

The clock for the TWIS bus interface (CLK_TWIS) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the TWIS before disabling the clock, to avoid freezing the TWIS in an undefined state.

28.7.4 DMA

The TWIS DMA handshake interface is connected to the Peripheral DMA Controller. Using the TWIS DMA functionality requires the Peripheral DMA Controller to be programmed after setting up the TWIS.

28.7.5 Interrupts

The TWIS interrupt request lines are connected to the interrupt controller. Using the TWIS interrupts requires the interrupt controller to be programmed first.

28.7.6 Debug Operation

When an external debugger forces the CPU into debug mode, the TWIS continues normal operation. If the TWIS is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

28.8 Functional Description

28.8.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 28-4 on page 736).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 28-3 on page 735).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

Figure 28-3. START and STOP Conditions

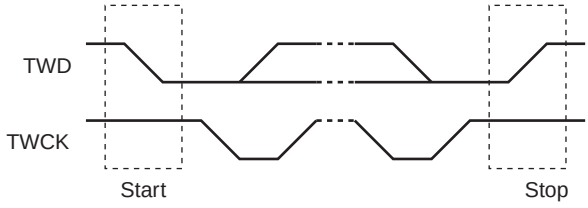
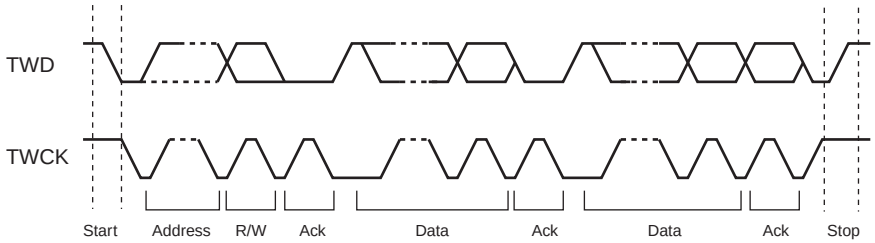


Figure 28-4. Transfer Format



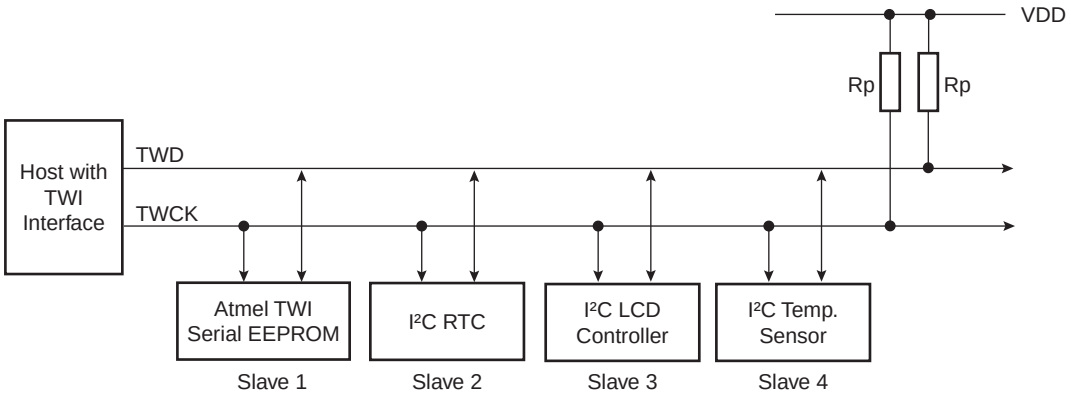
28.8.2 Operation

TWIS has two modes of operation:

- Slave transmitter mode
- Slave receiver mode

A master is a device which starts and stops a transfer and generates the TWCK clock. A slave is assigned an address and responds to requests from the master. These modes are described in the following chapters.

Figure 28-5. Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

28.8.2.1 Bus Timing

The Timing Register (TR) is used to control the timing of bus signals driven by TWIS. TR describes bus timings as a function of cycles of the prescaled CLK_TWIS. The clock prescaling can be selected through TR.EXP.

$$f_{prescaled} = \frac{f_{CLK-TWIS}}{2^{(EXP+1))}}$$

TR has the following fields:

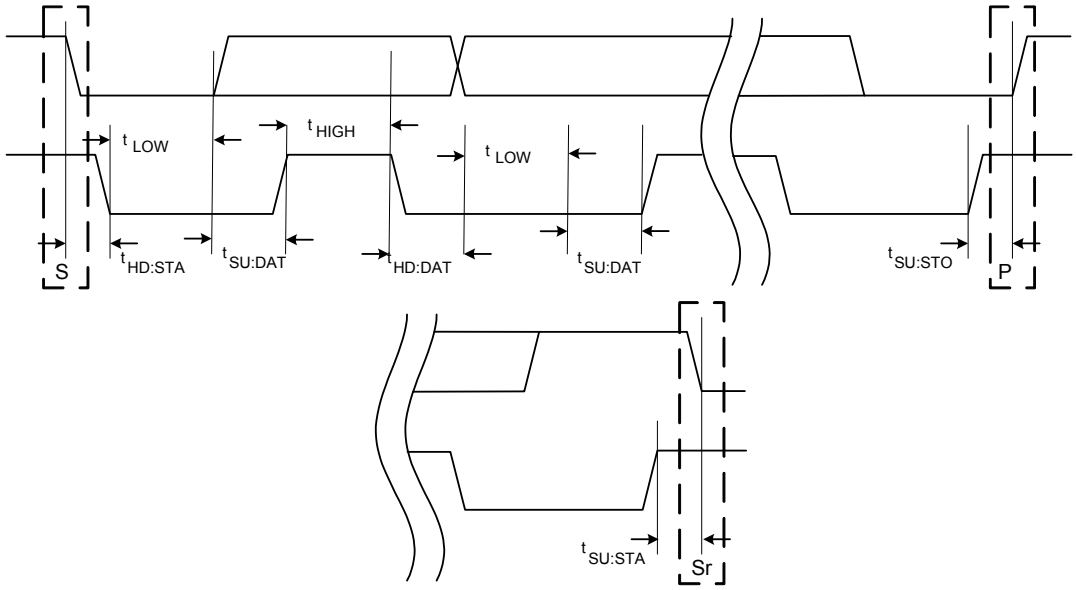
TLOWS: Prescaled clock cycles used to time SMBUS timeout $T_{LOW:SEXT}$.

TTOUT: Prescaled clock cycles used to time SMBUS timeout $T_{TIMEOUT}$.

SUDAT: Non-prescaled clock cycles for data setup and hold count. Used to time T_{SU_DAT}.

EXP: Specifies the clock prescaler setting used for the SMBUS timeouts.

Figure 28-6. Bus Timing Diagram



28.8.2.2 Setting Up and Performing a Transfer

Operation of TWIS is mainly controlled by the Control Register (CR). The following list presents the main steps in a typical communication:

1. Before any transfers can be performed, bus timings must be configured by programming the Timing Register (TR).
2. If a DMA controller is to be used for the transfers, it must be set up.
3. The Control Register (CR) must be configured with information such as the slave address, SMBus mode, Packet Error Checking (PEC), number of bytes to transfer, and which addresses to match.

The interrupt system can be set up to give interrupt request on specific events or error conditions, for example when a byte has been received.

The NBYTES register is only used in SMBus mode, when PEC is enabled. In I²C mode or in SMBus mode when PEC is disabled, the NBYTES register is not used, and should be written to 0. NBYTES is updated by hardware, so in order to avoid hazards, software updates of NBYTES can only be done through writes to the NBYTES register.

28.8.2.3 Address Matching

TWIS can be set up to match several different addresses. More than one address match may be enabled simultaneously, allowing TWIS to be assigned to several addresses. The address matching phase is initiated after a START or REPEATED START condition. When TWIS receives an address that generates an address match, an ACK is automatically returned to the master.

In I²C mode:

- The address in CR.ADR is checked for address match if CR.SMATCH is set.
- The General Call address is checked for address match if CR.GCMATCH is set.

In SMBus mode:

- The address in CR.ADR is checked for address match if CR.SMATCH is set.
- The Alert Response Address is checked for address match if CR.SMAL is set.
- The Default Address is checked for address match if CR.SMDA is set.
- The Host Header Address is checked for address match if CR.SMHH is set.

28.8.2.4 Clock Stretching

Any slave or bus master taking part in a transfer may extend the TWCK low period at any time. TWIS may extend the TWCK low period after each byte transfer if CR.STREN=1 and:

- Module is in slave transmitter mode, data should be transmitted, but THR is empty, or
- Module is in slave receiver mode, a byte has been received and placed into the internal shifter, but RHR is full, or
- Stretch-on-address-match bit CR.SOAM=1 and slave was addressed. Bus clock remains stretched until all address match bits in SR have been cleared.

If CR.STREN=0 and:

- Module is in slave transmitter mode, data should be transmitted but THR is empty: Transmit the value present in THR (the last transmitted byte or reset value), and set SR.URUN.
- Module is in slave receiver mode, a byte has been received and placed into the internal shifter, but RHR is full: Discard the received byte and set SR.ORUN.

28.8.2.5 Bus Errors

If a bus error (misplaced START or STOP) condition is detected, the SR.BUSERR bit is set and TWIS waits for a new START condition.

28.8.3 Slave Transmitter Mode

If TWIS matches an address in which the R/$\overline{W}$ bit in the TWI address phase transfer is set, it will enter slave transmitter mode and set the SR.TRA bit (note that SR.TRA is set one CLK_TWIS cycle after the relevant address match bit in the same register is set).

After the address phase, the following actions are performed:

1. If SMBus mode and PEC is used, NBYTES must be set up with the number of bytes to transmit. This is necessary in order to know when to transmit PEC byte. NBYTES can also be used to count the number of bytes received if using DMA.
2. Byte to transmit depends on I²C/SMBus mode and CR.PEC:
   – If in I²C mode or CR.PEC=0 or NBYTES!=0: TWIS waits until THR contains a valid data byte, possibly stretching the low period of TWCK. After THR contains a valid data byte, the data byte is transferred to a shifter, and then SR.TXRDY is changed to one because the THR is empty again.
   – SMBus mode and CR.PEC=1: If NBYTES=0, the generated PEC byte is automatically transmitted instead of a data byte from THR. TWCK will not be stretched by TWIS.
3. The data byte in the shifter is transmitted.
4. NBYTES is updated. If CR.CUP is set, NBYTES is incremented, otherwise NBYTES is decremented.
5. After each data byte has been transmitted, the master transmits an ACK (Acknowledge) or NAK (Not Acknowledge) bit. If a NAK bit is received by TWIS, the SR.NAK bit is set. Note that this is done two CLK_TWIS cycles after TWCK has been sampled by TWIS to be HIGH (see Figure 28-9 on page 740). The NAK indicates that the transfer is finished, and TWIS will wait for a STOP or REPEATED START. If an ACK bit is received, the SR.NAK bit remains LOW. The ACK indicates that more data should be transmitted, so jump to step 2. At the end of the ACK/NAK clock cycle, the SR.BTF (Byte Transfer Finished) bit is set. Note that this is done two CLK_TWIS cycles after TWCK has been sampled by TWIS to be LOW (see Figure 28-9 on page 740). Also note that in the event that SR.NAK bit is set, it must not be cleared before the SR.BTF bit is set to ensure correct TWIS behavior.
6. If STOP is received, SR.TCOMP and SR.STO will be set.
7. If REPEATED START is received, SR.REP will be set.

The TWI transfers require the receiver to acknowledge each received data byte. During the acknowledge clock pulse (9th pulse), the slave releases the data line (HIGH), enabling the master to pull it down in order to generate the acknowledge. The slave polls the data line during this clock pulse and sets the NAK bit in the Status Register if the master does not acknowledge the data byte. A NAK means that the master does not wish to receive additional data bytes. As with the other status bits, an interrupt can be generated if enabled in the Interrupt Enable Register (IER).

TXRDY is used as Transmit Ready for the Peripheral DMA Controller transmit channel.

The end of the complete transfer is marked by the SR.TCOMP bit set to one. See Figure 28-7 on page 740 and Figure 28-8 on page 740.

Figure 28-7. Slave Transmitter with One Data Byte

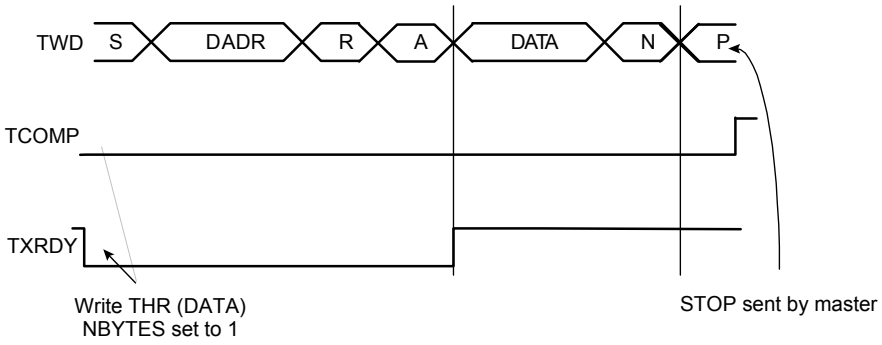


Figure 28-8. Slave Transmitter with Multiple Data Bytes

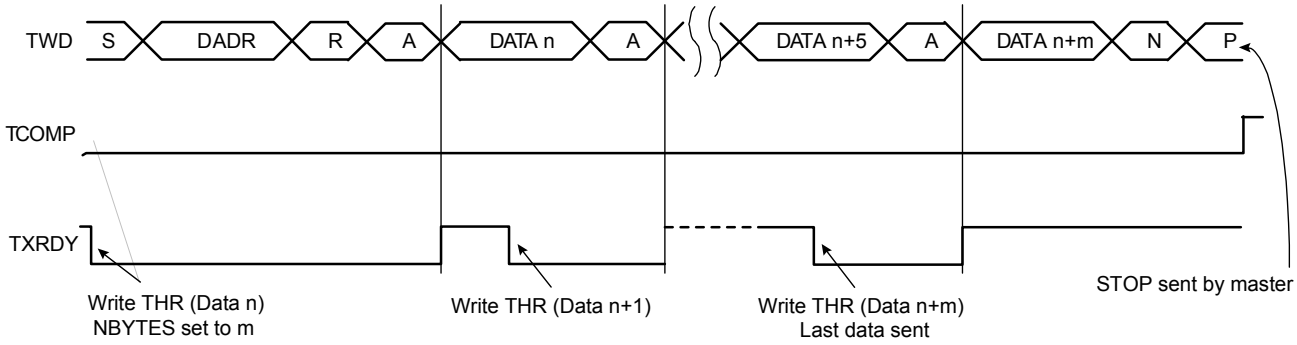
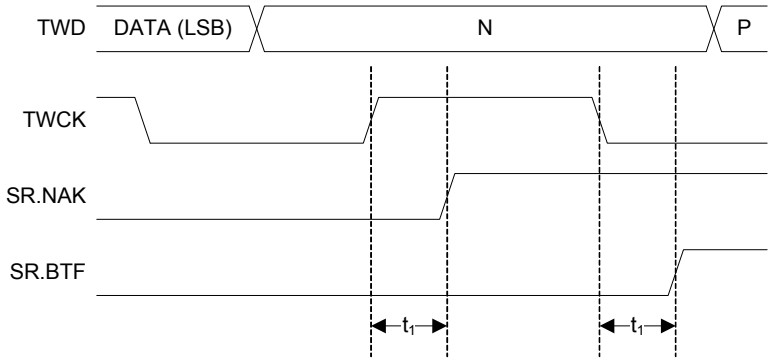


Figure 28-9. Timing Relationship between TWCK, SR.NAK, and SR.BTF



t_1: (CLK_TWIS period) x 2

28.8.4 Slave Receiver Mode

If TWIS matches an address in which the R/$\overline{W}$ bit in the TWI address phase transfer is cleared, it will enter slave receiver mode and clear SR.TRA (note that SR.TRA is cleared one CLK_TWIS cycle after the relevant address match bit in the same register is set).

After the address phase, the following is repeated:

1. If SMBus mode and PEC is used, NBYTES must be set up with the number of bytes to receive. This is necessary in order to know which of the received bytes is the PEC byte. NBYTES can also be used to count the number of bytes received if using DMA.
2. Receive a byte. Set SR.BTF when done.

3. Update NBYTES. If CR.CUP is written to one, NBYTES is incremented, otherwise NBYTES is decremented. NBYTES is usually configured to count downwards if PEC is used.
4. After a data byte has been received, the slave transmits an ACK or NAK bit. For ordinary data bytes, the CR.ACK field controls if an ACK or NAK should be returned. If PEC is enabled and the last byte received was a PEC byte (indicated by NBYTES=0), TWIS will automatically return an ACK if the PEC value was correct, otherwise a NAK will be returned.
5. If STOP is received, SR.TCOMP will be set.
6. If REPEATED START is received, SR.REP will be set.

The TWI transfers require the receiver to acknowledge each received data byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse.

RXRDY is used as Receive Ready for the Peripheral DMA Controller receive channel.

Figure 28-10. Slave Receiver with One Data Byte

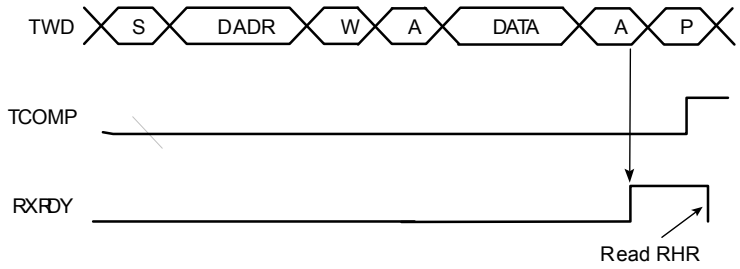
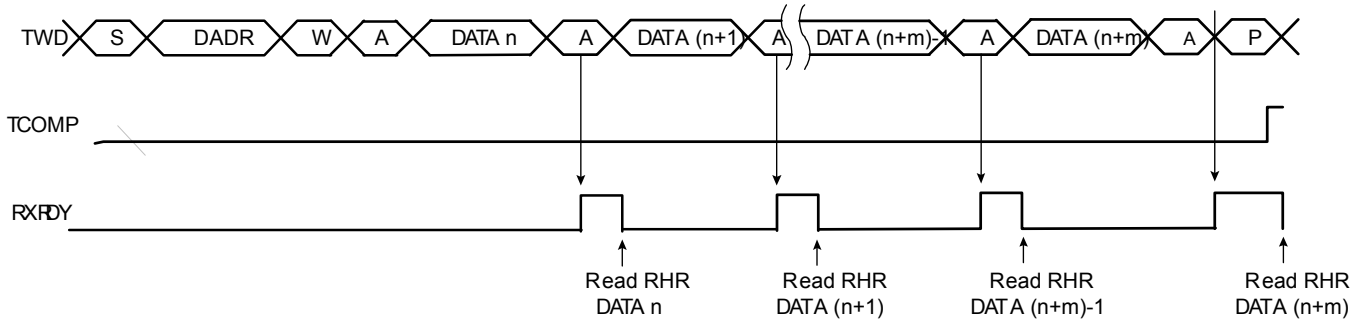


Figure 28-11. Slave Receiver with Multiple Data Bytes



28.8.5 Interactive ACKing Received Data Bytes

When implementing a register interface over TWI, it may sometimes be necessary or just useful to report reads and writes to invalid register addresses by sending a NAK to the host. To be able to do this, one must first receive the register address from the TWI bus, and then tell the TWIS

whether to ACK or NAK it. In normal operation of the TWIS, this is not possible because the controller will automatically ACK the byte at about the same time as the RXRDY interrupt flag is set. Writing a one to the Stretch on Data Byte Received bit (CR[SODR]) will stretch the clock allowing the user to update CR[ACK] bit before returning the desired value. After the last bit in the data byte is received, the TWI bus clock is stretched, the received data byte is transferred to the RHR register, and SR[BTF] is set. At this time, the user can examine the received byte and write the desired ACK or NACK value to CR[ACK]. When the user clears SR[BTF], the desired ACK value is transferred on the TWI bus. This makes it possible to look at the byte received, determine if it's valid, and then decide to ACK or NAK it.

28.8.6 Using the Peripheral DMA Controller

The use of the Peripheral DMA Controller significantly reduces the CPU load. The programmer can set up ring buffers for the DMA controller, containing data to transmit or free buffer space to place received data. By initializing NBYTES to 0 before a transfer, and setting CR.CUP, NBYTES is incremented by 1 each time a data has been transmitted or received. This allows the programmer to detect how much data was actually transferred by the DMA system.

To assure correct behavior, respect the following programming sequences:

28.8.6.1 Data Transmit with the Peripheral DMA Controller

1. Initialize the transmit Peripheral DMA Controller (memory pointers, size, etc.).
2. Configure the TWIS (ADR, NBYTES, etc.).
3. Start the transfer by setting the Peripheral DMA Controller TXTEN bit.
4. Wait for the Peripheral DMA Controller end TX flag.
5. Disable the Peripheral DMA Controller by setting the Peripheral DMA Controller TXDIS bit.

28.8.6.2 Data Receive with the Peripheral DMA Controller

1. Initialize the receive Peripheral DMA Controller (memory pointers, size - 1, etc.).
2. Configure the TWIS (ADR, NBYTES, etc.).
3. Start the transfer by setting the Peripheral DMA Controller RXTEN bit.
4. Wait for the Peripheral DMA Controller end RX flag.
5. Disable the Peripheral DMA Controller by setting the Peripheral DMA Controller RXDIS bit.

28.8.7 SMBus Mode

SMBus mode is enabled when CR.SMEN is written to one. SMBus mode operation is similar to I²C operation with the following exceptions:

- Only 7-bit addressing can be used.
- The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into TR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A dedicated bus line, SMBALERT, allows a slave to get a master's attention.
- A set of addresses have been reserved for protocol handling, such as Alert Response Address (ARA) and Host Header (HH) Address. Address matching on these addresses can be enabled by configuring CR appropriately.

28.8.7.1 Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to CR.PECEN enables automatic PEC handling in the current transfer. The PEC generator is always updated on every bit transmitted or received, so that PEC handling on following linked transfers will be correct.

In slave receiver mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave will compare it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave will return an ACK to the master. If the PEC values differ, data was corrupted, and the slave will return a NAK value. The SR.SMBPECERR bit is set automatically if a PEC error occurred.

In slave transmitter mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master will compare it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the master must take appropriate action.

The PEC byte is automatically inserted in a slave transmitter transmission if PEC enabled when NBYTES reaches zero. The PEC byte is identified in a slave receiver transmission if PEC enabled when NBYTES reaches zero. NBYTES must therefore be set to the total number of data bytes in the transmission, including the PEC byte.

28.8.7.2 Timeouts

The Timing Register (TR) configures the SMBus timeout values. If a timeout occurs, the slave will leave the bus. The SR.SMBTOUT bit is also set.

28.8.7.3 SMBALERT

A slave can get the master's attention by pulling the SMBALERT line low. This is done by setting the CR.SMBAL bit. This will also enable address match on the Alert Response Address (ARA).

28.8.8 Wakeup from Sleep Modes by TWI Address Match

TWIS is able to wake the device up from a sleep mode upon an address match, including sleep modes where CLK_TWIS is stopped. The behavior of TWIS when a TWI Start condition is received during a sleep mode where CLK_TWIS is stopped depends on whether the auxiliary address matching feature is implemented (refer to the Module Configuration section at the end of this chapter) and enabled. If implemented, auxiliary address matching can be enabled by writing a '1' to the Auxiliary Address Matching (AUXAM) bit of the Control Register (CR).

If auxiliary address matching is not enabled, TWIS will stretch TWCK until CLK_TWIS has started. The time required for starting CLK_TWIS depends on which sleep mode the device is in. After CLK_TWIS has started, TWIS releases its TWCK stretching and receives one byte of data on the bus. At this time, only a limited part of the device, including TWIS, receives a clock, thus saving power. If the received byte is a master code, TWIS enters HS-mode (provided that HS-mode is implemented; refer to the Module Configuration section at the end of this chapter) and goes on to receive the slave address. If the address phase (either in F/S- or HS-mode) causes a TWIS address match, the entire device is wakened and normal TWIS address matching actions are performed. Normal TWI transfer then follows. If TWIS is not addressed, CLK_TWIS is automatically stopped and the device returns to its original sleep mode. If TWIS is in HS-mode, it remains so until it detects a STOP condition on the bus, after which it switches back to F/S-mode.

On the other hand, if auxiliary address matching is enabled, TWIS will operate using TWCK directly as its clock without stretching TWCK and waiting for CLK_TWIS to start. If an address match occurs, the entire device is wakened and normal TWI transfer follows. The time required for the device to wake up depends on which sleep mode it is in.

28.8.9 Identifying Bus Events

This chapter lists the different bus events, and how these affects bits in the TWIS registers. This is intended to help writing drivers for the TWIS.

Table 28-5. Bus Events

Event	Effect
Slave transmitter has sent a data byte	SR.THR is cleared. SR.BTF is set. The value of the ACK bit sent immediately after the data byte is given by CR.ACK.
Slave receiver has received a data byte	SR.RHR is set. SR.BTF is set. SR.NAK updated according to value of ACK bit received from master.
Start+Sadr on bus, but address is to another slave	None.
Start+Sadr on bus, current slave is addressed, but address match enable bit in CR is not set	None.
Start+Sadr on bus, current slave is addressed, corresponding address match enable bit in CR set	Correct address match bit in SR is set. SR.TRA updated according to transfer direction (updating is done one CLK_TWIS cycle after address match bit is set) Slave enters appropriate transfer direction mode and data transfer can commence.
Start+Sadr on bus, current slave is addressed, corresponding address match enable bit in CR set, SR.STREN and SR.SOAM are set.	Correct address match bit in SR is set. SR.TRA updated according to transfer direction (updating is done one CLK_TWIS cycle after address match bit is set). Slave stretches TWCK immediately after transmitting the address ACK bit. TWCK remains stretched until all address match bits in SR have been cleared. Slave the enters appropriate transfer direction mode and data transfer can commence.
Repeated Start received after being addressed	SR.REP set. SR.TCOMP unchanged.
Stop received after being addressed	SR.STO set. SR.TCOMP set.
Start, Repeated Start or Stop received in illegal position on bus	SR.BUSERR set.
Data is to be received in slave receiver mode, SR.STREN is set, and RHR is full	TWCK is stretched until RHR has been read.

Table 28-5. Bus Events

Event	Effect
Data is to be transmitted in slave receiver mode, SR.STREN is set, and THR is empty	TWCK is stretched until THR has been written.
Data is to be received in slave receiver mode, SR.STREN is cleared, and RHR is full	TWCK is not stretched, read data is discarded. SR.ORUN is set.
Data is to be transmitted in slave receiver mode, SR.STREN is cleared, and THR is empty	TWCK is not stretched, previous contents of THR is written to bus. SR.URUN is set.
SMBus timeout received	SR.SMBTOUT is set. TWCK and TWD are immediately released.
Slave transmitter in SMBus PEC mode has transmitted a PEC byte, that was not identical to the PEC calculated by the master receiver.	Master receiver will transmit a NAK as usual after the last byte of a master receiver transfer. Master receiver will retry the transfer at a later time.
Slave receiver discovers SMBus PEC Error	SR.SMBPECERR is set. NAK returned after the data byte.

28.9 User Interface

Table 28-6. TWIS Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Read/Write	0x00000000
0x04	NBYTES Register	NBYTES	Read/Write	0x00000000
0x08	Timing Register	TR	Read/Write	0x00000000
0x0C	Receive Holding Register	RHR	Read-only	0x00000000
0x10	Transmit Holding Register	THR	Write-only	0x00000000
0x14	Packet Error Check Register	PECR	Read-only	0x00000000
0x18	Status Register	SR	Read-only	0x00000002
0x1c	Interrupt Enable Register	IER	Write-only	0x00000000
0x20	Interrupt Disable Register	IDR	Write-only	0x00000000
0x24	Interrupt Mask Register	IMR	Read-only	0x00000000
0x28	Status Clear Register	SCR	Write-only	0x00000000
0x2C	Parameter Register	PR	Read-only	(1)
0x30	Version Register	VR	Read-only	(1)

Note: 1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

28.9.1 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-AUXAM	-	TENBIT	ADR[9:8]	

23	22	21	20	19	18	17	16
ADR[7:0]							

15	14	13	12	11	10	9	8
SODR	SOAM	CUP	ACK	PECEN	SMHH	SMDA	SMBALERT

7	6	5	4	3	2	1	0
SWRST	-	-	STREN	GCMATCH	SMATCH	SMEN	SEN

- **AUXAM: Auxiliary Address Matching**
  Write this bit to zero to disable auxiliary address matching.
  Write this bit to one to enable auxiliary address matching.
- **TENBIT: Ten Bit Address Match**
  Write this bit to zero to disable Ten Bit Address Match.
  Write this bit to one to enable Ten Bit Address Match.
- **ADR: Slave Address**
  Slave address used in slave address match. Bits 9:0 are used if in 10-bit mode, bits 6:0 otherwise.
- **SODR: Stretch Clock on Data Byte Reception**
  Writing this bit to zero will not stretch bus clock immediately before ACKing a received data byte.
  Writing this bit to one will strech bus clock immediately before ACKing a received data byte.
- **SOAM: Stretch Clock on Address Match**
  Writing this bit to zero will not strech bus clock after address match.
  Writing this bit to one will strech bus clock after address match.
- **CUP: NBYTES Count Up**
  Writing this bit to zero causes NBYTES to count down (decrement) per byte transferred.
  Writing this bit to one causes NBYTES to count up (increment) per byte transferred.
- **ACK: Slave Receiver Data Phase ACK Value**
  Writing this bit to zero causes a low value to be returned in the ACK cycle of the data phase in slave receiver mode.
  Writing this bit to one causes a high value to be returned in the ACK cycle of the data phase in slave receiver mode.
- **PECEN: Packet Error Checking Enable**
  Writing this bit to zero disables SMBus PEC (CRC) generation and check.
  Writing this bit to one enables SMBus PEC (CRC) generation and check.
- **SMHH: SMBus Host Header**
  Writing this bit to zero causes TWIS not to acknowledge the SMBus Host Header.
  Writing this bit to one causes TWIS to acknowledge the SMBus Host Header.
- **SMDA: SMBus Default Address**
  Writing this bit to zero causes TWIS not to acknowledge the SMBus Default Address.
  Writing this bit to one causes TWIS to acknowledge the SMBus Default Address.

- **SMBALERT: SMBus Alert**
  Writing this bit to zero causes TWIS to release the SMBALERT line and not to acknowledge the SMBus Alert Response Address (ARA).
  Writing this bit to one causes TWIS to pull down the SMBALERT line and to acknowledge the SMBus Alert Response Address (ARA).
- **SWRST: Software Reset**
  This bit will always read as 0.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets the TWIS.
- **STREN: Clock Stretch Enable**
  Writing this bit to zero disables clock stretching if RHR/THR buffer full/empty. May cause over/underrun.
  Writing this bit to one enables clock stretching if RHR/THR buffer full/empty.
- **GCMATCH: General Call Address Match**
  Writing this bit to zero causes TWIS not to acknowledge the General Call Address.
  Writing this bit to one causes TWIS to acknowledge the General Call Address.
- **SMATCH: Slave Address Match**
  Writing this bit to zero causes TWIS not to acknowledge the Slave Address.
  Writing this bit to one causes TWIS to acknowledge the Slave Address.
- **SMEN: SMBus Mode Enable**
  Writing this bit to zero disables SMBus mode.
  Writing this bit to one enables SMBus mode.
- **SEN: Slave Enable**
  Writing this bit to zero disables the slave interface.
  Writing this bit to one enables the slave interface.

28.9.2 NBYTES Register

Name: NBYTES

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
-							

15	14	13	12	11	10	9	8
-							

7	6	5	4	3	2	1	0
NBYTES							

- **NBYTES: Number of Bytes to Transfer**
  Writing to this field updates the NBYTES counter. Can also be read to to learn the progress of the transfer. Can be incremented or decremented automatically by hardware.

28.9.3 Timing Register

Name: TR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
EXP				-			

23	22	21	20	19	18	17	16
SUDAT							

15	14	13	12	11	10	9	8
TTOUT							

7	6	5	4	3	2	1	0
TLOWS							

- **EXP: Clock Prescaler**
  Used to specify how to prescale the SMBus TLOWS counter. The counter is prescaled according to the following formula:

$$f_{prescaled} = \frac{f_{clkpb}}{2^{(EXP+1)}}$$

- **SUDAT: Data Setup Cycles**
  Non-prescaled clock cycles for data setup count. Used to time T_{SU_DAT}. Data is driven SUDAT cycles after TWCK low detected. This timing is used for timing the ACK/NAK bits, and any data bits driven in slave transmitter mode.
- **TTOUT: SMBus Ttimeout Cycles**
  Prescaled clock cycles used to time SMBus $T_{TIMEOUT}$.
- **TLOWS: SMBus Tlow:sext Cycles**
  Prescaled clock cycles used to time SMBus $T_{LOW:SEXT}$.

28.9.4 Receive Holding Register

Name: RHR

Access Type: Read-only

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: Received Data Byte**
  When the RXRDY bit in the Status Register (SR) is set, this field contains a byte received from the TWI bus.

28.9.5 Transmit Holding Register

Name: THR

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
TXDATA							

- **TXDATA: Data Byte to Transmit**
  Write data to be transferred on the TWI bus here.

28.9.6 Packet Error Check Register

Name: PECR

Access Type: Read-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
PEC							

- **PEC: Calculated PEC Value**

  The calculated PEC value. Updated automatically by hardware after each byte has been transferred. Reset by hardware after a STOP condition. Provided if the user manually wishes to control when the PEC byte is transmitted, or wishes to access the PEC value for other reasons. In ordinary operation, the PEC handling is done automatically by hardware.

28.9.7 Status Register

Name: SR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x000000002

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	TRA	-	TCOMP	SEN	TXRDY	RXRDY

- **BTF: Byte Transfer Finished**
  This bit is set when byte transfer has completed.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **REP: Repeated Start Received**
  This bit is set when REPEATED START condition received.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **STO: Stop Received**
  This bit is set when STOP condition received.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SMBDAM: SMBus Default Address Match**
  This bit is set when received address matched SMBus Default Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SMBHHM: SMBus Host Header Address Match**
  This bit is set when received address matched SMBus Host Header Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SMBALERTM: SMBus Alert Response Address Match**
  This bit is set when received address matched SMBus Alert Response Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **GCM: General Call Match**
  This bit is set when received address matched General Call Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SAM: Slave Address Match**
  This bit is set when received address matched Slave Address.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **BUSERR: Bus Error**
  This bit is set when a misplaced start or stop condition has occurred.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **SMBPECERR: SMBus PEC Error**
  This bit is set when SMBus PEC error has occurred.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SMBTOUT: SMBus Timeout**
  This bit is set when SMBus timeout has occurred.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **NAK: NAK Received**
  This bit is set when NAK was received from master during slave transmitter operation.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **ORUN: Overrun**
  This bit is set when overrun has occurred in slave receiver mode. Can only occur if CR.STREN=0.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **URUN: Underrun**
  This bit is set when underrun has occurred in slave transmitter mode. Can only occur if CR.STREN=0.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **TRA: Transmitter Mode**
  0: The slave is in slave receiver mode.
  1: The slave is in slave transmitter mode.
- **TCOMP: Transmission Complete**
  This bit is set when transmission is complete. Set after receiving a STOP after being addressed.
  This bit is cleared when the corresponding bit in SCR is written to one.
- **SEN: Slave Enabled**
  0: The slave interface is disabled.
  1: The slave interface is enabled.
- **TXRDY: TX Buffer Ready**
  0: The TX buffer is full and should not be written to.
  1: The TX buffer is empty, and can accept new data.
- **RXRDY: RX Buffer Ready**
  0: No RX data ready in RHR.
  1: RX data is ready to be read from RHR.

28.9.8 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	-	-	TCOMP	-	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

28.9.9 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	-	-	TCOMP	-	TXRDY	RXRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

28.9.10 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	-	-	TCOMP	-	TXRDY	RXRDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
This bit is cleared when the corresponding bit in IDR is written to one.
This bit is set when the corresponding bit in IER is written to one.

28.9.11 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x28

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	SMBALERTM	GCM	SAM

15	14	13	12	11	10	9	8
-	BUSERR	SMBPECERR	SMBTOUT	-	-	-	NAK

7	6	5	4	3	2	1	0
ORUN	URUN	-	-	TCOMP	-	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

28.9.12 Parameter Register

Name: PR

Access Type: Read-only

Offset: 0x2C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-AUXAM	-

- **AUXAM: Auxiliary Address Matching**
  0: Auxiliary address matching is not supported.
  1: Auxiliary address matching is supported.

28.9.13 Version Register (VR)

Name: VR

Access Type: Read-only

Offset: 0x30

Reset Value: Device-specific

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION [11:8]			

7	6	5	4	3	2	1	0
VERSION [7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

28.10 Module Configuration

The specific configuration for each TWIS instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 28-7. Module Configuration

Feature	TWIM0	TWIM1	TWIM2
SMBus ALERT interface	Implemented	Implemented	Not Implemented

Table 28-8. Module Clock Name

Module Name	Clock Name	Description
TWIS0	CLK_TWIS0	Peripheral Bus clock from the PBA clock domain
TWIS1	CLK_TWIS1	Peripheral Bus clock from the PBA clock domain
TWIS2	CLK_TWIS2	Peripheral Bus clock from the PBC clock domain

Table 28-9. Register Reset Values

Register	Reset Value
VR	0x00000120
PR	0x00000000

29. CAN Interface (CANIF)

Version 1.1.0.3

29.1 Features

- **Supports CAN 2.0A and 2.0B protocol specifications**
- **1 Mb/s maximum bitrate**
- **2 CAN channels**
- **16 Message Objects per CAN channel**
- **1 identifier (11 or 29 bits), 1 identifier mask and 8 bytes buffer per MOb**
- **Single shot and automatic transmit/receive modes**
- **Overrun mode**
- **Loop-back mode for bit rate detection**
- **Listen mode for bus monitoring**
- **System sleep mode support with wake-up on bus activity**
- **Programmable CAN clock source**

29.2 Overview

Control Area Network (CAN) is a serial communication protocol with high level of security. Each node is master on the bus but only one at a time is able to send a message.

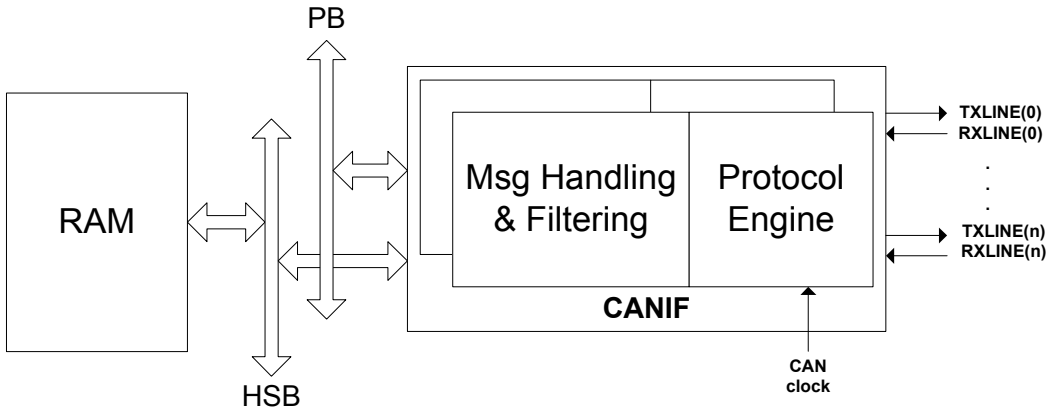
CANIF is a 32-bit interface for CAN channels. Each channel provides the following services:

- Message filtering
- Message and status handling
- Fault confinement
- Error detection and signalling
- Message validation and acknowledgement
- Bus arbitration
- Message framing
- Transfer rate and timing

These services, with the exception of message filtering and message handling, are described in the CAN protocol, please refer to *Bosch - CAN Specification* for more details.

29.3 Block Diagram

Figure 29-1. CANIF Block Diagram



29.4 I/O Lines Description

Table 29-1. I/O Lines Description

Pin Name	Pin Description	Type
TXLINE(n)	Transmission line of channel n	Output
RXLINE(n)	Reception line of channel n	Input

29.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

29.5.1 I/O Lines

CANIF pins are multiplexed with other peripherals. User must first program the I/O Controller to give control of the pins to the CANIF.

29.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by CANIF, it will stop functioning and resume operation after the system wakes up from sleep mode.

29.5.3 Clocks

CANIF is connected to both the HSB and the PB, and therefore uses a HSB clock (CLK_CANIF_HSB) and a PB clock (CLK_CANIF_PB). These clocks are generated by the Power Manager. These clocks are enabled at reset, and can be disabled in the Power Manager.

CANIF uses a GCLK as clock source (CAN clock) for the CAN bus communication (GCLK_CANIF). User must make sure this clock is running and frequency is correct before any operation.

29.5.4 Memory

Messages can be stored in CPU or HSB RAM, so user must allocate RAM space for CAN messages.

29.5.5 Interrupts

CANIF interrupt request line is connected to the interrupt controller. Using the CANIF interrupt requires the interrupt controller to be programmed first.

29.5.6 Debug Operation

All CAN channels are disabled when the CPU enters Debug mode. Communication in progress is not stopped. Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual, and the OCD Module Configuration section, for details.

29.6 Functional Description

29.6.1 Channel Configuration

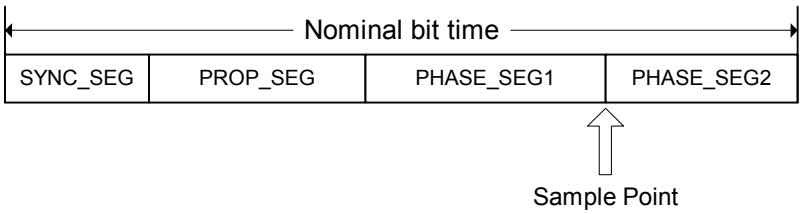
Channel configuration is done via the Configuration Register (CANCFG). This register is not write accessible once the channels have been enabled.

29.6.1.1 Bit timing

This section refers to chapter 8 (Bit timing requirements) of the CAN Specification.

The CAN bit rate is defined by the nominal bit time. Nominal bit time is divided into 4 time segments.

Figure 29-2. Partition of the Bit Time



The duration of each time segment is divided into time quanta (TQ). The total number of TQ in a bit time must be in the range [8..25].

The Time Quantum is a fixed unit of time derived from the GCLK_CANIF clock period:

$$TQ = \text{Prescaler} \times P_{GCLK\_CANIF} = (CANCFG.PRES+1) \times P_{GCLK\_CANIF}$$

Re-synchronization may lengthen or shorten the bit time, the upper bound is given by Synchronization Jump Width field in the Configuration Register (CANCFG.SJW).

The value of all previous parameters are defined in CANCFG register.

Table 29-2. CAN Parameter Settings

Parameter	Range	CANCFG field
SYNC_SEG	1	-
PROP_SEG	[1..8]TQ	PRS + 1
PHASE_SEG1	[1..8]TQ	PHS1 + 1

Table 29-2. CAN Parameter Settings

Parameter	Range	CANCFG field
PHASE_SEG2	[1..8]TQ	PHS2 + 1
Prescaler	[2..32]	PRES + 1
Sync Jump Width	[1..4]	SJW + 1

The bit duration is given by the formula:

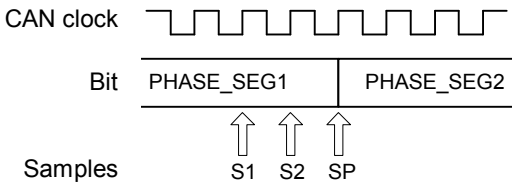
$$Tbit = (PRS + PHS1 + PHS2 + 4) \times (PRES + 1) \times P_{GCLK\_CANIF}$$

Note: PRES should not be set to 0, therefore CAN clock is at least divided by 2.

29.6.1.2 *Sampling method*

Bits are sampled between PHASE_SEG1 and PHASE_SEG2. By writing the Sampling Method bit (CANCFG.SM) to one, three samples are taken and a majority vote is performed.

Figure 29-3. Sampling by Majority Voting



Majority voting must not be used when the Prescaler field (CANCFG.PRES) is equal to zero.

29.6.1.3 *Operating modes*

CANIF has three operating modes, selectable by the Channel Mode field (CANCFG.CMODE):

– Normal mode (CANCFG.CMODE=00)
Default mode, TX and RX lines are connected to the transceiver. This mode is used to communicate with other nodes on the bus.

– Listening mode (CANCFG.CMODE=01)
The TX line is disconnected from the transceiver. The CAN channel cannot send any message nor acknowledge when a message has been received. The channel is in Error Passive mode and Transmit/Receive Error Counters (TEC/REC) are frozen. This mode is used to listen to CAN bus.

– Loop back mode (CANCFG.CMODE=10)
The TX line is internally connected to the RX line and disconnected from the transceiver. The CAN channel can only send messages or acknowledges to itself. The channel is in Error Passive mode and TEC/REC are frozen. This mode is used to detect the bit rate of the CAN bus by successive configuration of bit timing.

29.6.1.4 *Overrun mode*

When Overrun Mode is disabled, the MOb is disabled after successfully receiving a message. This prevents overwriting the received message if a second message is received. Overrun Mode is disabled by writing a zero to CANCFG.OVRM. Overrun Mode is disabled by default.

When Overrun Mode is enabled, the MOb is not disabled after a successful reception. Overrun Mode is enabled by writing a one to CANCFG.OVRM. The Overwrite bit in the MOb Status Register (MObSR.OVW) is set if a previously received message has been overwritten.

The mode configured by CANCFG.OVRM is used by all MObs configured for reception.

29.6.1.5 Memory pointer

Each channel uses a section of RAM for storing messages. User must allocate RAM space for the channels and store the base address of this space into the Channel RAM Base Address Register (CANRAMB). Four words per MOb in use must be allocated.

Channels operate independently so the allocated memory spaces do not need to be consecutive. Make sure that the memory spaces do not overlap.

29.6.2 Channel Handling

29.6.2.1 Initialization

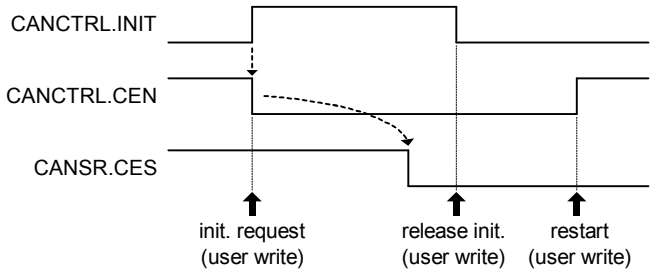
CAN channels are initialized by writing a one to the Initialization bit in the Control Register (CANCTRL.INIT).

Initialization resets all internal state machines and clears all user interface registers except CANRAMB, CANCFG and CANCTRL.INIT.

CANCTRL.INIT should not be cleared until the channel has been disabled. The channel is disabled by writing the Channel Enable bit (CANCTRL.CEN) to zero. When the Channel Enable status bit (CANSR.CEN) is zero, the channel has been disabled and CANCTRL.INIT can be written to zero. Thereafter the channel can be restarted by writing a one to CANCTRL.CEN. See Figure 29-4 for details.

It is not possible to write to other CANCTRL bits when CANCTRL.INIT is one. User must write a zero to CANCTRL.INIT before writing a new value to CANCTRL.

Figure 29-4. Initialization Sequence



Note: Initialization requires all clocks to be running.

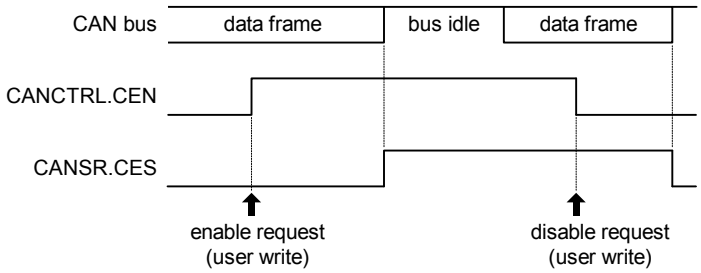
29.6.2.2 Enabling / Disabling

A channel is enabled and ready to communicate on the bus when it has detected a bus idle condition (i.e. 11 consecutive recessive bits).

The channel is enabled by writing a one to CANCTRL.CEN and disabled by writing a zero to this bit. The enable status of channel can be read in CANSR.CES bit.

The channel mode is not changed when the channel is disabled, i.e. the Fault Confinement Register (CANFC) is not cleared. Therefore, if the channel was in error passive mode before being disabled, it stays in error passive mode when re-enabled.

Figure 29-5. Enable and Disable Sequences

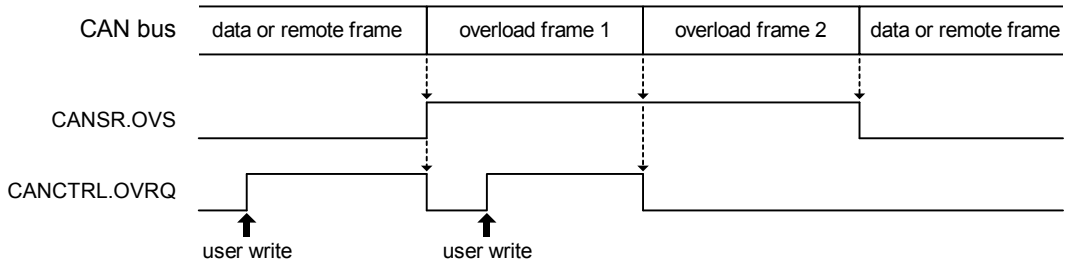


29.6.2.3 *Overload frames*

The CAN protocol allows nodes to send overload frames to provide extra delay between two messages.

User can at any time request to send overload frames by writing the Overload Request bit (CANCTRL.OVRQ) to one. A single overload frame will then be sent at the end of the next message. When transission of the overload frame starts, CANSR.OVS is set and CANCTRL.OVRQ is cleared. At this time, user can write CANCTRL.OVRQ to one again to send a second overload frame at the end of the first one. The CAN protocol specifies that maximum two overload frames can be sent.

Figure 29-6. Overload Frame Request



29.6.2.4 *Errors and fault confinement*

This section refers to chapter 6 (Error handling) and chapter 7 (Fault confinement) of the CAN Specification.

There are 5 different error types which are not mutually exclusive. Error status can be read in the Interrupt Status Register (CANISR). Error status bits are set by hardware and can only be cleared by user.

When the channel enters bus off state, the Bus Off Status bit (CANISR.BOFF) is set. In this state the channel can no longer communicate on the bus. The channel can leave bus off state if it detects 128 occurrences of 11 consecutive recessive bits on the bus.

The Last Selected MOb Status field (CANISR.LSMOB) identifies the MOb that was selected when the error occurred. For some error types, a MOb has not been selected yet when the error occurs. In this case, CANISR.LSMOB returns NONE.

According to the CAN specification, a channel can be in error active, error passive or bus off state. The bus state can be read in the CANFC.EMODE field:

Table 29-3. Bus State Coding

EMODE	State
0	error active state
1	error passive state
2 or 3	bus off state

This state depends on both transmit and receive counters value (TEC/REC), also available in the CANFC register (see CAN specification for more details).

29.6.2.5 Wake-up mode

In this mode the CAN channel is a wake-up source for the CPU. Detection of a falling edge on the CAN bus is interpreted as the start of frame (SOF) bit of the wake-up frame and will wake the CPU.

In order to use this mode, execute the following steps:

- Disable CAN channel by writing CANCTRL.CEN to zero
- Wait for channel disabled (CANSR.CES is cleared)
- Enable wake-up mode by writing CANCTRL.WKEN to one
- Optionally mask wake-up interrupt source by writing CANIMR.WKUPIM
- Enter sleep mode, stopping PB and CAN clocks
- Sleep
- Wake-up frame is detected, CANISR.WKUP is set and CPU is woken-up
- Clear CANISR.WKUP by writing it to zero
- Disable wake-up mode and enable CAN channel

Note 1: when channel is disabled PB registers are not cleared, user can resume current application by enabling channel again.

Note 2: wake-up frame cannot be received. Moreover next frames cannot be received until CPU is woken-up. Wake-up time depends on sleep modes (see sleep modes section).

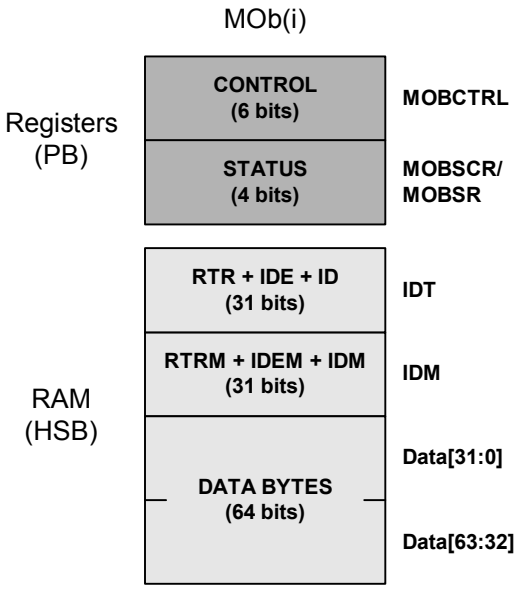
29.6.3 Message Handling

29.6.3.1 Message object structure

Message Objects (MOb) are message descriptors, used to store and handle CAN frames. User configures and gets status of MObs via user interface registers and writing and reading frames into allocated RAM space (Section 29.6.1.5).

MObs are independent and are allocated to one channel.

Figure 29-7. Message Object Structure



Where:

- RTR is the Remote Transmission Request, 0 means data frame and 1 remote frame
- IDE is the Identifier Extension Bit, 0 means standard format and 1 means extended format
- ID is the CAN identifier of message (11 bits in standard format and 29 bits in extended format)
- RTRM is the RTR bit Mask
- IDEM is the IDE bit Mask
- IDM are the ID bits Mask

The length of the message (DLC) is stored in the MOb Control Register (MOBCTRL).

The data stored in RAM should have the following format:

Figure 29-8. Identifier Tag (IDT)

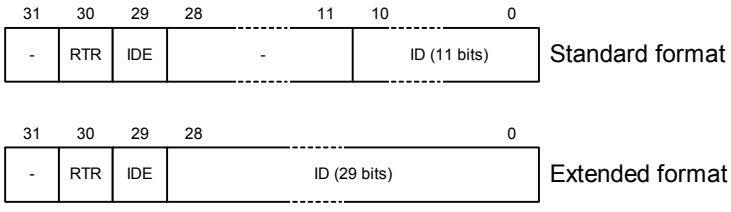


Figure 29-9. Identifier Mask (IDM)

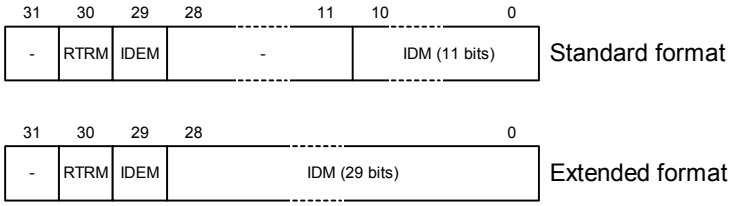
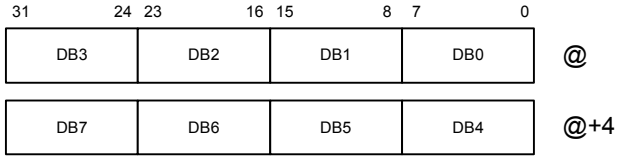


Figure 29-10. Data Fields (64 bits)



29.6.3.2 Transmission

Once a message has been written into RAM at the address corresponding to the selected MOb, user controls transmission through the MOBCTRL register:

- DLC[3:0] field: Data length code i.e. the number of byte to send, from 0 to 8
- DIR bit: MOb direction, 1 stands for transmission

Once MOb is enabled (by writing to MOBER), transmission starts as soon as bus idle is detected on the CAN bus. User can check if channel is sending a frame by reading CANSR.TS bit.

At the end of the successful transmission bit MOBESR.MENn is cleared and MOBSR.TXOK is set. To acknowledge interrupt and to free the MOb user must clear this status bit by writing a one to the associated bit in MOb Status Clear Register (MOBSCR).

CAN errors detected during transmission are reported in CANISR. Message will not be transmitted but the MOb remains enabled. The message will be automatically re-transmitted until successfully transmitted.

Several MObs can be enabled/disabled in one operation by writing to the MOBER/MOBDR registers:

- MOBER: Each bit correspond to an enable bit for a single MOb. Write 1 to set a bit and 0 to keep it unchanged.
- MOBDR: Each bit correspond to an enable bit for a single MOb. Write 1 to clear a bit and 0 to keep it unchanged.

If several MObs are enabled, the MOb with the lowest number is transmitted first. This rule is also used in case of a re-transmission (due to transmission error or contention).

29.6.3.3 Reception

Once the expected message has been written into RAM at the address corresponding to the selected MOb, user controls reception through the MOBCTRL register:

- DLC[3:0] field: Data length code i.e. the number of byte to receive, from 0 to 8

- DIR bit: MOb direction, 0 stands for reception

Once a MOb is enabled (by writing to MOBER), an incoming frame is compared (Section 29.6.4) with every MOb enabled for reception in order to select the MOb for storing the frame. User can check if the channel is receiving a frame by reading the CANSR.RS bit.

At the end of the successful reception, the complete message (ID + RTR + IDE + DATA bits) is stored in RAM, the MOBESR.MENn bit is cleared and MOBSR.RXOK is set. To acknowledge any interrupt and to free the MOb user must clear this status bit by writing a one to the associated bit in MOBSCR. The MOBSR.DLCW bit indicates if the received DLC does not correspond to MOBCTRL.DLC. Any such status should also be cleared by user.

CAN errors detected during reception are reported in CANISR register. A corrupted message is not stored to RAM but the selected MOb remains enabled.

User can enable/disable several MObs in one operation, by writing to MOBER/MOBDR registers (Section 29.6.3.2).

If several MObs are enabled, the MOb priority is given by the filtering order which is from low to high MOb number.

29.6.3.4 Automatic mode

All MObs are configured in automatic mode if the Automatic Mode (MOBCTRL.AM) bit is set.

The main configurations are:

- Remote frame reception - Data frame transmission
  MOb configuration: MOBCTRL.AM = 1, MOBCTRL.DIR = 0, IDT.RTR = 1
  IDT/IDM can be set to receive any identifier but transmission will be done with identifier received.
- Remote frame transmission - Data frame reception
  MOb configuration: MOBCTRL.AM = 1, MOBCTRL.DIR = 1, IDT.RTR = 1
  Remote frame is sent with IDT value. Reception uses current IDT value but IDM can be set to filter data frame.

Other configurations (IDT.RTR = 0) are possible but does not make sense.
Properties:

  – MOb handling is identical to single configuration (priority, access,...) but:
  – Bits MOBCTRL.AM and IDT.RTR are inverted at the end of first reception/transmission
  – Bit MOBSR.TXOK (or RXOK) is only set at the end of the transmission (or reception)

29.6.4 Message Filtering

The filtering process uses the ID tag (IDT) and ID mask (IDM) values defined in RAM. Comparison is done on the bits IDENTIFIER, RTR and IDE. Messages can therefore be filtered according to the identifier value, frame type (remote or data frame) and the format (standard or extended).

Each received bit is compared with the corresponding bit in the ID tag only if the corresponding bit in ID mask is set. Otherwise the received bit is considered as don’t care. The filtering result is true if all comparisons are true.

Examples with 11 bits of identifier (‘-‘ means don’t care):

ID received:	000.0010.1001 b	000.0010.1001 b
IDT:	000.0010.1010 b	000.0100.1000 b
IDM:	111.1111.0000 b	111.1111.0000 b
Comparison:	111.1111.- - - - b	111.1001.- - - - b
Accepted:	Y	N

The filtering process scans each MOb enabled and configured for reception, from MOb 0, in order to find the MOb that matches the conditions. The first MOb to match is selected for storing the message once received successfully. If no MOb matches, the message is discarded.

29.6.5 Channel Interrupts

There are several sources of interrupts and user can mask each of them. Some sources are grouped into a single interrupt request line. There are 5 interrupt request lines per channel.

- Wake-up interrupt: Wake-up condition detected
- Error interrupt: Any CAN error detected during a communication
- Bus off interrupt: The CAN protocol engine entered in bus off state
- Took interrupt: At least one MOb completed a transmission
- Waxed interrupt: At least one MOb completed a reception

The CANIMR and MOBIMR are used for masking interrupts. These registers are read-only. In order to set or clear interrupt mask bits, user must write to the following registers:

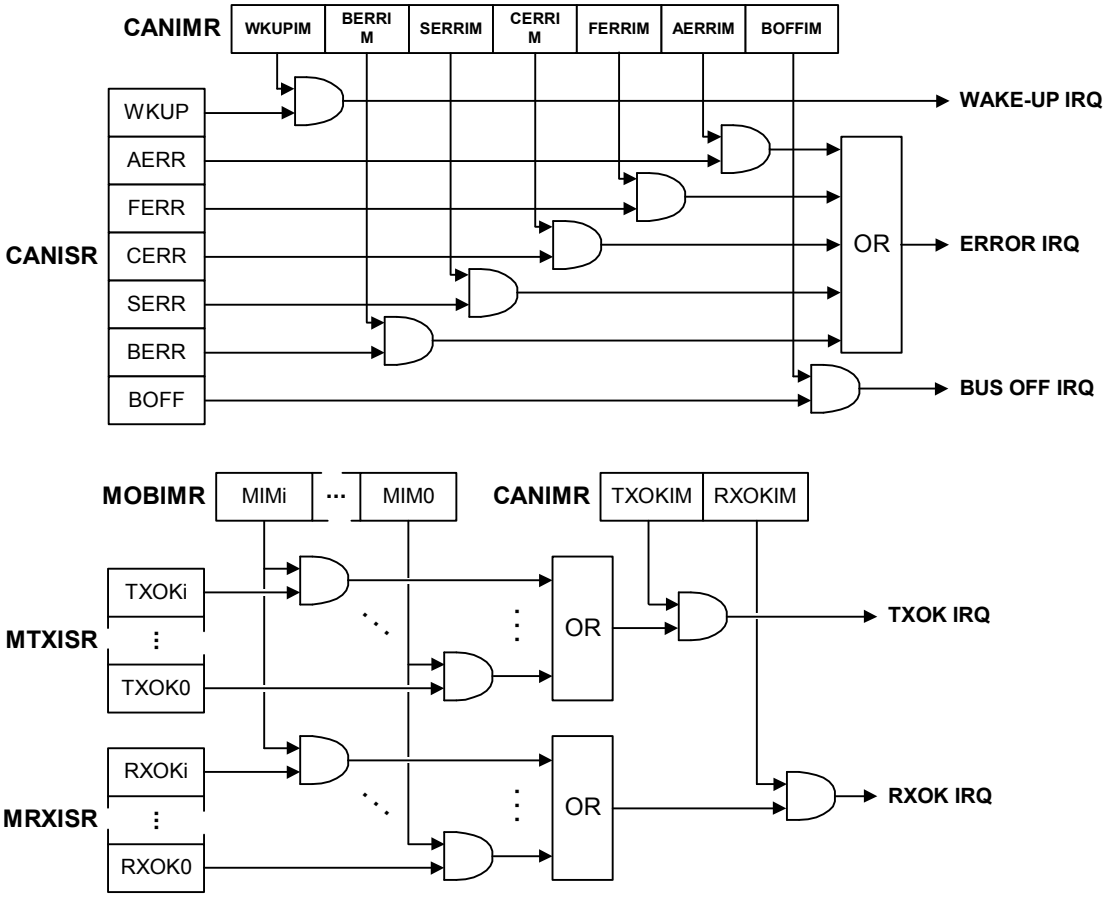
- CANIER / MOBIER: Writing a bit to one sets the corresponding bit in CANIMR / MOBIMR. Writing a bit to 0 has no effect.
- CANIDR / MOBIDR: Writing a bit to one clears the corresponding bit in CANIMR / MOBIMR. Writing a bit to 0 has no effect.

To acknowledge an interrupt request, user must clear the corresponding bit in the corresponding status register (CANISR, MTXISR or MRXISR). To clear status bits, user must access the following write-only registers:

- CANISCR / MTXISCR / MRXISCR: Writing a bit to one clears the corresponding bit in CANISR / MTXISR / MRXISR. Writing a bit to 0 has no effect.

For each MOb, the bits TXOK and RXOK are also accessible in MOBSCR register for clear access and MOBSR register for read access.

Figure 29-11. Interrupt Channel Structure



29.7 User Interface

Offsets are relative to the base address allocated to CANIF and the channel number.

Figure 29-12. Address Map Overview

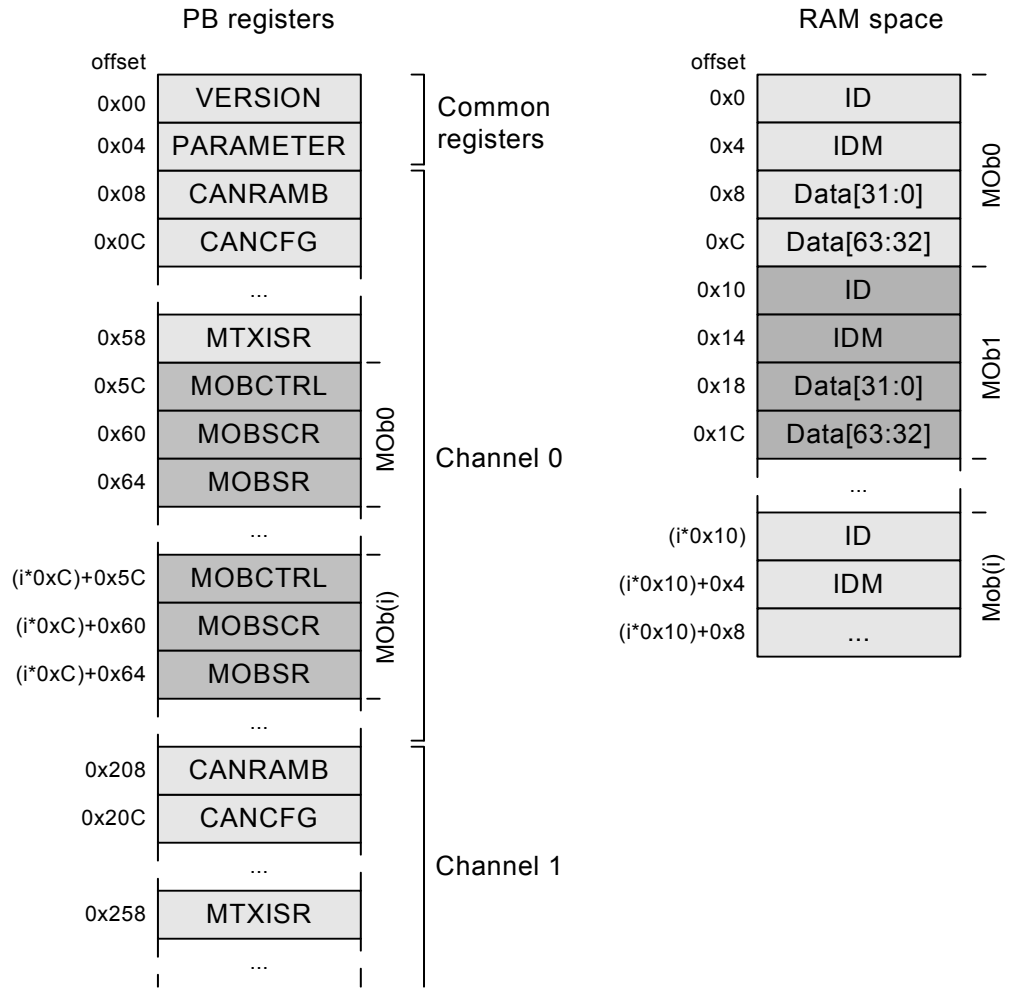


Table 29-4. CANIF Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Version Register	VERSION	Read-only	-(Note:)
0x04	Parameter Register	PARAMETER	Read-only	-(Note:)
0x08	RAM Base Address Register	CANRAMB	Read/Write	0x00000000
0x0C	Configuration Register	CANCFG	Read/Write	0x00000001
0x10	Control Register	CANCTRL	Read/Write	0x00000000
0x14	Status Register	CANSR	Read-only	0x00000000
0x18	Fault Confinement Register	CANFC	Read-only	0x00000000
0x1C	Interrupt Enable Register	CANIER	Write-only	0x00000000
0x20	Interrupt Disable Register	CANIDR	Write-only	0x00000000

Table 29-4. CANIF Register Memory Map

0x24	Interrupt Mask Register	CANIMR	Read-only	0x00000000
0x28	Interrupt Status Clear Register	CANISCR	Write-only	0x00200000
0x2C	Interrupt Status Register	CANISR	Read-only	0x00200000
0x30	MOb Search Register	MOBSCH	Read-only	0x00202020
0x34	MOb Enable Register	MOBER	Write-only	0x00000000
0x38	MOb Disable Register	MOBDR	Write-only	0x00000000
0x3C	MOb Enable Status Register	MOBESR	Read-only	0x00000000
0x40	MOb Interrupt Enable Register	MOBIER	Write-only	0x00000000
0x44	MOb Interrupt Disable Register	MOBIDR	Write-only	0x00000000
0x48	MOb Interrupt Mask Register	MOBIMR	Read-only	0x00000000
0x4C	MOb RX Interrupt Status Clear Register	MRXISCR	Write-only	0x00000000
0x50	MOb RX Interrupt Status Register	MRXISR	Read-only	0x00000000
0x54	MOb TX Interrupt Status Clear Register	MTXISCR	Write-only	0x00000000
0x58	MOb TX Interrupt Status Register	MTXISR	Read-only	0x00000000
0x5C	MOb Control Register	MOBCTRL	Read/Write	0x00000000
0x60	MOb Status Clear Register	MOBSCR	Write-only	0x00000000
0x64	MOb Status Register	MOBSR	Read-only	0x00000000

Note: The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

29.7.1 Version Register

Name: VERSION

Access type: Read-only

Offset: 0x00

Reset Value: -

31	30	29	28	27	26	25	24
-	-	MNCH0					

23	22	21	20	19	18	17	16
-	CHNO			VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **MNCH0: MOb Number Channel #0**
  Number of MOb for channel 0 (1..32).
- **CHNO: Channel Number**
  Number of CAN channels (1..5).
- **Variant: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

29.7.2 Parameter Register

Name: PARAMETER

Access type: Read-only

Offset: 0x04

Reset Value: -

31	30	29	28	27	26	25	24
-	-	MNCH4					

23	22	21	20	19	18	17	16
-	-	MNCH3					

15	14	13	12	11	10	9	8
-	-	MNCH2					

7	6	5	4	3	2	1	0
-	-	MNCH1					

- **MNCH4: MOb Number Channel #4**
  Number of MOb for channel 4 (0..32).
- **MNCH3: MOb Number Channel #3**
  Number of MOb for channel 3 (0..32).
- **MNCH2: MOb Number Channel #2**
  Number of MOb for channel 2 (0..32).
- **MNCH1: MOb Number Channel #1**
  Number of MOb for channel 1 (0..32).

29.7.3 RAM Base Address Register

Name: CANRAMB

Access type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RAMBASE[31:24]							

23	22	21	20	19	18	17	16
RAMBASE[23:16]							

15	14	13	12	11	10	9	8
RAMBASE[15:8]							

7	6	5	4	3	2	1	0
RAMBASE[7:0]							

- **RAMBASE: RAM Base Address**
  CAN channel RAM base address.

29.7.4 Configuration Register

Name: CANCFG

Access type: Read/Write

Offset: 0x0C

Reset Value: 0x00000001

31	30	29	28	27	26	25	24
-	-	-	-	-	OVRM	CMODE	

23	22	21	20	19	18	17	16
-	-	SM	SJW		PRS		

15	14	13	12	11	10	9	8
-	-	PHS2			PHS1		

7	6	5	4	3	2	1	0
-	-	PRES					

- **OVRM: Overrun Mode**
  Overrun mode (MOb is not disabled after successful reception, therefore overwrite is possible).
- **CMODE: Channel Mode**
  00: Normal mode.
  01: Listening mode.
  10: Loop back mode.
- **SM: Sampling Method**
  0: Once.
  1: Three times.
- **SJW: Synchronization Jump Width**
  Maximum number of time quanta for resynchronization.
- **PRS: Propagation Segment**
  Number of time quanta for propagation segment.
- **PHS2: Phase Segment 2**
  Number of time quanta for phase segment 2.
- **PHS1: Phase Segment 1**
  Number of time quanta for phase segment 1.
- **PRES: Prescaler**
  CAN clock prescaler. Defines time quantum duration. Should not be set to 0.

29.7.5 Control Register

Name: CANCTRL

Access type: Read/Write

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	WKEN	OVRQ	CEN	INIT

- **WKEN: Wake-up Enable**
  0: Wake-up mode disabled.
  1: Wake-up mode enabled, any bus activity will set CANISR.WKUP.
- **OVRQ: Overload Request**
  0: No overload frame request pending.
  1: Overload frame request pending, overload frame will be sent at the end of next received frame.
- **CEN: Channel Enable**
  0: No CAN channel enable request pending.
  1: CAN channel enable request pending.
- **INIT: Initialization**
  0: CAN channel not initialized.
  1: Initialize CAN channel.

29.7.6 Status Register

Name: CANSR

Access type: Read-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	RS	TS	OVS	CES

- **RS: Reception Status**
  - 0: No frame is being received
  - 1: Frame is being received
- **TS: Transmission Status**
  - 0: No frame is being transmitted
  - 1: Frame is being transmitted
- **OVS: Overload Status**
  - 0: No overload frame is being transmitted
  - 1: Overload frame is being transmitted
- **CES: Channel Enable Status**
  - 0: Channel disabled
  - 1: Channel ready

29.7.7 Fault Confinement Register

Name: CANFC

Access type: Read-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	EMODE	

15	14	13	12	11	10	9	8
TEC							

7	6	5	4	3	2	1	0
REC							

- **EMODE: Error Mode**
  - 00: Error active
  - 01: Error passive
  - 1X: Bus off
- **TEC: Transmit Error Counter**
- **REC: Reception Error Counter**

29.7.8 Interrupt Enable Register

Name: CANIER

Access type: Write-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	TXOKIM
7	6	5	4	3	2	1	0
RXOKIM	WKUPIM	BERRIM	SERRIM	CERRIM	FERRIM	AERRIM	BOFFIM

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

29.7.9 Interrupt Disable Register

Name: CANIDR

Access type: Write-only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	TXOKIM
7	6	5	4	3	2	1	0
RXOKIM	WKUPIM	BERRIM	SERRIM	CERRIM	FERRIM	AERRIM	BOFFIM

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

29.7.10 Interrupt Mask Register

Name: CANIMR

Access type: Read-only

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	TXOKIM
7	6	5	4	3	2	1	0
RXOKIM	WKUPIM	BERRIM	SERRIM	CERRIM	FERRIM	AERRIM	BOFFIM

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

29.7.11 Interrupt Status Clear Register

Name: CANISCR

Access type: Write-only

Offset: 0x28

Reset Value: 0x00200000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	LSMOB					

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	WKUP	BERR	SERR	CERR	FERR	AERR	BOFF

- **LSMOB: Last Selected MOb Status Clear**
  Write all bits to one to clear Last Selected MOb number.
  Writing these bits to any other value has no effect.
- **WKUP: Wake-up Status Clear**
  Write to one to clear Wake-up status.
  Writing this bit to zero has no effect.
- **BERR: Bit Error Status Clear**
  Write to one to clear Bit Error status.
  Writing this bit to zero has no effect.
- **SERR: Stuff Error Status Clear**
  Write to one to clear Stuffing Error status.
  Writing this bit to zero has no effect.
- **CERR: CRC Error Status Clear**
  Write to one to clear CRC Error status.
  Writing this bit to zero has no effect.
- **FERR: Form Error Status Clear**
  Write to one to clear Form Error status.
  Writing this bit to zero has no effect.
- **AERR: Acknowledge Error Status Clear**
  Write to one to clear Acknowledge Error status.
  Writing this bit to zero has no effect.
- **BOFF: Bus Off Status Clear**
  Write to one to clear Bus Off status.
  Writing this bit to zero has no effect.

29.7.12 Interrupt Status Register

Name: CANISR

Access type: Read-only

Offset: 0x2C

Reset Value: 0x00200000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	LSMOB					

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	WKUP	BERR	SERR	CERR	FERR	AERR	BOFF

- **LSMOB: Last Selected MOb Status**
  Last selected MOB number (1X: none, 0X: MOb X selected).
- **WKUP: Wake-up Status**
  0: No wake-up request detected.
  1: Wake-up request (bus activity detected while CANCTRL.WKEN=1).
- **BERR: Bit Error Status**
  0: No bit error detected in current frame.
  1: Bit error detected in current frame.
- **SERR: Stuff Error Status**
  0: No stuffing error detected in current frame.
  1: Stuffing error detected in current frame.
- **CERR: CRC ERror Status**
  0: No CRC error detected in current frame.
  1: CRC error detected in current frame.
- **FERR: Form Error Status**
  0: No form error detected in current frame.
  1: Form error detected in current frame.
- **AERR: Acknowledge Error Status**
  0: No acknowledge error detected in current frame.
  1: Acknowledge error detected in current frame.
- **BOFF: Bus Off Status**
  0: CAN channel not in Bus Off state.
  1: CAN channel switched to Bus Off state.

29.7.13 MOb Search Register

Name: MOSCH

Access type: Read-only

Offset: 0x30

Reset Value: 0x00202020

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	MTXOK					

15	14	13	12	11	10	9	8
-	-	MRXOK					

7	6	5	4	3	2	1	0
-	-	MAV					

- **MTXOK: MOb TxOK**
  MOB TXOK number, with highest priority
  1XXXXX: No MOb found
  0XXXXX: MOb X found
- **MRXOK: MOb RxOK**
  MOB RXOK number, with highest priority
  1XXXXX: No MOb found
  0XXXXX: MOb X found
- **MAV: MOb Available**
  MOB available number, with highest priority
  1XXXXX: No MOb found
  0XXXXX: MOb X found

29.7.14 MOb Enable Register

Name: MOBER

Access type: Write-only

Offset: 0x34

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MEN[31:24]							

23	22	21	20	19	18	17	16
MEN[23:16]							

15	14	13	12	11	10	9	8
MEN[15:8]							

7	6	5	4	3	2	1	0
MEN[7:0]							

- **MEN: MOb Enable**
  Writing a bit to zero has no effect.
  Writing a bit to one will enable the corresponding MOb.

29.7.15 MOb Disable Register

Name: MOBDR

Access type: Write-only

Offset: 0x38

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MDIS[31:24]							

23	22	21	20	19	18	17	16
MDIS[23:16]							

15	14	13	12	11	10	9	8
MDIS[15:8]							

7	6	5	4	3	2	1	0
MDIS[7:0]							

- **MDIS: MOb Disable**

  Writing a bit to zero has no effect.

  Writing a bit to one will disable the corresponding MOb.

29.7.16 MOb Enable Status Register

Name: MOBESR

Access type: Read-only

Offset: 0x3C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MEN[31:24]							

23	22	21	20	19	18	17	16
MEN[23:16]							

15	14	13	12	11	10	9	8
MEN[15:8]							

7	6	5	4	3	2	1	0
MEN[7:0]							

- **MEN: MOb Enable**
  0: The corresponding MOb is disabled
  1: The corresponding MOb is enabled
  This bit is cleared when the corresponding bit in MOBDR is written to one.
  This bit is set when the corresponding bit in MOBER is written to one.

29.7.17 MOb Interrupt Enable Register

Name: MOBIER

Access type: Write-only

Offset: 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MIM[31:24]							

23	22	21	20	19	18	17	16
MIM[23:16]							

15	14	13	12	11	10	9	8
MIM[15:8]							

7	6	5	4	3	2	1	0
MIM[7:0]							

- **MIM: MOb Interrupt Mask**
  Writing a bit to zero has no effect.
  Writing a bit to one will set the corresponding bit in MOBIMR.

29.7.18 MOb Interrupt Disable Register

Name: MOBIDR

Access type: Write-only

Offset: 0x44

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MIM[31:24]							

23	22	21	20	19	18	17	16
MIM[23:16]							

15	14	13	12	11	10	9	8
MIM[15:8]							

7	6	5	4	3	2	1	0
MIM[7:0]							

- **MIM: MOb Interrupt Mask**
  Writing a bit to zero has no effect.
  Writing a bit to one will set the corresponding bit in MOBIMR.

29.7.19 MOb Interrupt Mask Register

Name: MOBIMR

Access type: Read-only

Offset: 0x48

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
MIM[31:24]							

23	22	21	20	19	18	17	16
MIM[23:16]							

15	14	13	12	11	10	9	8
MIM[15:8]							

7	6	5	4	3	2	1	0
MIM[7:0]							

- **MIM: MOb Interrupt Mask**

  0: The corresponding MOb interrupt is disabled.

  1: The corresponding MOb interrupt is enabled.

  This bit is cleared when the corresponding bit in MOBIDR is written to one.

  This bit is set when the corresponding bit in MOBIER is written to one.

29.7.20 MOb RX Interrupt Status Clear Register

Name: MRXISCR

Access type: Write-only

Offset: 0x4C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RXOK[31:24]							

23	22	21	20	19	18	17	16
RXOK[23:16]							

15	14	13	12	11	10	9	8
RXOK[15:8]							

7	6	5	4	3	2	1	0
RXOK[7:0]							

- **RXOK: Reception Successful**
  Writing a bit to zero has no effect.
  Writing a bit to one will clear the corresponding bit in MRXISR.

29.7.21 MOb RX Interrupt Status Register

Name: MRXISR

Access type: Read-only

Offset: 0x50

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RXOK[31:24]							

23	22	21	20	19	18	17	16
RXOK[23:16]							

15	14	13	12	11	10	9	8
RXOK[15:8]							

7	6	5	4	3	2	1	0
RXOK[7:0]							

- **RXOK: Reception Successful**

  0: The corresponding MOb has not completed a reception.

  1: The corresponding MOb has completed a reception (same bit as MOBSRn.RXOK).

  This bit is cleared when the corresponding bit in MRXISCR is written to one.

  This bit is set when the corresponding MOb has completed a reception.

29.7.22 MOb TX Interrupt Status Clear Register

Name: MTXISCR

Access type: Write-only

Offset: 0x54

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
TXOK[31:24]							

23	22	21	20	19	18	17	16
TXOK[23:16]							

15	14	13	12	11	10	9	8
TXOK[15:8]							

7	6	5	4	3	2	1	0
TXOK[7:0]							

- **TXOK: Transmission successful**
  Writing a bit to zero has no effect.
  Writing a bit to one will clear the corresponding bit in MTXISR.

29.7.23 MOb TX Interrupt Status Register

Name: MTXISR

Access type: Read-only

Offset: 0x58

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
TXOK[31:24]							

23	22	21	20	19	18	17	16
TXOK[23:16]							

15	14	13	12	11	10	9	8
TXOK[15:8]							

7	6	5	4	3	2	1	0
TXOK[7:0]							

- **TXOK: Transmission Successful**

  0: The corresponding MOb has not completed a transmission.

  1: The corresponding MOb has completed a transmission (same bit as MOBSRn.TXOK).

  This bit is cleared when the corresponding bit in MTXISCR is written to one.

  This bit is set when the corresponding MOb has completed a transmission.

29.7.24 MOb Control Register

Name: MOBCTRLn

Access type: Read/Write

Offset: 0x5C + [n * 0xC]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	AM	DIR	DLC			

- **AM: Automatic Mode**
  0: Automatic transmit mode
  1: Automatic receive mode
- **DIR: Transfer Direction**
  0: Reception
  1: Transmission
- **DLC: Data Length Code**
  Valid data length code is from 0 to 8.

29.7.25 MOb Status Clear Register

Name: MOBSCRn

Access type: Write-only

Offset: 0x60 + [n * 0xC]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	OVW	DLCW	TXOK	RXOK

- **OVW: Overwrite**
  Writing this bit to zero has no effect.
  Writing this bit to one clears the OVW status in MOBSR.
- **DLCW: DLC Warning**
  Writing this bit to zero has no effect.
  Writing this bit to one clears DLCW status in MOBSR.
- **TXOK: Transmission successful**
  Writing this bit to zero has no effect.
  Writing this bit to one clears TXOK status in MOBSR.
- **RXOK: Reception successful**
  Writing this bit to zero has no effect.
  Writing this bit to one clears RXOK status in MOBSR.

29.7.26 MOb Status Register

Name: MOBSRn

Access type: Read-only

Offset: 0x64 + [n * 0xC]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	OVW	DLCW	TXOK	RXOK

- **OVW: Overwrite**
  0: Previous message has not been overwritten.
  1: A new message has been received and overwritten previous one (if CANCFG.OVRM is set).
- **DLCW: DLC Warning**
  0: Received DLC matches MOBCTRL.DLC.
  1: Received DLC is different from MOBCTRL.DLC.
- **TXOK: Transmission Successful**
  0: Transmission not completed or not successful.
  1: Transmission completed and successful.
- **RXOK: Reception Successful**
  0: Reception not completed or not successful.
  1: Reception completed and successful.

29.8 Module Configuration

The specific configuration for each CANIF instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Refer to the Power Manager chapter for details.

Table 29-5. Module Configuration

Feature	CANIF
CANIF channels	2

Table 29-6. Module Clock Name

Module name	Clock name	Description
CANIF	CLK_CANIF_HSB	HSB clock
	CLK_CANIF_PB	Peripheral Bus clock from the PBC clock domain
	GCLK	The generic clock used for the CANIF is GCLK1

Table 29-7. Register Reset Values

Register	Reset Value
VERSION	0x10200110
PARAMETER	0x00000010

30. Inter-IC Sound Controller (IISC)

Rev.: 2.0.0.0

30.1 Features

- **Compliant with Inter-IC Sound (I^2S) bus specification**
- **Master, slave, and controller modes:**
  - **Slave: data received/transmitted**
  - **Master: data received/transmitted and clocks generated**
  - **Controller: clocks generated**
- **Individual enable and disable of receiver, transmitter, and clocks**
- **Configurable clock generator common to receiver and transmitter:**
  - **Suitable for a wide range of sample frequencies (fs), including 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz, 96 kHz, and 192 kHz**
  - **16 fs to 1024 fs Master Clock generated for external oversampling ADCs**
- **Several data formats supported:**
  - **32-, 24-, 20-, 18-, 16-, and 8-bit mono or stereo format**
  - **16- and 8-bit compact stereo format, with left and right samples packed in the same word to reduce data transfers**
- **Several data frame formats supported:**
  - **2-channel I^2S with Word Select**
  - **1- to 8-channel Time Division Multiplexed (TDM) with Frame Sync**
- **DMA interfaces for receiver and transmitter to reduce processor overhead:**
  - **Either one DMA channel for all audio channels, or**
  - **One DMA channel per audio channel**
- **Smart holding registers management to avoid audio channels mix after overrun or underrun**

30.2 Overview

The Inter-IC Sound Controller (IISC) provides a 5-wire, bidirectional, synchronous, digital audio link with external audio devices: ISDI, ISDO, IWS, ISCK, and IMCK pins.

This controller is compliant with the Inter-IC Sound (I^2S) bus specification and supports TDM interface with external multi-channel audio codecs.

The IISC consists of a Receiver, a Transmitter, and a common Clock Generator, that can be enabled separately, to provide Master, Slave, or Controller modes with Receiver, Transmitter, or both active.

Peripheral DMA channels, separate for the Receiver and for the Transmitter, allow a continuous high bitrate data transfer without processor intervention to the following:

- Audio CODECs in Master, Slave, or Controller mode
- Stereo DAC or ADC through dedicated I^2S serial interface
- Multi-channel or multiple stereo DACs or ADCs, using the TDM format

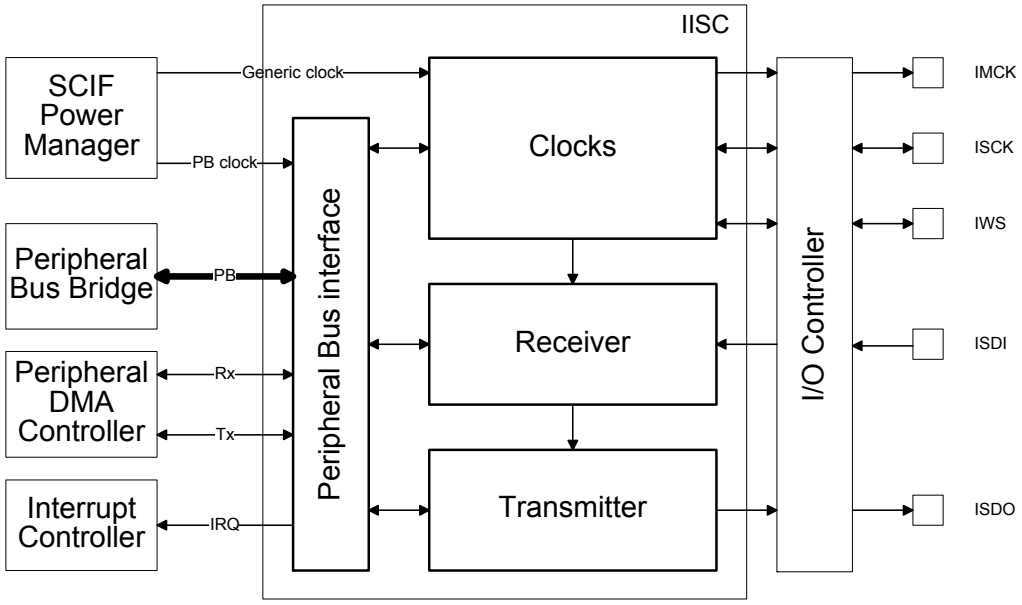
The IISC can use either a single DMA channel for all audio channels or one DMA channel per audio channel.

The 8- and 16-bit compact stereo format allows reducing the required DMA bandwidth by transferring the left and right samples within the same data word.

In Master Mode, the IISC allows outputting a 16 fs to 1024 fs Master Clock, in order to provide an oversampling clock to an external audio codec or digital signal processor (DSP).

30.3 Block Diagram

Figure 30-1. IISC Block Diagram



30.4 I/O Lines Description

Table 30-1. I/O Lines Description

Pin Name	Pin Description	Type
IMCK	Master Clock	Output
ISCK	Serial Clock	Input/Output
IWS	I^2S Word Select or TDM Frame Sync	Input/Output
ISDI	Serial Data Input	Input
ISDO	Serial Data Output	Output

30.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

30.5.1 I/O lines

The IISC pins may be multiplexed with I/O Controller lines. The user must first program the I/O Controller to assign the desired IISC pins to their peripheral function. If the IISC I/O lines are not used by the application, they can be used for other purposes by the I/O Controller. It is required to enable only the IISC inputs and outputs actually in use.

30.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the IISC, the IISC will stop functioning and resume operation after the system wakes up from sleep mode.

30.5.3 Clocks

The clock for the IISC bus interface (CLK_IISC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the IISC before disabling the clock, to avoid freezing the IISC in an undefined state.

One of the generic clocks is connected to the IISC. The generic clock (GCLK_IISC) can be set to a wide range of frequencies and clock sources. The GCLK_IISC must be enabled and configured before use. Refer to the module configuration section for details on the GCLK_IISC used for the IISC. The frequency for this clock has to be set as described in Table.

30.5.4 DMA

The IISC DMA handshake interfaces are connected to the Peripheral DMA Controller. Using the IISC DMA functionality requires the Peripheral DMA Controller to be programmed first.

30.5.5 Interrupts

The IISC interrupt line is connected to the Interrupt Controller. Using the IISC interrupt requires the Interrupt Controller to be programmed first.

30.5.6 Debug Operation

When an external debugger forces the CPU into debug mode, the IISC continues normal operation. If this module is configured in a way that requires it to be periodically serviced by the CPU through interrupt requests or similar, improper operation or data loss may result during debugging.

30.6 Functional Description

30.6.1 Initialization

The IISC features a Receiver, a Transmitter, and, for Master and Controller modes, a Clock Generator. Receiver and Transmitter share the same Serial Clock and Word Select.

Before enabling the IISC, the chosen configuration must be written to the Mode Register (MR). The IMCKMODE, MODE, FORMAT, and DATALENGTH fields in the MR register must be written. If FORMAT is configured in one of the TDM formats, then the NBCHAN and TDMFS fields must also be written. If the IMCKMODE field is written as one, then the IMCKFS field should be written with the chosen ratio, as described in Section 30.6.6 ”Serial Clock and Word Select Generation” on page 808.

Once the Mode Register has been written, the IISC Clock Generator, Receiver, and Transmitter can be enabled by writing a one to the CKEN, RXEN, and TXEN bits in the Control Register (CR). The Clock Generator can be enabled alone, in Controller Mode, to output clocks to the IMCK, ISCK, and IWS pins. The Clock Generator must also be enabled if the Receiver or the Transmitter is enabled.

The Clock Generator, Receiver, and Transmitter can be disabled independently by writing a one to CR.CXDIS, CR.RXDIS and/or CR.TXDIS respectively. Once requested to stop, they will only stop when the transmission of the pending frame transmission will be completed.

30.6.2 Basic Operation

The Receiver can be operated by reading the Receiver Holding Register (RHR), whenever the Receive Ready (RXRDY) bit in the Status Register (SR) is set. Successive values read from RHR will correspond to the samples from the left and right audio channels, or from channels 0 to MR.NBCHAN in TDM mode, for the successive frames.

The Transmitter can be operated by writing to the Transmitter Holding Register (RHR), whenever the Transmit Ready (TXRDY) bit in the Status Register (SR) is set. Successive values written to THR should correspond to the samples from the left and right audio channels, or from channels 0 to MR.NBCHAN in TDM mode, for the successive frames.

The Receive Ready and Transmit Ready bits can be polled by reading the Status Register.

The IISC processor load can be reduced by enabling interrupt-driven operation. The RXRDY and/or TXRDY interrupt requests can be enabled by writing a one to the corresponding bit in the Interrupt Enable Register (IER). The interrupt service routine associated to the IISC interrupt request will then be executed whenever the Receive Ready or the Transmit Ready status bit is set.

30.6.3 Master, Controller, and Slave Modes

In Master and Controller modes, the IISC provides the Master Clock, the Serial Clock and the Word Select. IMCK, ISCK, and IWS pins are outputs.

In Controller mode, the IISC Receiver and Transmitter are disabled. Only the clocks are enabled and used by an external receiver and/or transmitter.

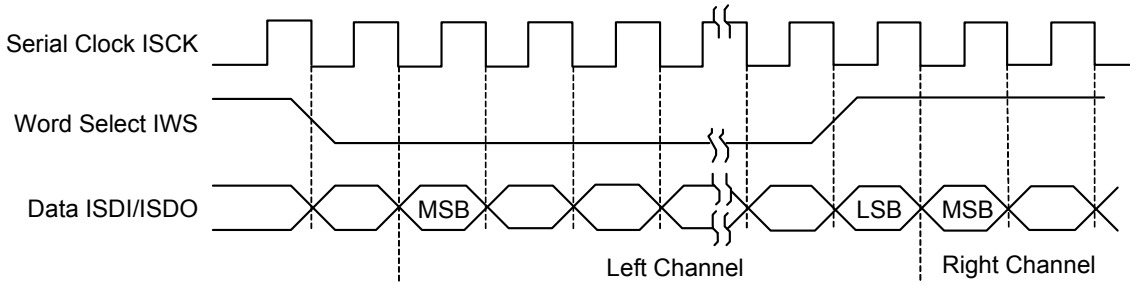
In Slave mode, the IISC receives the Serial Clock and the Word Select from an external master. ISCK and IWS pins are inputs.

The mode is selected by writing the MODE field of the Mode Register (MR). Since the MODE field changes the direction of the IWS and ISCK pins, the Mode Register should only be written when the IISC is stopped, in order to avoid unwanted glitches on the IWS and ISCK pins.

30.6.4 I^2S Reception and Transmission Sequence

As specified in the I^2S protocol, data bits are left-adjusted in the Word Select time slot, with the MSB transmitted first, starting one clock period after the transition on the Word Select line.

Figure 30-2. I^2S Reception and Transmission Sequence



Data bits are sent on the falling edge of the Serial Clock and sampled on the rising edge of the Serial Clock. The Word Select line indicates the channel in transmission, a low level for the left channel and a high level for the right channel.

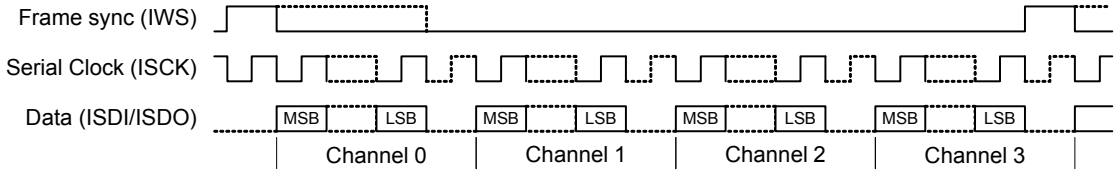
The length of transmitted words can be chosen among 8, 16, 18, 20, 24, and 32 bits by writing the MR.DATALENGTH field.

If the time slot allows for more data bits than written in the MR.DATALENGTH field, zeroes are appended to the transmitted data word or extra received bits are discarded. If the time slot allows for less data bits than written, the extra bits to be transmitted are not sent or the missing bits are set to zero in the received data word.

30.6.5 TDM Reception and Transmission Sequence

In Time Division Multiplexed (TDM) format, 1 to 8 data words are sent or received within each frame, As in the I^2S protocol, data bits are left-adjusted in the channel time slot, with the MSB transmitted first, starting one clock period after the transition on the Word Select line. Each time slot is 32-bit long.

Figure 30-3. TDM Reception and Transmission Sequence



Data bits are sent on the falling edge of the Serial Clock and sampled on the rising edge of the Serial Clock. The IWS pin provides a frame synchronization signal, starting one ISCK period before the MSB of channel 0.

The Time Division Multiplexed (TDM) format is selected by writing a one to the MR.FORMAT field.

The Frame Sync pulse can be either one ISCK period or one 32-bit time slot. This selection is done by writing the MR.TDMFS bit.

The number of channels is selected by writing the MR.CHANNELS field.

The length of transmitted words can be chosen among 8, 16, 18, 20, 24, and 32 bits by writing the MR.DATALENGTH field.

If the time slot allows for more data bits than programmed in the MR.DATALENGTH field, zeroes are appended to the transmitted data word or extra received bits are discarded. If the time slot allows for less data bits than programmed, the extra bits to be transmitted are not sent or the missing bits are set to zero in the right-adjusted received data word.

30.6.6 Serial Clock and Word Select Generation

The generation of clocks in the IISC is described in Figure 30-4 on page 810.

In Slave mode, the Serial Clock and Word Select Clock are driven by an external master. ISCK and IWS pins are inputs and no generic clock is required by the IISC.

In Master mode, the user can configure the Master Clock, Serial Clock, and Word Select Clock through the Mode Register (MR). IMCK, ISCK, and IWS pins are outputs and a generic clock is used to derive the IISC clocks.

Audio codecs connected to the IISC pins may require a Master Clock signal with a frequency multiple of the audio sample frequency (fs), such as 256fs. When the IISC is in Master mode, writing a one to MR.IMCKMODE will output GCLK_IISC as Master Clock to the IMCK pin, and will divide GCLK_IISC to create the internal bit clock, output on the ISCK pin. The clock division factor is defined by writing to MR.IMCKFS and MR.DATALENGTH, as described "IMCKFS: Master Clock to fs Ratio" on page 816.

The Master Clock (IMCK) frequency is 8*(NBCHAN+1)*(IMCKFS+1) times the sample frequency (fs), i.e. IWS frequency. The Serial Clock (ISCK) frequency is (NBCHAN+1)*Slot Length times the sample frequency (fs), where Slot Length is defined in Table 30-2 on page 809.

Table 30-2. Slot Length

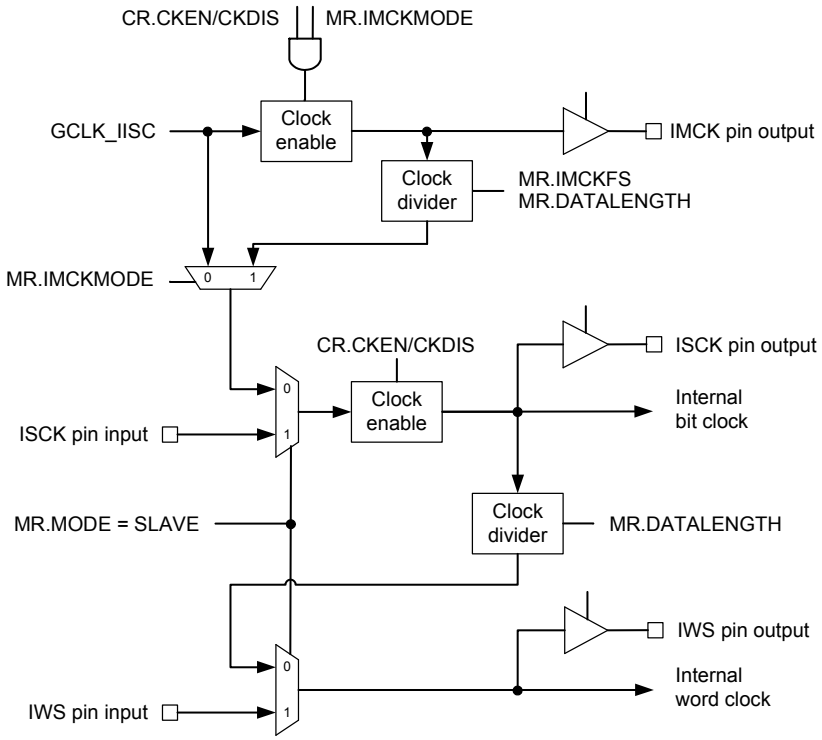
MR.DATALENGTH	Word Length	Slot Length
0	32 bits	32
1	24 bits	32 if MR.IWS24 is zero 24 if MR.IWS24 is one
2	20 bits	
3	18 bits	
4	16 bits	16
5	16 bits compact stereo	
6	8 bits	8
7	8 bits compact stereo	

Warning: MR.IMCKMODE should only be written as one if the Master Clock frequency is strictly higher than the Serial Clock.

If a Master Clock output is not required, the GCLK_IISC generic clock is used as ISCK, by writing a zero to MR.IMCKMODE. Alternatively, if the frequency of the generic clock used is a multiple of the required ISCK frequency, the IMCK to ISCK divider can be used with the ratio defined by writing the MR.IMCKFS field.

The IWS pin is used as Word Select in I2S format and as Frame Synchronization in TDM format, as described in Section 30.6.4 and Section 30.6.5 respectively.

Figure 30-4. IISC Clocks Generation



30.6.7 Mono

When the Transmit Mono (TXMONO) in the Mode Register is set, data written to the left channel is duplicated to the right output channel. In TDM mode with more than two channels, numbered from 0, data written to the even-numbered channels is duplicated to the following odd-numbered channel.

When the Receive Mono (RXMONO) in the Mode Register is set, data received from the left channel is duplicated to the right channel. In TDM mode with more than two channels, numbered from 0, data received from the even-numbered channels is duplicated to the following odd-numbered channel.

30.6.8 Holding Registers

The IISC user interface includes a Receive Holding Register (RHR) and a Transmit Holding Register (THR). RHR and THR are used to access audio samples for all audio channels.

When a new data word is available in the RHR register, the Receive Ready bit (RXRDY) in the Status Register (SR) is set. Reading the RHR register will clear this bit.

A receive overrun condition occurs if a new data word becomes available before the previous data word has been read from the RHR register. Then, the Receive Overrun bit in the Status Register will be set and bit i of the RXORCH field in the Status Register is set, where i is the current receive channel number.

When the THR register is empty, the Transmit Ready bit (TXRDY) in the Status Register (SR) is set. Writing into the THR register will clear this bit.

A transmit underrun condition occurs if a new data word needs to be transmitted before it has been written to the THR register. Then, the Transmit Underrun bit in the Status Register will be

set and bit i of the TXORCH field in the Status Register is set, where i is the current transmit channel number. If the TXSAME bit in the Mode Register is zero, then a zero data word is transmitted in case of underrun. If MR.TXSAME is one, then the previous data word for the current transmit channel number is transmitted.

Data words are right-justified in the RHR and THR registers. For 16-bit compact stereo, the left sample uses bits 15 through 0 and the right sample uses bits 31 through 16 of the same data word. For 8-bit compact stereo, the left sample uses bits 7 through 0 and the right sample uses bits 15 through 8 of the same data word.

30.6.9 DMA Operation

The Receiver and the Transmitter can each be connected either to one single Peripheral DMA channel or to one Peripheral DMA channel per data channel. This is selected by writing to the MR.RXDMA and MR.TXDMA bits. If a single Peripheral DMA channel is selected, all data samples use IISC Receiver or Transmitter DMA channel 0.

The Peripheral DMA reads from the RHR register and writes to the RHR register for all audio channels, successively.

The Peripheral DMA transfers may use 32-bit word, 16-bit halfword, or 8-bit byte according to the value of the MR.DATALENGTH field.

30.6.10 Loop-back Mode

For debugging purposes, the IISC can be configured to loop back the Transmitter to the Receiver. Writing a one to the MR.LOOP bit will internally connect ISDO to ISDI, so that the transmitted data is also received. Writing a zero to MR.LOOP will restore the normal behavior with independent Receiver and Transmitter. As for other changes to the Receiver or Transmitter configuration, the IISC Receiver and Transmitter must be disabled before writing to the MR register to update MR.LOOP.

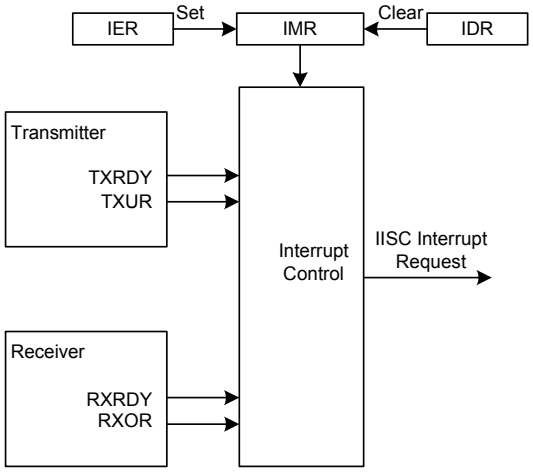
30.6.11 Interrupts

An IISC interrupt request can be triggered whenever one or several of the following bits are set in the Status Register (SR): Receive Ready (RXRDY), Receive Overrun (RXOR), Transmit Ready (TXRDY), or Transmit Underrun (TXOR).

The interrupt request will be generated if the corresponding bit in the Interrupt Mask Register (IMR) is set. Bits in IMR are set by writing a one to the corresponding bit in the Interrupt Enable Register (IER), and cleared by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The interrupt request remains active until the corresponding bit in SR is cleared by writing a one the corresponding bit in the Status Clear Register (SCR).

For debugging purposes, interrupt requests can be simulated by writing a one to the corresponding bit in the Status Set Register (SSR).

Figure 30-5. Interrupt Block Diagram



30.7 IISC Application Examples

The IISC can support several serial communication modes used in audio or high-speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the IISC are not listed here.

Figure 30-6. Audio Application Block Diagram

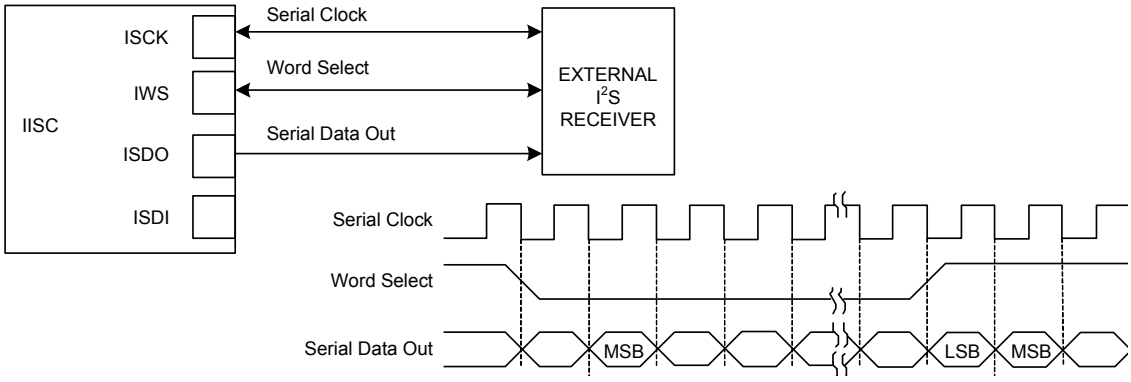


Figure 30-7. Codec Application Block Diagram

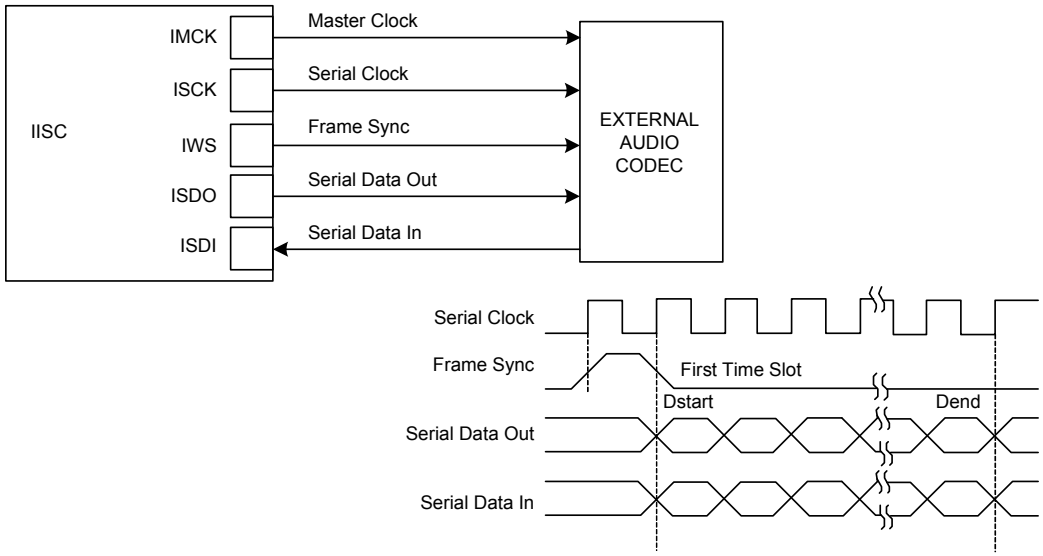
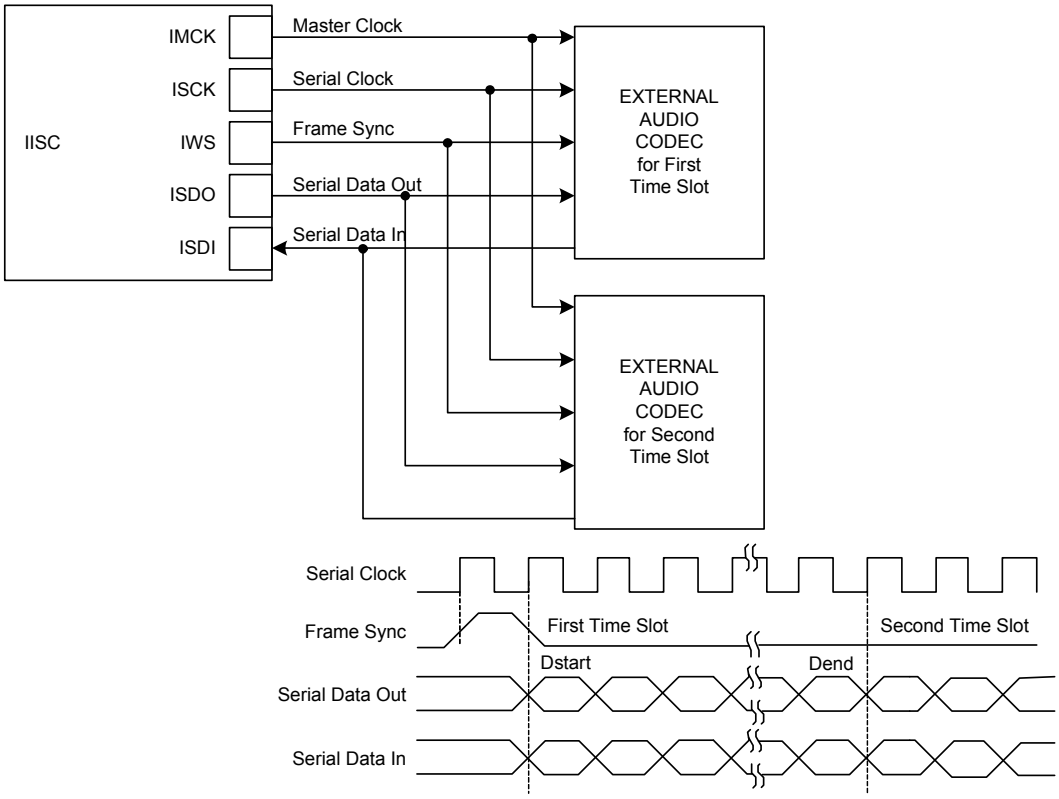


Figure 30-8. Time Slot Application Block Diagram



30.8 User Interface

Table 30-3. IISC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	0x00000000
0x04	Mode Register	MR	Read/Write	0x00000000
0x08	Status Register	SR	Read-only	0x00000000
0x0C	Status Clear Register	SCR	Write-only	0x00000000
0x10	Status Set Register	SSR	Write-only	0x00000000
0x14	Interrupt Enable Register	IER	Write-only	0x00000000
0x18	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x20	Receiver Holding Register	RHR	Read-only	0x00000000
0x24	Transmitter Holding Register	THR	Write-only	0x00000000
0x28	Version Register	VERSION	Read-only	-(1)
0x2C	Parameter Register	PARAMETER	Read-only	-(1)

Note: 1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

30.8.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
SWRST	-	TXDIS	TXEN	CKDIS	CKEN	RXDIS	RXEN

The Control Register should only be written to enable the IISC after the chosen configuration has been written to the Mode Register, in order to avoid unwanted glitches on the IWS, ISCK, and ISDO outputs. The proper sequence is to write the MR register, then write the CR register to enable the IISC, or to disable the IISC before writing a new value into MR.

- **SWRST: Software Reset**
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all the registers in the module. The module will be disabled after the reset.
  This bit always reads as zero.
- **TXDIS: Transmitter Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the IISC Transmitter. SR.TXEN will be cleared when the Transmitter is effectively stopped.
- **TXEN: Transmitter Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the IISC Transmitter, if TXDIS is not one. SR.TXEN will be set when the Transmitter is effectively started.
- **CKDIS: Clocks Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the IISC clocks generation.
- **CKEN: Clocks Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the IISC clocks generation, if CKDIS is not one.
- **RXDIS: Receiver Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the IISC Receiver. SR.TXEN will be cleared when the Transmitter is effectively stopped.
- **RXEN: Receiver Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the IISC Receiver, if RXDIS is not one. SR.RXEN will be set when the Receiver is effectively started.

30.8.2 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
IWS24	IMCKMODE	IMCKFS					

23	22	21	20	19	18	17	16
TDMFS		-	-	-	NBCHAN		

15	14	13	12	11	10	9	8
-	TXSAME	TXDMA	TXMONO		RXLOOP	RXDMA	RXMONO

7	6	5	4	3	2	1	0
FORMAT		-	DATALENGTH			-	MODE

The Mode Register should only be written when the IISC is stopped, in order to avoid unwanted glitches on the IWS, ISCK, and ISDO outputs. The proper sequence is to write the MR register, then write the CR register to enable the IISC, or to disable the IISC before writing a new value into MR.

- **IWS24: IWS TDM Slot Width**
  0: IWS slot is 32-bit wide for DATALENGTH=18/20/24-bit
  1: IWS slot is 24-bit wide for DATALENGTH=18/20/24-bit
  Refer to Table 30-2, “Slot Length,” on page 809.
- **IMCKMODE: Master Clock Mode**
  0: No Master Clock generated (generic clock is used as ISCK output)
  1: Master Clock generated (generic clock is used as IMCK output)
  Warning: if IMCK frequency is the same as ISCK, IMCKMODE should not be written as one. Refer to Section 30.6.6 ”Serial Clock and Word Select Generation” on page 808 and Table 30-2, “Slot Length,” on page 809.
- **IMCKFS: Master Clock to fs Ratio**
  Master Clock frequency is 8*(NBCHAN+1)*(IMCKFS+1) times the sample rate, i.e. IWS frequency:

Table 30-4. Master Clock to Sampling Frequency (fs) Ratio

	IMCKFS			
fs Ratio	2 channels	4 channels	6 channels	8 channels
16 fs	0	-	-	-
32 fs	1	0	-	-
48fs	2	-	0	-
64 fs	3	1	-	0
96fs	5	2	1	-
128 fs	7	3	-	1
192fs	11	5	3	2

Table 30-4. Master Clock to Sampling Frequency (fs) Ratio

fs Ratio	IMCKFS			
	2 channels	4 channels	6 channels	8 channels
256 fs	15	7	-	3
384 fs	23	11	7	5
512 fs	31	15	-	7
768 fs	47	23	15	11
1024 fs	63	31	-	15

- **TDMFS: TDM Frame Sync**

Table 30-5. TDM Frame Sync

TDMFS		Description
0	SLOT	IWS pulse is high for one time slot at beginning of frame
1	HALF	IWS pulse is high for half the time slots at beginning of frame, i.e. half the IWS period
3	BIT	IWS pulse is high for one bit period at beginning of frame, i.e. one ISCK period
3	-	Reserved

- **NBCHAN: Number of TDM Channels - 1**
  This field should be written with the number of TDM channels minus one
- **TXSAME: Transmit Data when Underrun**
  0: Zero sample transmitted when underrun
  1: Previous sample transmitted when underrun (in I2S mode only)
- **TXDMA: Single or multiple DMA Channels for Transmitter**
  0: Transmitter uses a single DMA channel for all audio channels
  1: Transmitter uses one DMA channel per audio channel
- **TXMONO: Transmit Mono**
  0: Stereo
  1: Mono, with left audio samples duplicated to right audio channel by the IISC
- **RXLOOP: Loop-back Test Mode**
  0: Normal mode
  1: ISDO output of IISC is internally connected to ISDI input
- **RXMONO: Receive Mono**
  0: Stereo
  1: Mono, with left audio samples duplicated to right audio channel by the IISC
- **RXDMA: Single or multiple DMA Channels for Receiver**
  0: Receiver uses a single DMA channel for all audio channels
  1: Receiver uses one DMA channel per audio channel-

- **FORMAT: I2S or TDM Format**

Table 30-6. Frame Format

FORMAT		Description
0	I2S	I^2S format, stereo with IWS low for left channel, and MSB of sample starting one ISCK period after IWS edge
1	LJ	Left-justified format, stereo with IWS high for left channel, and MSB of sample starting on IWS edge
2	TDM	TDM format, with (NBCHAN+1) channels, IWS high at beginning of first channel, and MSB of sample starting one ISCK period after IWS edge
3	TDMLJ	TDM format, left-justified, with (NBCHAN+1) channels, IWS high at beginning of first channel, and MSB of sample starting on IWS edge

- **DATALENGTH: Data Word Length**

Table 30-7. Data Word Length

DATALENGTH	Word Length	Comments
0	32 bits	
1	24 bits	
2	20 bits	
3	18 bits	
4	16 bits	
5	16 bits compact stereo	Left sample in bits 15 through 0 and right sample in bits 31 through 16 of the same word
6	8 bits	
7	8 bits compact stereo	Left sample in bits 7 through 0 and right sample in bits 15 through 8 of the same word

- **MODE: Mode**

Table 30-8. Mode

MODE		Comments
0	SLAVE	ISCK and IWS pin inputs used as Bit Clock and Word Select/Frame Sync.
1	MASTER	Bit Clock and Word Select/Frame Sync generated by IISC from GCLK_IISC and output to ISCK and IWS pins. GCLK_IISC is output as Master Clock on IMCK if MR.IMCKMODE is one.

30.8.3 Status Register

Name: SR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	TXURCH[7:4]			

23	22	21	20	19	18	17	16
TXURCH[3:0]				-	-	-	-

15	14	13	12	11	10	9	8
RXORCH							

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	TXEN	-	RXOR	RXRDY	RXEN

- **TXURCH: Transmit Underrun Channel**
  This field is cleared when SCR.TXUR is written to one
  Bit i of this field is set when a transmit underrun error occurred in channel i (i=0 for first channel of the frame)
- **RXORCH: Receive Overrun Channel**
  This field is cleared when SCR.RXOR is written to one
  Bit i of this field is set when a receive overrun error occurred in channel i (i=0 for first channel of the frame)
- **TXUR: Transmit Underrun**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when an underrun error occurs on the THR register or when the corresponding bit in SSR is written to one
- **TXRDY: Transmit Ready**
  This bit is cleared when data is written to THR
  This bit is set when the THR register is empty and can be written with new data to be transmitted
- **TXEN: Transmitter Enabled**
  This bit is cleared when the Transmitter is effectively disabled, following a CR.TXDIS or CR.SWRST request
  This bit is set when the Transmitter is effectively enabled, following a CR.TXEN request
- **RXOR: Receive Overrun**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when an overrun error occurs on the RHR register or when the corresponding bit in SSR is written to one
- **RXRDY: Receive Ready**
  This bit is cleared when the RHR register is read
  This bit is set when received data is present in the RHR register
- **RXEN: Receiver Enabled**
  This bit is cleared when the Receiver is effectively disabled, following a CR.RXDIS or CR.SWRST request
  This bit is set when the Receiver is effectively enabled, following a CR.RXEN request

30.8.4 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	TXURCH[7:4]			

23	22	21	20	19	18	17	16
TXURCH[3:0]				-	-	-	-

15	14	13	12	11	10	9	8
RXORCH							

7	6	5	4	3	2	1	0
-	TXUR	-	-	-	RXOR	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

30.8.5 Status Set Register

Name: SSR

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	TXURCH[7:4]			

23	22	21	20	19	18	17	16
TXURCH[3:0]				-	-	-	-

15	14	13	12	11	10	9	8
RXORCH							

7	6	5	4	3	2	1	0
-	TXUR	-	-	-	RXOR	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in SR.

30.8.6 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	-	-	RXOR	RXRDY	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

30.8.7 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	-	-	RXOR	RXRDY	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

30.8.8 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	-	-	RXOR	RXRDY	-

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

30.8.9 Receive Holding Register

Name: RHR

Access Type: Read-only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RHR[31:24]							

23	22	21	20	19	18	17	16
RHR[23:16]							

15	14	13	12	11	10	9	8
RHR[15:8]							

7	6	5	4	3	2	1	0
RHR[7:0]							

- **RHR: Received Word**
  This field is set by hardware to the last received data word. If MR.DATALENGTH specifies less than 32 bits, data shall be right-justified into the RHR field.

30.8.10 Transmit Holding Register

Name: THR

Access Type: Write-only

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
THR[31:24]							

23	22	21	20	19	18	17	16
THR[23:16]							

15	14	13	12	11	10	9	8
THR[15:8]							

7	6	5	4	3	2	1	0
THR[7:0]							

- **THR: Data Word to Be Transmitted**
  Next data word to be transmitted after the current word if TXRDY is not set. If MR.DATALENGTH specifies less than 32 bits, data shall be right-justified into the THR field.

30.8.11 Module Version

Name: VERSION

Access Type: Read-only

Offset: 0x28

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

30.8.12 Module Parameters

Name: PARAMETER

Access Type: Read-only

Offset: 0x2C

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Reserved. No functionality associated.

30.9 Module configuration

The specific configuration for each IISC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks according to the table in the System Bus Clock Connections section.

Table 30-9. Module configuration

Feature	IISC
Number of TDM channels	8 channels
Number of Peripheral DMA channels	8 channels

Table 30-10. Module clock name

Module name	Clock name	Clock name
IISC	CLK_IISC	Peripheral Bus clock from the PBA clock domain
	GCLK	The generic clock used for the IISC is GCLK11

Table 30-11. Register Reset Values

Register	Reset Value
VERSION	0x00000200

31. Timer/Counter (TC)

Rev: 2.2.3.3

31.1 Features

- **Three 16-bit Timer Counter channels**
- **A wide range of functions including:**
  - **Frequency measurement**
  - **Event counting**
  - **Interval measurement**
  - **Pulse generation**
  - **Delay timing**
  - **Pulse width modulation**
  - **Up/down capabilities**
- **Each channel is user-configurable and contains:**
  - **Three external clock inputs**
  - **Five internal clock inputs**
  - **Two multi-purpose input/output signals**
- **Internal interrupt signal**
- **Two global registers that act on all three TC channels**
- **Peripheral event input on all A/B lines in capture mode**

31.2 Overview

The Timer Counter (TC) includes three identical 16-bit Timer Counter channels.

Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing, and pulse width modulation.

Each channel has three external clock inputs, five internal clock inputs, and two multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

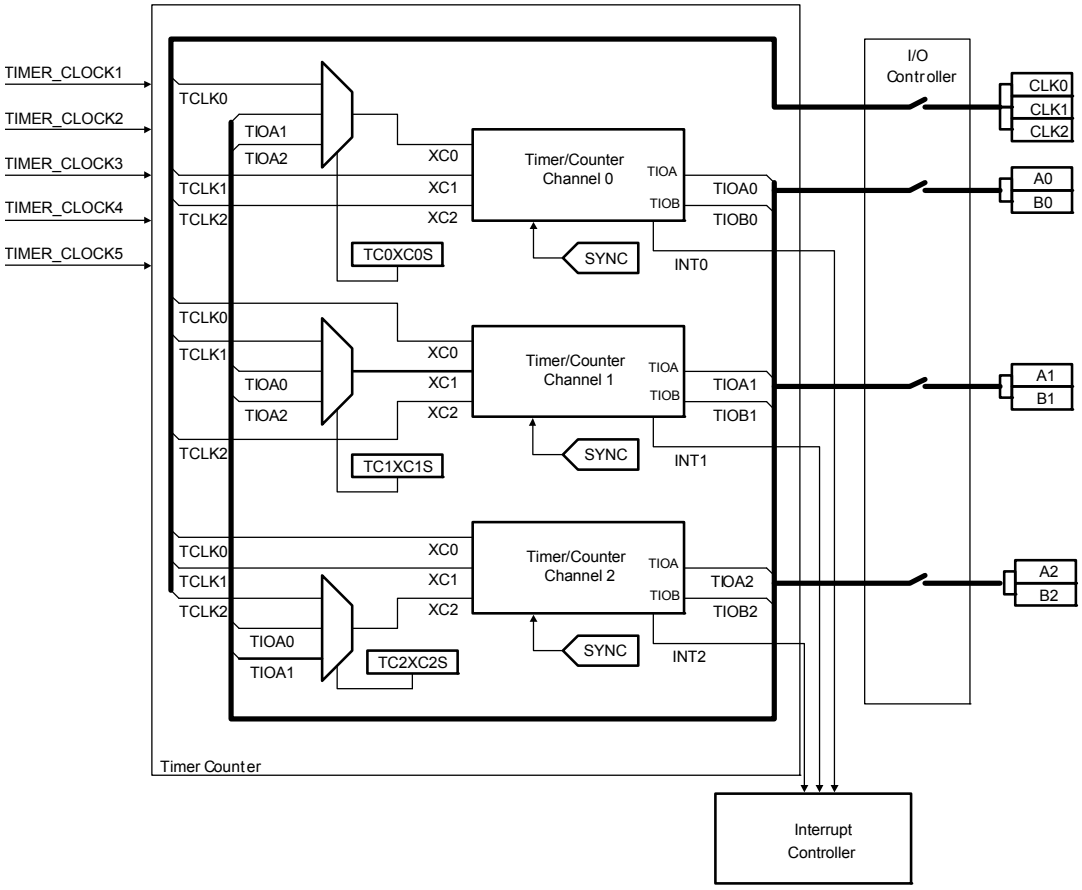
The TC block has two global registers which act upon all three TC channels.

The Block Control Register (BCR) allows the three channels to be started simultaneously with the same instruction.

The Block Mode Register (BMR) defines the external clock inputs for each channel, allowing them to be chained.

31.3 Block Diagram

Figure 31-1. TC Block Diagram



31.4 I/O Lines Description

Table 31-1. I/O Lines Description

Pin Name	Description	Type
CLK0-CLK2	External Clock Input	Input
A0-A2	I/O Line A	Input/Output
B0-B2	I/O Line B	Input/Output

31.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

31.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with I/O lines. The user must first program the I/O Controller to assign the TC pins to their peripheral functions.

When using the TIOA/TIOB lines as inputs the user must make sure that no peripheral events are generated on the line. Refer to the Peripheral Event System chapter for details.

31.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the TC, the TC will stop functioning and resume operation after the system wakes up from sleep mode.

31.5.3 Clocks

The clock for the TC bus interface (CLK_TC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the TC before disabling the clock, to avoid freezing the TC in an undefined state.

31.5.4 Interrupts

The TC interrupt request line is connected to the interrupt controller. Using the TC interrupt requires the interrupt controller to be programmed first.

31.5.5 Peripheral Events

The TC peripheral events are connected via the Peripheral Event System. Refer to the Peripheral Event System chapter for details.

31.5.6 Debug Operation

The Timer Counter clocks are frozen during debug operation, unless the OCD system keeps peripherals running in debug operation.

31.6 Functional Description

31.6.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Figure 31-3 on page 847.

31.6.1.1 Channel I/O Signals

As described in Figure 31-1 on page 831, each Channel has the following I/O signals.

Table 31-2. Channel I/O Signals Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture mode: Timer Counter Input Waveform mode: Timer Counter Output
	TIOB	Capture mode: Timer Counter Input Waveform mode: Timer Counter Input/Output
	INT	Interrupt Signal Output
	SYNC	Synchronization Input Signal

31.6.1.2 16-bit counter

Each channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the Counter Overflow Status bit in the Channel n Status Register (SRn.COVFS) is set.

The current value of the counter is accessible in real time by reading the Channel n Counter Value Register (CVn). The counter can be reset by a trigger. In this case, the counter value passes to 0x0000 on the next valid edge of the selected clock.

31.6.1.3 Clock selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the configurable I/O signals A0, A1 or A2 for chaining by writing to the BMR register. See Figure 31-2 on page 833.

Each channel can independently select an internal or external clock source for its counter:

- Internal clock signals: TIMER_CLOCK1, TIMER_CLOCK2, TIMER_CLOCK3, TIMER_CLOCK4, TIMER_CLOCK5. See the Module Configuration Chapter for details about the connection of these clock sources.
- External clock signals: XC0, XC1 or XC2. See the Module Configuration Chapter for details about the connection of these clock sources.

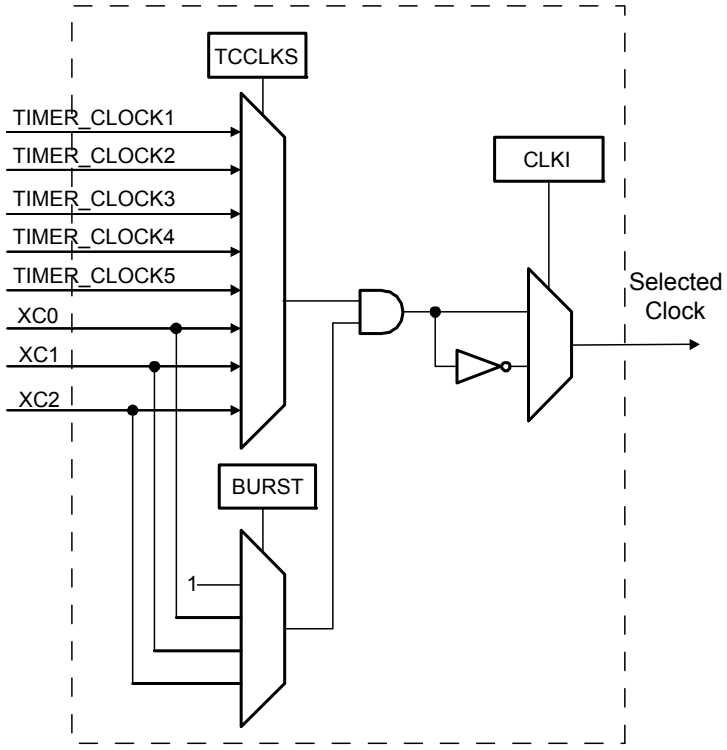
This selection is made by the Clock Selection field in the Channel n Mode Register (CMRn.TCCLKS).

The selected clock can be inverted with the Clock Invert bit in CMRn (CMRn.CLKI). This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The Burst Signal Selection field in the CMRn register (CMRn.BURST) defines this signal.

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the CLK_TC period. The external clock frequency must be at least 2.5 times lower than the CLK_TC.

Figure 31-2. Clock Selection

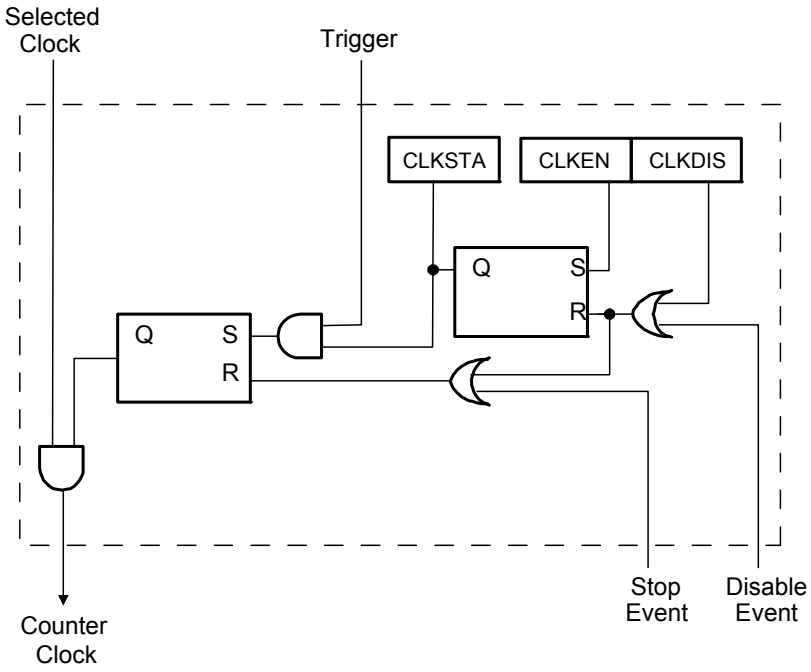


31.6.1.4 Clock control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 31-3 on page 834.

- The clock can be enabled or disabled by the user by writing to the Counter Clock Enable/Disable Command bits in the Channel n Clock Control Register (CCRn.CLKEN and CCRn.CLKDIS). In Capture mode it can be disabled by an RB load event if the Counter Clock Disable with RB Loading bit in CMRn is written to one (CMRn.LDBDIS). In Waveform mode, it can be disabled by an RC Compare event if the Counter Clock Disable with RC Compare bit in CMRn is written to one (CMRn.CPCDIS). When disabled, the start or the stop actions have no effect: only a CLKEN command in CCRn can re-enable the clock. When the clock is enabled, the Clock Enabling Status bit is set in SRn (SRn.CLKSTA).
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. In Capture mode the clock can be stopped by an RB load event if the Counter Clock Stopped with RB Loading bit in CMRn is written to one (CMRn.LDBSTOP). In Waveform mode it can be stopped by an RC compare event if the Counter Clock Stopped with RC Compare bit in CMRn is written to one (CMRn.CPCSTOP). The start and the stop commands have effect only if the clock is enabled.

Figure 31-3. Clock Control



31.6.1.5 TC operating modes

Each channel can independently operate in two different modes:

- Capture mode provides measurement on signals.
- Waveform mode provides wave generation.

The TC operating mode selection is done by writing to the Wave bit in the CCRn register (CCRn.WAVE).

In Capture mode, TIOA and TIOB are configured as inputs.

In Waveform mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

31.6.1.6 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

The following triggers are common to both modes:

- Software Trigger: each channel has a software trigger, available by writing a one to the Software Trigger Command bit in CCRn (CCRn.SWTRG).
- SYNC: each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing a one to the Synchro Command bit in the BCR register (BCR.SYNC).
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if the RC Compare Trigger Enable bit in CMRn (CMRn.CPCTRG) is written to one.

The channel can also be configured to have an external trigger. In Capture mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform mode, an external event can be programmed to be one of the following signals: TIOB, XC0, XC1, or XC2. This external event can then be programmed to perform a trigger by writing a one to the External Event Trigger Enable bit in CMRn (CMRn.ENETRG).

If an external trigger is used, the duration of the pulses must be longer than the CLK_TC period in order to be detected.

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

31.6.1.7 Peripheral events on TIOA/TIOB inputs

The TIOA*/TIOB* input lines are ored internally with peripheral events from the Peripheral Event System. To capture using events the user must ensure that the corresponding pin functions for the TIOA*/TIOB* line are disabled. When capturing on the external TIOA*/TIOB* pin the user must ensure that no peripheral events are generated on this pin.

31.6.2 Capture Operating Mode

This mode is entered by writing a zero to the CMRn.WAVE bit.

Capture mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 31-4 on page 837 shows the configuration of the TC channel when programmed in Capture mode.

31.6.2.1 Capture registers A and B

Registers A and B (RA and RB) are used as capture registers. This means that they can be loaded with the counter value when a programmable event occurs on the signal TIOA.

The RA Loading Selection field in CMRn (CMRn.LDRA) defines the TIOA edge for the loading of the RA register, and the RB Loading Selection field in CMRn (CMRn.LDRB) defines the TIOA edge for the loading of the RB register.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

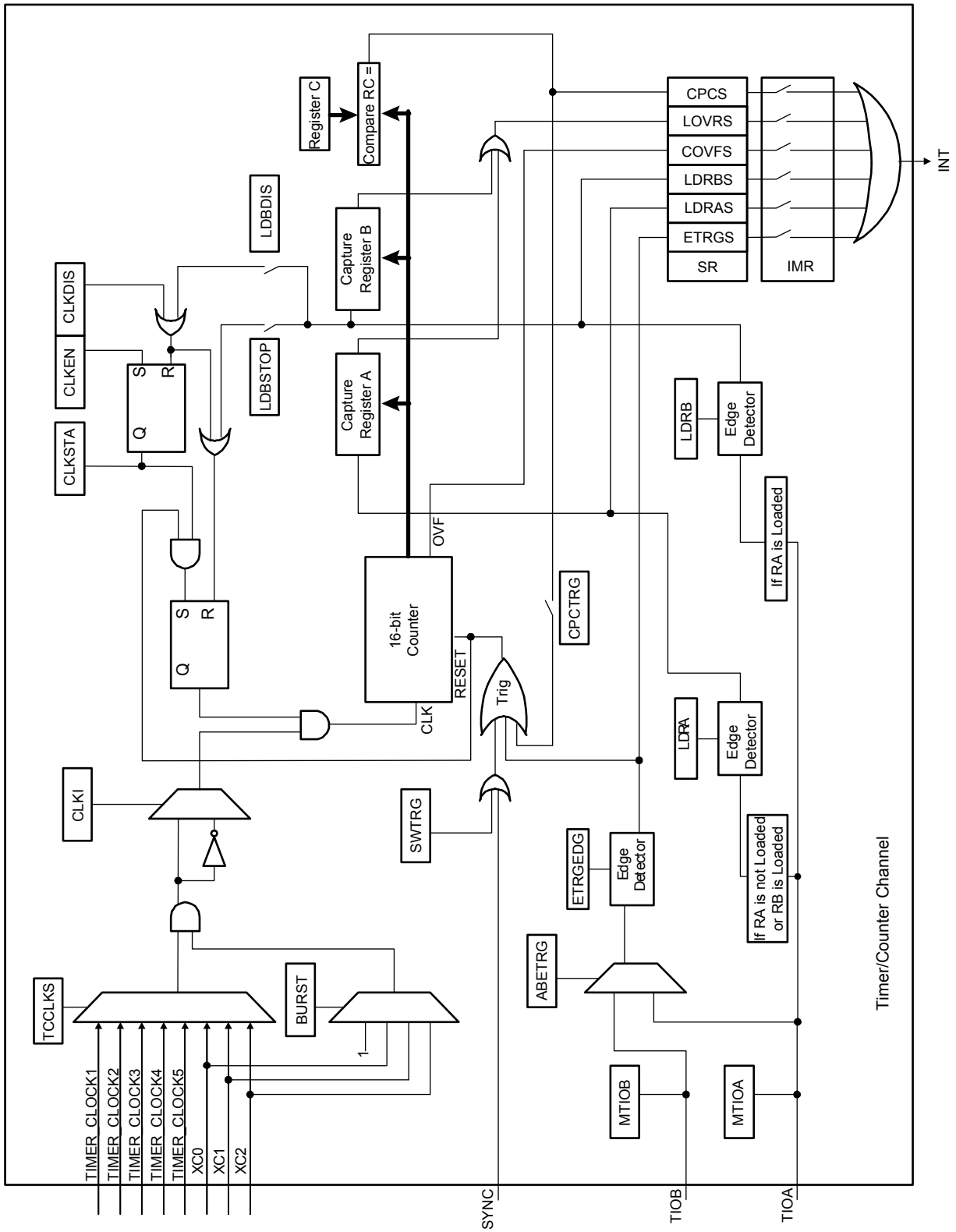
Loading RA or RB before the read of the last value loaded sets the Load Overrun Status bit in SRn (SRn.LOVRS). In this case, the old value is overwritten.

31.6.2.2 Trigger conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The TIOA or TIOB External Trigger Selection bit in CMRn (CMRn.ABETRG) selects TIOA or TIOB input signal as an external trigger. The External Trigger Edge Selection bit in CMRn (CMRn.ETREDG) defines the edge (rising, falling or both) detected to generate an external trigger. If CMRn.ETRGEDG is zero (none), the external trigger is disabled.

Figure 31-4. Capture Mode



31.6.3 Waveform Operating Mode

Waveform operating mode is entered by writing a one to the CMRn.WAVE bit.

In Waveform operating mode the TC channel generates one or two PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event.

Figure 31-5 on page 839 shows the configuration of the TC channel when programmed in Waveform operating mode.

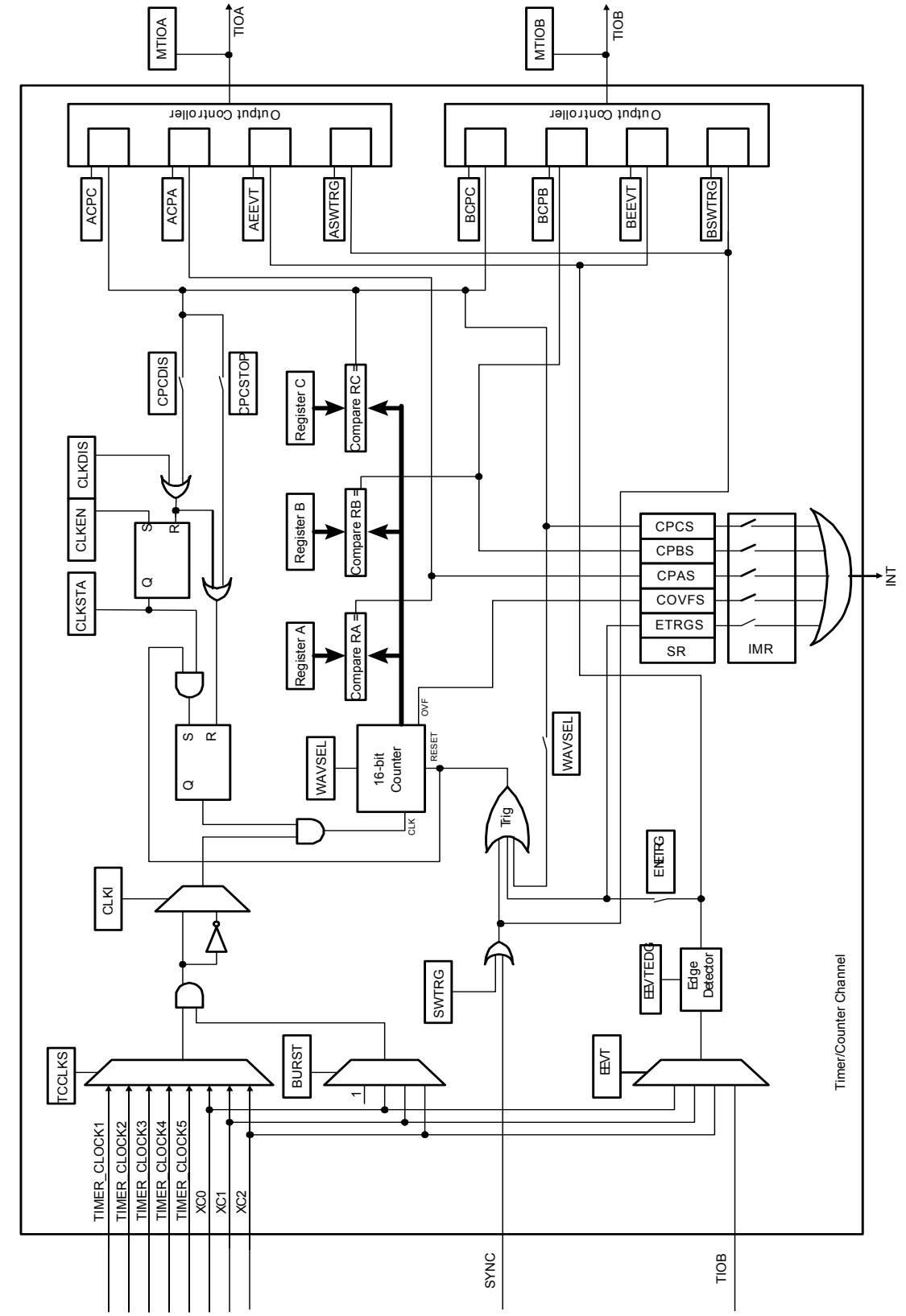
31.6.3.1 Waveform selection

Depending on the Waveform Selection field in CMRn (CMRn.WAVSEL), the behavior of CVn varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

Figure 31-5. Waveform Mode



31.6.3.2 WAVSEL = 0

When CMRn.WAVSEL is zero, the value of CVn is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of CVn is reset. Incrementation of CVn starts again and the cycle continues. See Figure 31-6 on page 840.

An external event trigger or a software trigger can reset the value of CVn. It is important to note that the trigger may occur at any time. See Figure 31-7 on page 841.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CMRn.CPCSTOP = 1) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 31-6. WAVSEL= 0 Without Trigger

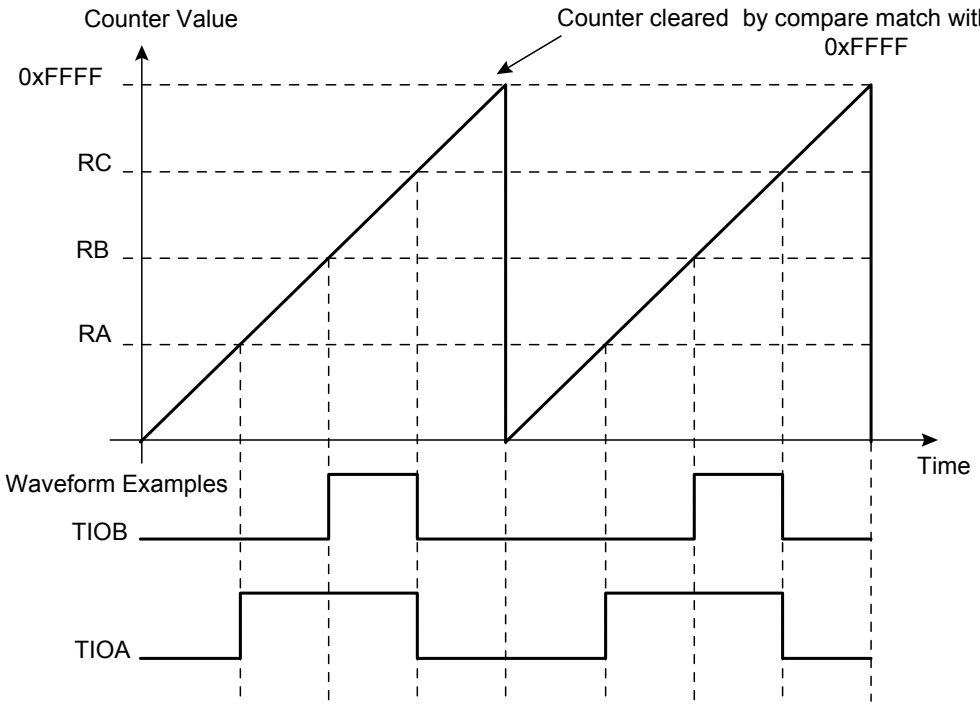
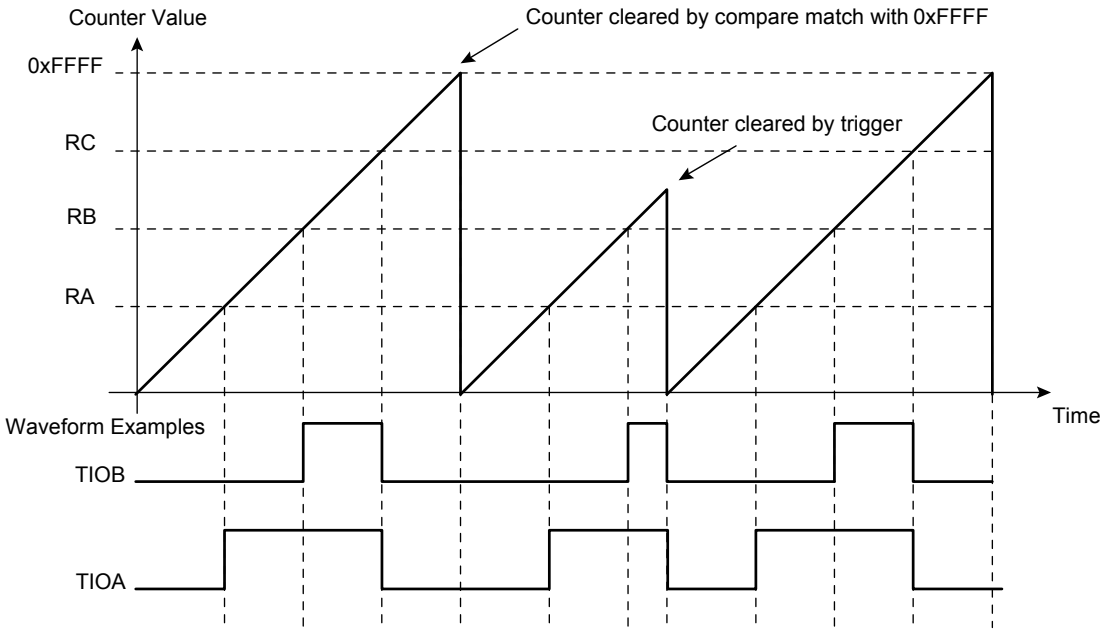


Figure 31-7. WAVSEL= 0 With Trigger



31.6.3.3 *WAVSEL = 2*

When CMRn.WAVSEL is two, the value of CVn is incremented from zero to the value of RC, then automatically reset on a RC Compare. Once the value of CVn has been reset, it is then incremented and so on. See Figure 31-8 on page 842.

It is important to note that CVn can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 31-9 on page 842.

In addition, RC Compare can stop the counter clock (CMRn.CPCSTOP) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 31-8. WAVSEL = 2 Without Trigger

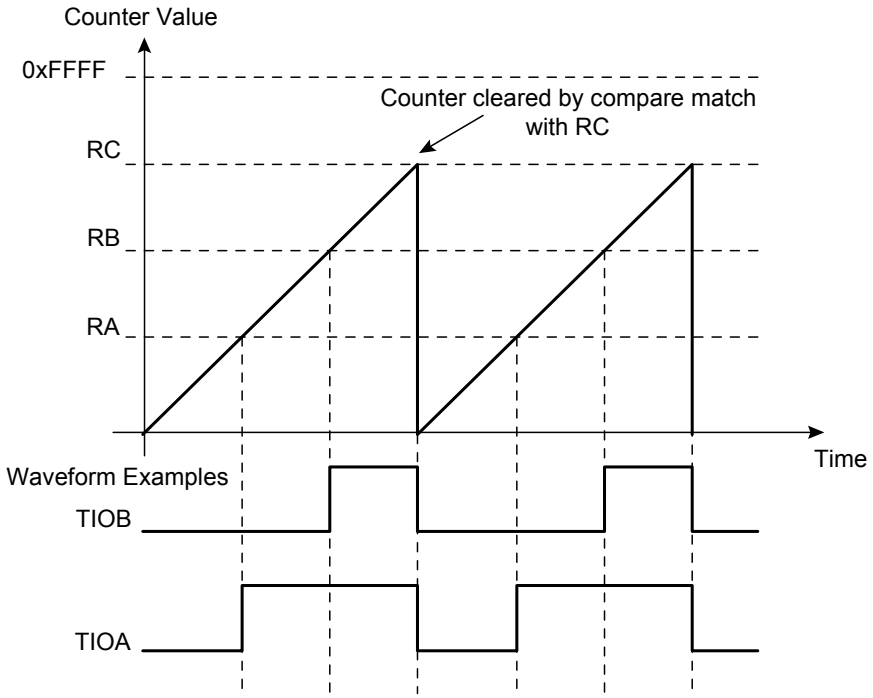
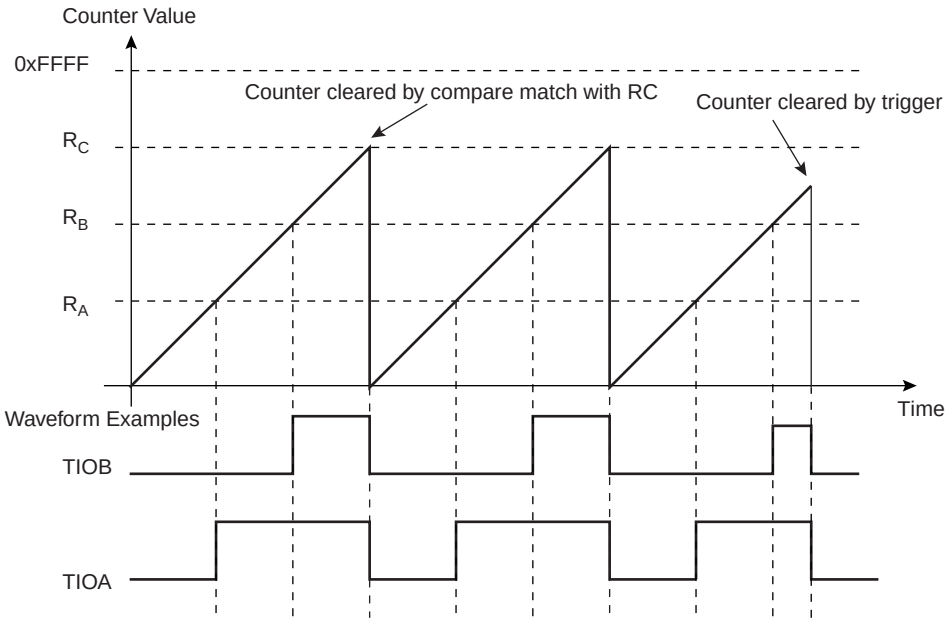


Figure 31-9. WAVSEL = 2 With Trigger



31.6.3.4 WAVSEL = 1

When CMRn.WAVSEL is one, the value of CVn is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of CVn is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 31-10 on page 843.

A trigger such as an external event or a software trigger can modify CVn at any time. If a trigger occurs while CVn is incrementing, CVn then decrements. If a trigger is received while CVn is decrementing, CVn then increments. See Figure 31-11 on page 843.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CMRn.CPCSTOP = 1) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 31-10. WAVSEL = 1 Without Trigger

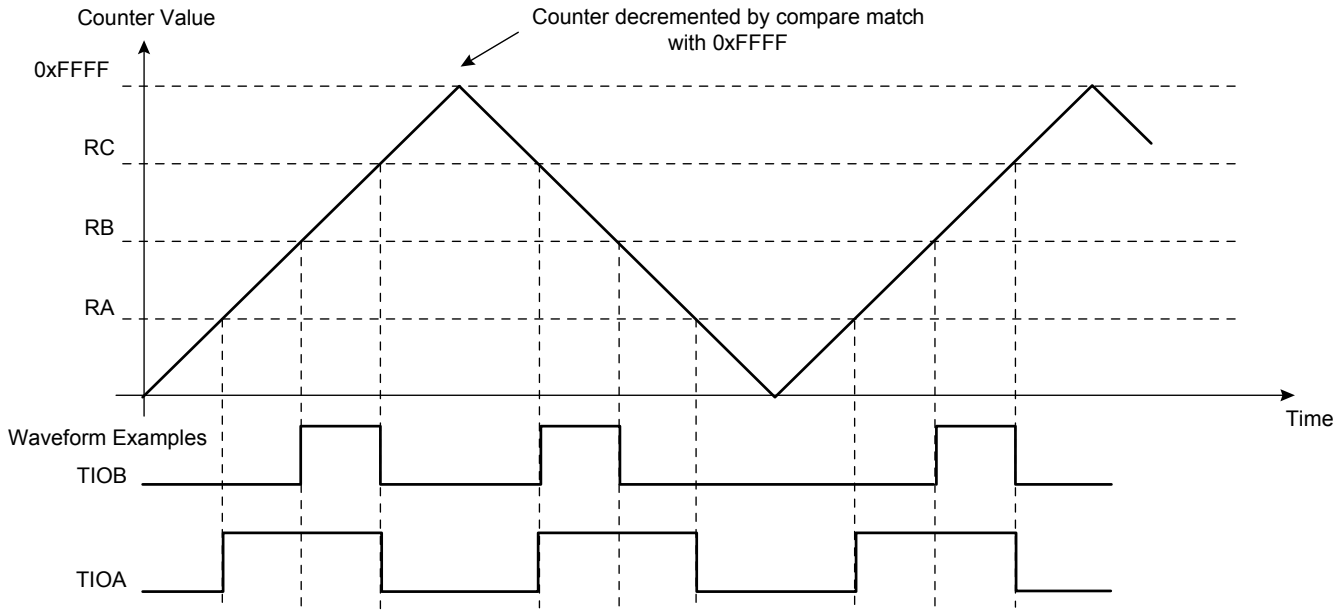
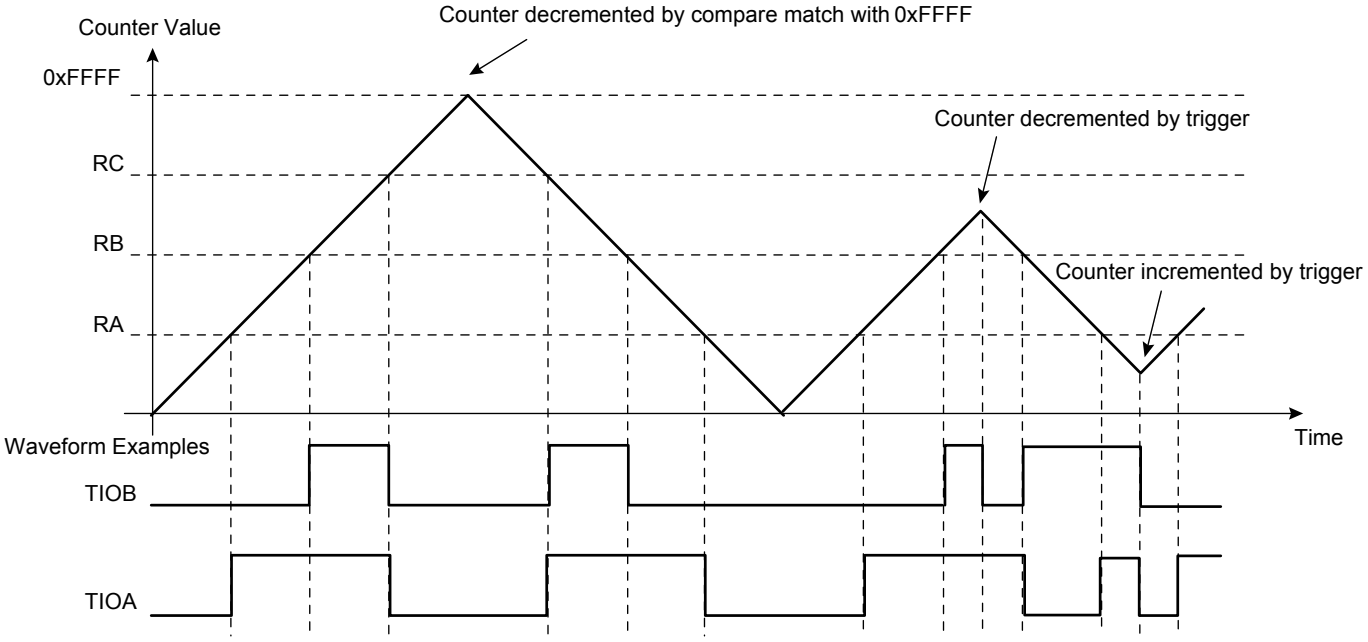


Figure 31-11. WAVSEL = 1 With Trigger



31.6.3.5 WAVSEL = 3

When CMRn.WAVSEL is three, the value of CVn is incremented from zero to RC. Once RC is reached, the value of CVn is decremented to zero, then re-incremented to RC and so on. See Figure 31-12 on page 844.

A trigger such as an external event or a software trigger can modify CVn at any time. If a trigger occurs while CVn is incrementing, CVn then decrements. If a trigger is received while CVn is decrementing, CVn then increments. See Figure 31-13 on page 845.

RC Compare can stop the counter clock (CMRn.CPCSTOP = 1) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 31-12. WAVSEL = 3 Without Trigger

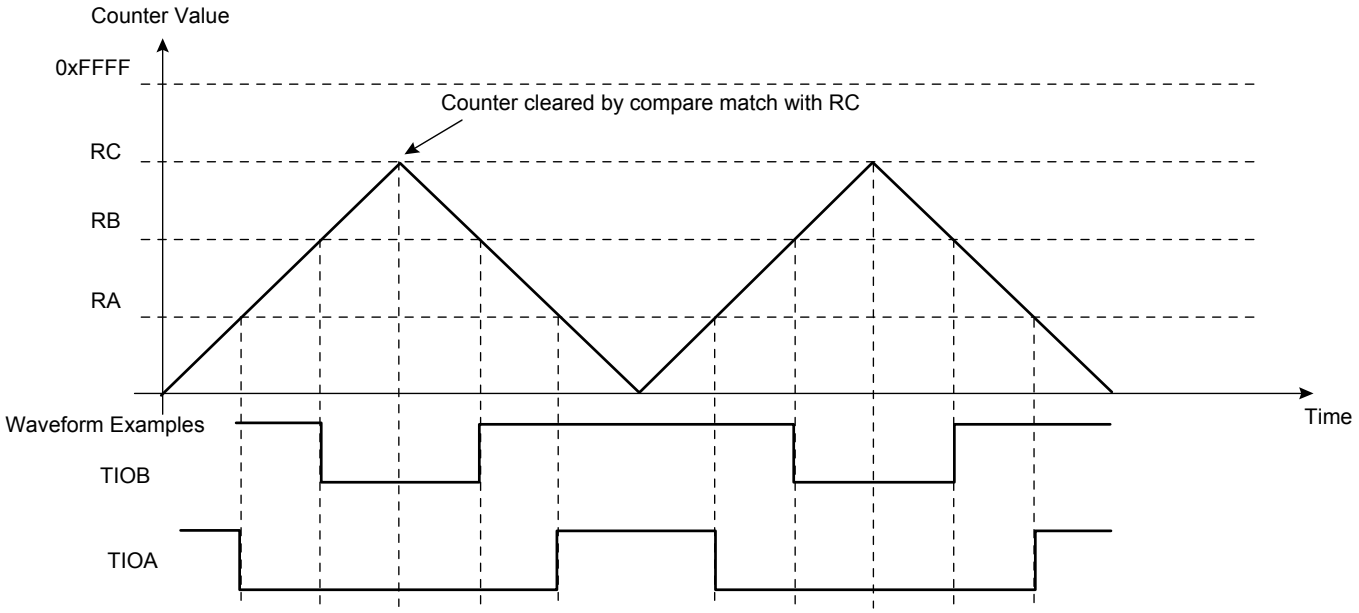
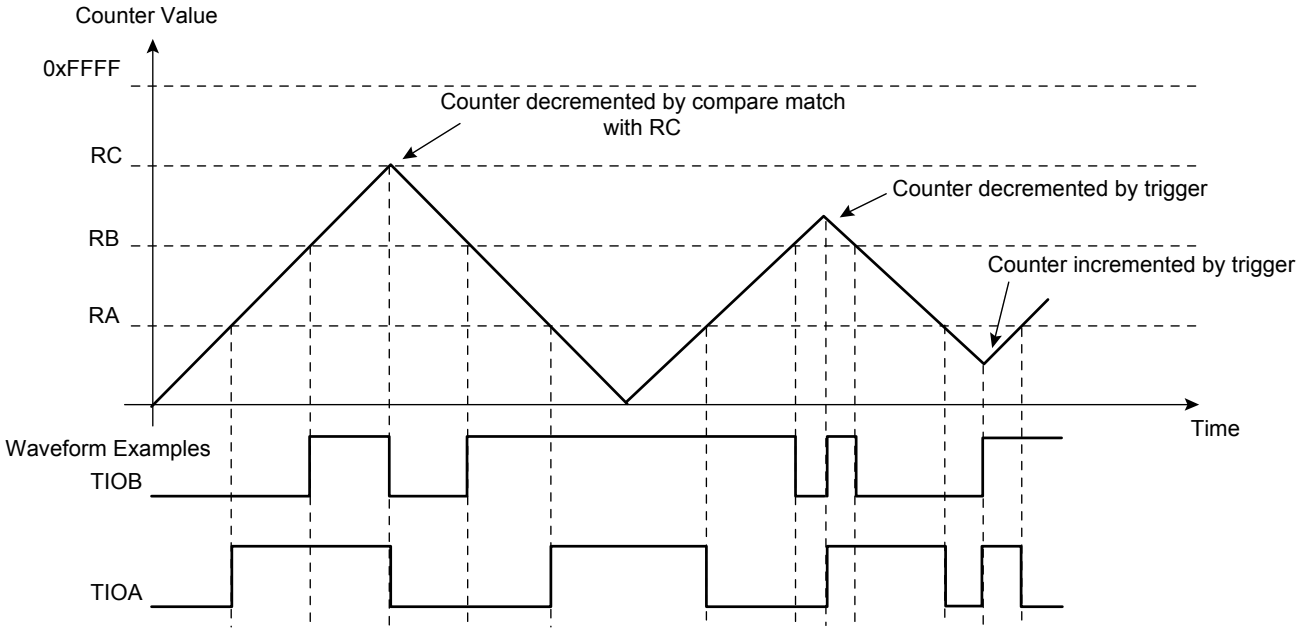


Figure 31-13. WAVSEL = 3 With Trigger



31.6.3.6 External event/trigger conditions

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The External Event Selection field in CMRn (CMRn.EEVT) selects the external trigger. The External Event Edge Selection field in CMRn (CMRn.EEVTEDG) defines the trigger edge for each of the possible external triggers (rising, falling or both). If CMRn.EEVTEDG is written to zero, no external event is defined.

If TIOB is defined as an external event signal (CMRn.EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by writing a one to the CMRn.ENETRG bit.

As in Capture mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the CMRn.WAVSEL field.

31.6.3.7 Output controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB:

- software trigger
- external event
- RC compare

RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the following fields in CMRn:

- RC Compare Effect on TIOB (CMRn.BCPC)

- RB Compare Effect on TIOB (CMRn.BCPB)
- RC Compare Effect on TIOA (CMRn.ACPC)
- RA Compare Effect on TIOA (CMRn.ACPA)

31.7 User Interface

Table 31-3. TC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Channel 0 Control Register	CCR0	Write-only	0x00000000
0x04	Channel 0 Mode Register	CMR0	Read/Write	0x00000000
0x10	Channel 0 Counter Value	CV0	Read-only	0x00000000
0x14	Channel 0 Register A	RA0	Read/Write(1)	0x00000000
0x18	Channel 0 Register B	RB0	Read/Write(1)	0x00000000
0x1C	Channel 0 Register C	RC0	Read/Write	0x00000000
0x20	Channel 0 Status Register	SR0	Read-only	0x00000000
0x24	Interrupt Enable Register	IER0	Write-only	0x00000000
0x28	Channel 0 Interrupt Disable Register	IDR0	Write-only	0x00000000
0x2C	Channel 0 Interrupt Mask Register	IMR0	Read-only	0x00000000
0x40	Channel 1 Control Register	CCR1	Write-only	0x00000000
0x44	Channel 1 Mode Register	CMR1	Read/Write	0x00000000
0x50	Channel 1 Counter Value	CV1	Read-only	0x00000000
0x54	Channel 1 Register A	RA1	Read/Write(1)	0x00000000
0x58	Channel 1 Register B	RB1	Read/Write(1)	0x00000000
0x5C	Channel 1 Register C	RC1	Read/Write	0x00000000
0x60	Channel 1 Status Register	SR1	Read-only	0x00000000
0x64	Channel 1 Interrupt Enable Register	IER1	Write-only	0x00000000
0x68	Channel 1 Interrupt Disable Register	IDR1	Write-only	0x00000000
0x6C	Channel 1 Interrupt Mask Register	IMR1	Read-only	0x00000000
0x80	Channel 2 Control Register	CCR2	Write-only	0x00000000
0x84	Channel 2 Mode Register	CMR2	Read/Write	0x00000000
0x90	Channel 2 Counter Value	CV2	Read-only	0x00000000
0x94	Channel 2 Register A	RA2	Read/Write(1)	0x00000000
0x98	Channel 2 Register B	RB2	Read/Write(1)	0x00000000
0x9C	Channel 2 Register C	RC2	Read/Write	0x00000000
0xA0	Channel 2 Status Register	SR2	Read-only	0x00000000
0xA4	Channel 2 Interrupt Enable Register	IER2	Write-only	0x00000000
0xA8	Channel 2 Interrupt Disable Register	IDR2	Write-only	0x00000000
0xAC	Channel 2 Interrupt Mask Register	IMR2	Read-only	0x00000000
0xC0	Block Control Register	BCR	Write-only	0x00000000
0xC4	Block Mode Register	BMR	Read/Write	0x00000000
0xF8	Features Register	FEATURES	Read-only	-(2)
0xFC	Version Register	VERSION	Read-only	-(2)

Notes:
1. Read-only if CMRn.WAVE is zero.
2. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

31.7.1 Channel Control Register

Name: CCR

Access Type: Write-only

Offset: 0x00 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	SWTRG	CLKDIS	CLKEN

- **SWTRG: Software Trigger Command**
  1: Writing a one to this bit will perform a software trigger: the counter is reset and the clock is started.
  0: Writing a zero to this bit has no effect.
- **CLKDIS: Counter Clock Disable Command**
  1: Writing a one to this bit will disable the clock.
  0: Writing a zero to this bit has no effect.
- **CLKEN: Counter Clock Enable Command**
  1: Writing a one to this bit will enable the clock if CLKDIS is not one.
  0: Writing a zero to this bit has no effect.

31.7.2 Channel Mode Register: Capture Mode

Name: CMR

Access Type: Read/Write

Offset: 0x04 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	LDRB		LDRA	

15	14	13	12	11	10	9	8
WAVE	CPCTRG	-	-	-	ABETRG	ETRGEDG	

7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- **LDRB: RB Loading Selection**

LDRB	Edge
0	none
1	rising edge of TIOA
2	falling edge of TIOA
3	each edge of TIOA

- **LDRA: RA Loading Selection**

LDRA	Edge
0	none
1	rising edge of TIOA
2	falling edge of TIOA
3	each edge of TIOA

- **WAVE**
  1: Capture mode is disabled (Waveform mode is enabled).
  0: Capture mode is enabled.
- **CPCTRG: RC Compare Trigger Enable**
  1: RC Compare resets the counter and starts the counter clock.
  0: RC Compare has no effect on the counter and its clock.
- **ABETRG: TIOA or TIOB External Trigger Selection**
  1: TIOA is used as an external trigger.

0: TIOB is used as an external trigger.

- **ETRGEDG: External Trigger Edge Selection**

ETRGEDG	Edge
0	none
1	rising edge
2	falling edge
3	each edge

- **LDBDIS: Counter Clock Disable with RB Loading**
  1: Counter clock is disabled when RB loading occurs.
  0: Counter clock is not disabled when RB loading occurs.
- **LDBSTOP: Counter Clock Stopped with RB Loading**
  1: Counter clock is stopped when RB loading occurs.
  0: Counter clock is not stopped when RB loading occurs.
- **BURST: Burst Signal Selection**

BURST	Burst Signal Selection
0	The clock is not gated by an external signal
1	XC0 is ANDed with the selected clock
2	XC1 is ANDed with the selected clock
3	XC2 is ANDed with the selected clock

- **CLKI: Clock Invert**
  1: The counter is incremented on falling edge of the clock.
  0: The counter is incremented on rising edge of the clock.
- **TCCLKS: Clock Selection**

TCCLKS	Clock Selected
0	TIMER_CLOCK1
1	TIMER_CLOCK2
2	TIMER_CLOCK3
3	TIMER_CLOCK4
4	TIMER_CLOCK5
5	XC0
6	XC1
7	XC2

31.7.3 Channel Mode Register: Waveform Mode

Name: CMR

Access Type: Read/Write

Offset: 0x04 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	

23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	

15	14	13	12	11	10	9	8
WAVE	WAVSEL		ENETRG	EEVT		EEVTEDG	

7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

- **BSWTRG: Software Trigger Effect on TIOB**

BSWTRG	Effect
0	none
1	set
2	clear
3	toggle

- **BEEVT: External Event Effect on TIOB**

BEEVT	Effect
0	none
1	set
2	clear
3	toggle

- **BCPC: RC Compare Effect on TIOB**

BCPC	Effect
0	none
1	set
2	clear
3	toggle

- **BCPB: RB Compare Effect on TIOB**

BCPB	Effect
0	none
1	set
2	clear
3	toggle

- **ASWTRG: Software Trigger Effect on TIOA**

ASWTRG	Effect
0	none
1	set
2	clear
3	toggle

- **AEEVT: External Event Effect on TIOA**

AEEVT	Effect
0	none
1	set
2	clear
3	toggle

- **ACPC: RC Compare Effect on TIOA**

ACPC	Effect
0	none
1	set
2	clear
3	toggle

- **ACPA: RA Compare Effect on TIOA**

ACPA	Effect
0	none
1	set
2	clear
3	toggle

- **WAVE**
  1: Waveform mode is enabled.
  0: Waveform mode is disabled (Capture mode is enabled).
- **WAVSEL: Waveform Selection**

WAVSEL	Effect
0	UP mode without automatic trigger on RC Compare
1	UPDOWN mode without automatic trigger on RC Compare
2	UP mode with automatic trigger on RC Compare
3	UPDOWN mode with automatic trigger on RC Compare

- **ENETRG: External Event Trigger Enable**
  1: The external event resets the counter and starts the counter clock.
  0: The external event has no effect on the counter and its clock. In this case, the selected external event only controls the TIOA output.
- **EEVT: External Event Selection**

EEVT	Signal selected as external event	TIOB Direction
0	TIOB	input(1)
1	XC0	output
2	XC1	output
3	XC2	output

Note: 1. *If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms and subsequently no IRQs*.

- **EEVTEDG: External Event Edge Selection**

EEVTEDG	Edge
0	none
1	rising edge
2	falling edge
3	each edge

- **CPCDIS: Counter Clock Disable with RC Compare**
  1: Counter clock is disabled when counter reaches RC.
  0: Counter clock is not disabled when counter reaches RC.

- **CPCSTOP: Counter Clock Stopped with RC Compare**
  1: Counter clock is stopped when counter reaches RC.
  0: Counter clock is not stopped when counter reaches RC.
- **BURST: Burst Signal Selection**

BURST	Burst Signal Selection
0	The clock is not gated by an external signal.
1	XC0 is ANDed with the selected clock.
2	XC1 is ANDed with the selected clock.
3	XC2 is ANDed with the selected clock.

- **CLKI: Clock Invert**
  1: Counter is incremented on falling edge of the clock.
  0: Counter is incremented on rising edge of the clock.
- **TCCLKS: Clock Selection**

TCCLKS	Clock Selected
0	TIMER_CLOCK1
1	TIMER_CLOCK2
2	TIMER_CLOCK3
3	TIMER_CLOCK4
4	TIMER_CLOCK5
5	XC0
6	XC1
7	XC2

31.7.4 Channel Counter Value Register

Name: CV

Access Type: Read-only

Offset: 0x10 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
CV[15:8]							

7	6	5	4	3	2	1	0
CV[7:0]							

- **CV: Counter Value**
  CV contains the counter value in real time.

31.7.5 Channel Register A

Name: RA

Access Type: Read-only if CMRn.WAVE = 0, Read/Write if CMRn.WAVE = 1

Offset: 0x14 + n * 0X40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
RA[15:8]							

7	6	5	4	3	2	1	0
RA[7:0]							

- **RA: Register A**
  RA contains the Register A value in real time.

31.7.6 Channel Register B

Name: RB

Access Type: Read-only if CMRn.WAVE = 0, Read/Write if CMRn.WAVE = 1

Offset: 0x18 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
RB[15:8]							

7	6	5	4	3	2	1	0
RB[7:0]							

- **RB: Register B**
  RB contains the Register B value in real time.

31.7.7 Channel Register C

Name: RC

Access Type: Read/Write

Offset: 0x1C + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
RC[15:8]							

7	6	5	4	3	2	1	0
RC[7:0]							

- **RC: Register C**
  RC contains the Register C value in real time.

31.7.8 Channel Status Register

Name: SR

Access Type: Read-only

Offset: 0x20 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	MTIOB	MTIOA	CLKSTA
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

Note: Reading the Status Register will also clear the interrupt bit for the corresponding interrupts.

- **MTIOB: TIOB Mirror**
  1: TIOB is high. If CMRn.WAVE is zero, this means that TIOB pin is high. If CMRn.WAVE is one, this means that TIOB is driven high.
  0: TIOB is low. If CMRn.WAVE is zero, this means that TIOB pin is low. If CMRn.WAVE is one, this means that TIOB is driven low.
- **MTIOA: TIOA Mirror**
  1: TIOA is high. If CMRn.WAVE is zero, this means that TIOA pin is high. If CMRn.WAVE is one, this means that TIOA is driven high.
  0: TIOA is low. If CMRn.WAVE is zero, this means that TIOA pin is low. If CMRn.WAVE is one, this means that TIOA is driven low.
- **CLKSTA: Clock Enabling Status**
  1: This bit is set when the clock is enabled.
  0: This bit is cleared when the clock is disabled.
- **ETRGS: External Trigger Status**
  1: This bit is set when an external trigger has occurred.
  0: This bit is cleared when the SR register is read.
- **LDRBS: RB Loading Status**
  1: This bit is set when an RB Load has occurred and CMRn.WAVE is zero.
  0: This bit is cleared when the SR register is read.
- **LDRAS: RA Loading Status**
  1: This bit is set when an RA Load has occurred and CMRn.WAVE is zero.
  0: This bit is cleared when the SR register is read.
- **CPCS: RC Compare Status**
  1: This bit is set when an RC Compare has occurred.
  0: This bit is cleared when the SR register is read.

- **CPBS: RB Compare Status**
  1: This bit is set when an RB Compare has occurred and CMRn.WAVE is one.
  0: This bit is cleared when the SR register is read.
- **CPAS: RA Compare Status**
  1: This bit is set when an RA Compare has occurred and CMRn.WAVE is one.
  0: This bit is cleared when the SR register is read.
- **LOVRS: Load Overrun Status**
  1: This bit is set when RA or RB have been loaded at least twice without any read of the corresponding register and CMRn.WAVE is zero.
  0: This bit is cleared when the SR register is read.
- **COVFS: Counter Overflow Status**
  1: This bit is set when a counter overflow has occurred.
  0: This bit is cleared when the SR register is read.

31.7.9 Channel Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x24 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

31.7.10 Channel Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x28 + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

31.7.11 Channel Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x2C + n * 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

31.7.12 Block Control Register

Name: BCR

Access Type: Write-only

Offset: 0xC0

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	SYNC

- **SYNC: Synchro Command**
  1: Writing a one to this bit asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.
  0: Writing a zero to this bit has no effect.

31.7.13 Block Mode Register

Name: BMR

Access Type: Read/Write

Offset: 0xC4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	TC2XC2S		TC1XC1S		TC0XC0S	

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S	Signal Connected to XC2
0	TCLK2
1	none
2	TIOA0
3	TIOA1

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S	Signal Connected to XC1
0	TCLK1
1	none
2	TIOA0
3	TIOA2

- **TC0XC0S: External Clock Signal 0 Selection**

TC0XC0S	Signal Connected to XC0
0	TCLK0
1	none
2	TIOA1
3	TIOA2

31.7.14 Features Register

Name: FEATURES

Access Type: Read-only

Offset: 0xF8

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	

15	14	13	12	11	10	9	8
-	-	-	-	-	-	BRPBHSB	UPDNIMPL

7	6	5	4	3	2	1	0
CTRSIZE							

- **BRPBHSB: Bridge type is PB to HSB**
  1: Bridge type is PB to HSB.
  0: Bridge type is not PB to HSB.
- **UPDNIMPL: Up/down is implemented**
  1: Up/down counter capability is implemented.
  0: Up/down counter capability is not implemented.
- **CTRSIZE: Counter size**
  This field indicates the size of the counter in bits.

31.7.15 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

31.8 Module Configuration

The specific configuration for each TC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks according to the table in the Power Manager section.

Table 31-4. Module Clock Name

Module name	Clock name	Description
TC0	CLK_TC0	Peripheral Bus clock from the PBC clock domain
TC1	CLK_TC1	Peripheral Bus clock from the PBA clock domain

31.8.1 Clock Connections

Each Timer/Counter channel can independently select an internal or external clock source for its counter:

Table 31-5. Timer/Counter clock connections

Module	Source	Name	Connection
TC0	Internal	TIMER_CLOCK1	32 KHz oscillator clock (OSC32K)
		TIMER_CLOCK2	PBC clock / 2 (TIMER0_CLOCK2)
		TIMER_CLOCK3	PBC clock / 8 (TIMER0_CLOCK3)
		TIMER_CLOCK4	PBC clock / 32 (TIMER0_CLOCK4)
		TIMER_CLOCK5	PBC clock / 128 (TIMER0_CLOCK5)
	External	TC0 - CLK0	See Peripheral Multiplexing on I/O line chapter
		TC0 - CLK1	
		TC0 - CLK2	
TC1	Internal	TIMER_CLOCK1	32 KHz oscillator clock (OSC32K)
		TIMER_CLOCK2	PBA clock / 2 (TIMER1_CLOCK2)
		TIMER_CLOCK3	PBA clock / 8 (TIMER1_CLOCK3)
		TIMER_CLOCK4	PBA clock / 32 (TIMER1_CLOCK4)
		TIMER_CLOCK5	PBA clock / 128 (TIMER1_CLOCK5)
	External	TC1 - CLK0	See Peripheral Multiplexing on I/O line chapter
		TC1 - CLK1	
		TC1 - CLK2	

32. USB Interface (USBC)

Rev: 2.1.0.14

32.1 Features

- **Compatible with the USB 2.0 specification**
- **Supports full (12Mbit/s) and low (1.5Mbit/s) speed communication**
- **Supports Embedded Host**
- **7 physical pipes/endpoints in ping-pong mode**
- **Flexible pipe/endpoint configuration and reallocation of data buffers in embedded RAM**
- **Supports an infinite number of virtual pipes (alternate pipe)**
- **Up to two memory banks per pipe/endpoint**
- **Built-in DMA with multi-packet support through ping-pong mode**
- **On-chip transceivers with built-in pull-ups and pull-downs**
- **On-chip Embedded Host pad with a VBUS analog comparator**

32.2 Overview

The Universal Serial Bus interface (USBC) module complies with the Universal Serial Bus (USB) 2.0 specification, .

Each pipe/endpoint can be configured into one of several transfer types. It can be associated with one or more memory banks (located inside the embedded system or CPU RAM) used to store the current data payload. If two banks are used (“ping-pong” mode), then one bank is read or written by the CPU (or any other HSB master) while the other is read or written by the USBC core.

Table 32-1 describes the hardware configuration of the USBC module.

Table 32-1. Description of USB pipes/endpoints

pipe/endpoint	Mnemonic	Max. size	Number of available banks	Type
0	PEP0	1023 bytes	1	Control/Isochronous/Bulk/Interrupt
1	PEP1	1023 bytes	2	Control/Isochronous/Bulk/Interrupt
2	PEP2	1023 bytes	2	Control/Isochronous/Bulk/Interrupt
...	...	...	...	...
6	PEP6	1023 bytes	2	Control/Isochronous/Bulk/Interrupt

32.3 Block Diagram

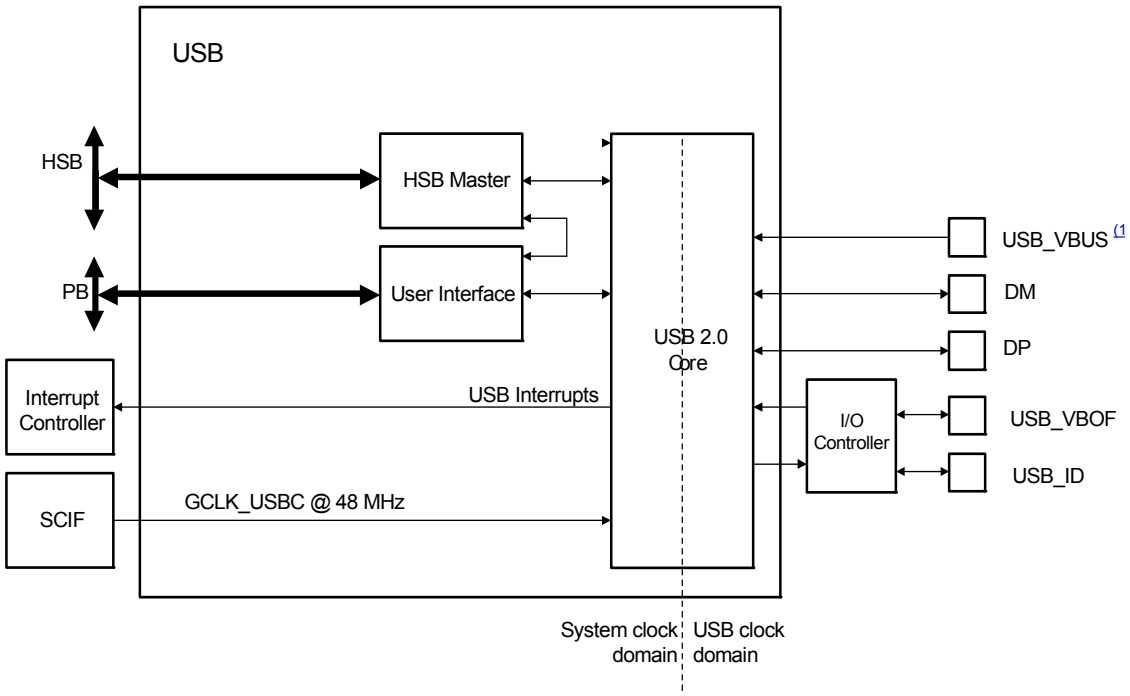
The USBC interfaces a USB link with a data flow stored in the embedded ram (CPU or HSB).

The USBC requires a 48MHz ± 0.25% reference clock, which is the USB generic clock. For more details see ”Clocks” on page 874. The 48MHz clock is used to generate either a 12MHz full-speed or a 1.5MHz low-speed bit clock from the received USB differential data, and to transmit data according to full- or low-speed USB device tolerances. Clock recovery is achieved by a digital phase-locked loop (a DPLL, not represented) in the USBC module, which complies with the USB jitter specifications.

The USBC module consists of:

- HSB master interface
- User interface
- USB Core
- Transceiver pads

Figure 32-1. USBC Block Diagram



Note: The USB_VBUS pin is 5V tolerant

32.4 I/O Lines Description

Table 32-2. I/O Lines Description

PIn Name	Pin Description	Type	Active Level
USB_VBOF	USB VBUS On/Off: Bus Power Control Port	Output	USBCON.VBUSPO
USB_VBUS	VBUS: Bus Power Measurement Port	Input	
DM	Data -: Differential Data Line - Port	Input/Output	
DP	Data +: Differential Data Line + Port	Input/Output	
USB_ID	USB Identification: Mini Connector Identification Port	Input	Low: Mini-A plug High Z: Mini-B plug

32.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

32.5.1 I/O Lines

The USBC pins may be multiplexed with the I/O Controller lines. The user must first configure the I/O Controller to assign the desired USBC pins to their peripheral functions.

If the USB_ID pin is used the user must also enable its internal pull-up resistor.

32.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the USBC, the USBC will stop functioning and resume operation after the system wakes up from sleep mode.

32.5.3 Clocks

The USBC has two bus clocks connected: One High Speed Bus clock (CLK_USBC_HSB) and one Peripheral Bus clock (CLK_USBC_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by the Power Manager. It is recommended to disable the USBC before disabling the clocks, to avoid freezing the USBC in an undefined state.

The 48 MHz USB clock is generated by a dedicated generic clock from the SCIF module. Before using the USB, the user must ensure that the USB generic clock (GCLK_USBC) is enabled at 48 MHz in the SCIF module.

32.5.4 Interrupts

The USBC interrupt request line is connected to the interrupt controller. Using the USBC interrupt requires the interrupt controller to be programmed first.

The USBC asynchronous interrupts can wake the CPU from any sleep mode:

- The ID Transition Interrupt (IDTI)
- The VBUS Transition Interrupt (VBUSTI) if the bandgap voltage reference is ON (Refer to the Power Manager chapter)
- The Wakeup Interrupt (WAKEUP)
- The Host Wakeup Interrupt (HWUPI)

32.6 Functional Description

32.6.1 USB General Operation

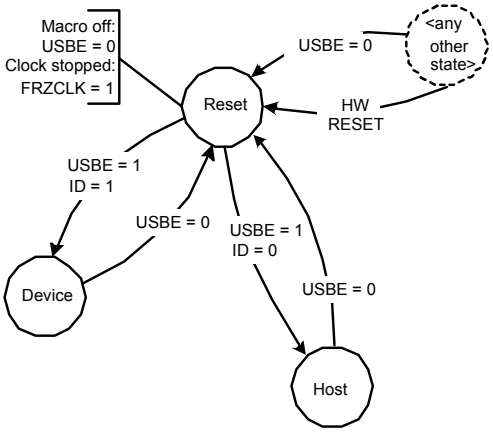
32.6.1.1 Initialization

After a hardware reset, the USBC is disabled. When enabled, the USBC runs in either device mode or in host mode according to the ID detection.

If the USB_ID pin is not connected to ground and the pull-up is enabled, the USB_ID state bit in the General Status register (USBSTA.ID) will be set and the device mode will be enabled.

If a low level is detected on the USB_ID pin, the USBSTA.ID bit will be cleared and the host mode will be enabled.

Figure 32-2. General states



After a hardware reset, the USBC is in the Reset state. In this state:

- The module is disabled. The USBC Enable bit in the General Control register (USBCON.USBE) is reset.
- The module clock is stopped in order to minimize power consumption. The Freeze USB Clock bit in USBCON (USBCON.FRZCLK) is set.
- The USB pad is in suspend mode.
- The internal states and registers of the device and host modes are reset.
- The USBSTA.ID bit and the VBUS Level bit (USBSTA.VBUS) reflect the states of the USB_ID and USB_VBUS input pins.
- The VBUS Level bit (USBSTA.VBUS) reflects the states of the USB_VBUS input pins.
- The OTG Pad Enable (OTGPADE), VBUS Polarity (VBUSPO), Freeze USB Clock (FRZCLK), USBC Enable (USBE), USB_ID Pin Enable (UIDE), USBC Mode (UIMOD) in USBCON, and the Low-Speed mode bit in the Device General Control register (UDCON.LS) can be written to by software, so that the user can configure pads and speed before enabling the module. These values are only taken into account once the module has been enabled and unfrozen.

After writing a one to USBCON.USBE, the USBC enters device or host mode (according to the ID detection) in idle state.

Refer to Section 32.6.2 for the basic operation of the device mode.

Refer to Section 32.6.3 for the basic operation of the host mode.

The USBC can be disabled at any time by writing a zero to USBCON.USBE, this acts as a hardware reset, except that the OTGPADE, VBUSPO, FRZCLK, UIDE, and UIMOD bits in USBCON, and the LS bits in UDCON are not reset.

32.6.1.2 *Interrupts*

One interrupt vector is assigned to the USBC.

See Section 32.6.2.18 and Section 32.6.3.16 for further details about device and host interrupts.

There are two kinds of general interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors (not related to CPU exceptions).

The processing general interrupts are:

- The ID Transition Interrupt (IDTI)
- The VBUS Transition Interrupt (VBUSTI)
- The SRP Interrupt (SRPI)
- The Role Exchange Interrupt (ROLEEXI)

The exception general interrupts are:

- The VBUS Error Interrupt (VBERRI)
- The B-Connection Error Interrupt (BCERRI)
- The HNP Error Interrupt (HNPERRI)
- The Suspend Time-Out Interrupt (STOI)

See Section 32.5.4 for asynchronous interrupts.

32.6.1.3 *Frozen clock*

When the USB clock is frozen, it is still possible to access the following bits: OTGPADE, VBUSPO, UIDE, UIMOD, FRZCLK, and USBE in the USBCON register, and LS in the UDCON register.

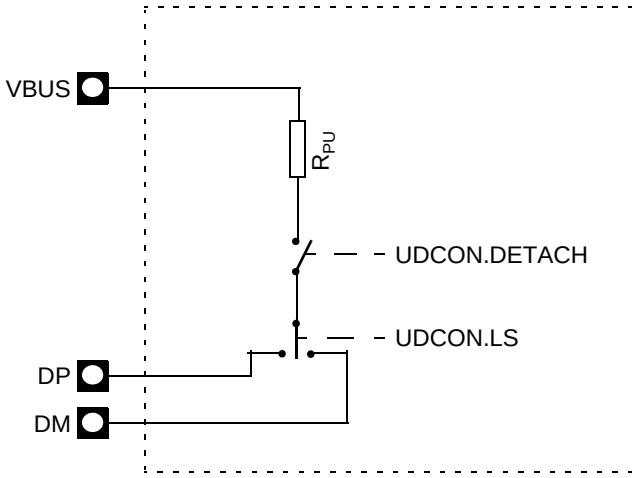
When FRZCLK is set, only the asynchronous interrupts can trigger a USB interrupt (see Section 32.5.4).

32.6.1.4 *Speed control*

- *Device mode*

When the USBC interface is in device mode, the speed selection is done by the UDCON.LS bit, connecting an internal pull-up resistor to either DP (full-speed mode) or DM (low-speed mode). The LS bit shall be written before attaching the device, which can be simulated by clearing the UDCON.DETACH bit.

Figure 32-3. Speed Selection in device mode



• *Host mode*

When the USBC interface is in host mode, internal pull-downs are enabled on both DP and DM. The interface detects the speed of the connected device and reflects this in the Speed Status field (USBSTA.SPEED).

32.6.1.5 Data management

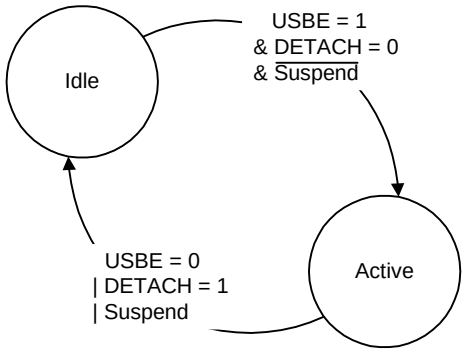
Endpoints and pipe buffers can be allocated anywhere in the embedded memory (CPU RAM or HSB RAM).

See ”RAM management” on page 883.

32.6.1.6 Pad Suspend

Figure 32-4 illustrates the behavior of the USB pad in device mode.

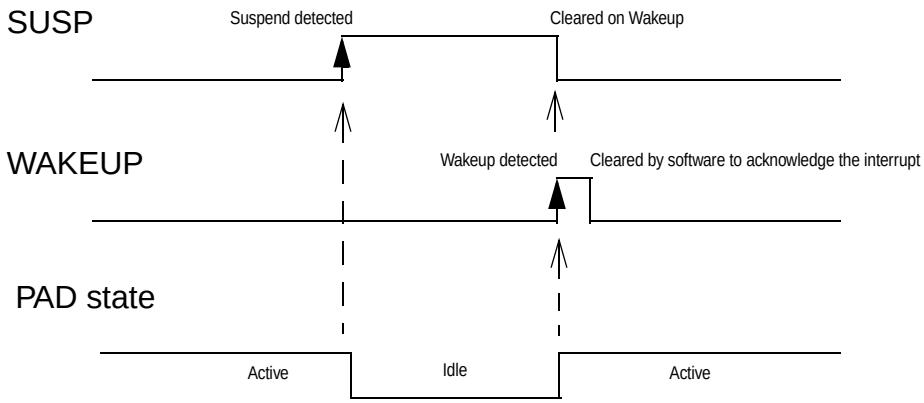
Figure 32-4. Pad Behavior



- In Idle state, the pad is in low power consumption mode.
- In Active state, the pad is working.

Figure 32-5 illustrates the pad events leading to a PAD state change.

Figure 32-5. Pad events



The Suspend Interrupt bit in the Device Global Interrupt register (UDINT.SUSP) is set and the Wakeup Interrupt (UDINT.WAKEUP) bit is cleared when a USB Suspend state has been detected on the USB bus. This event automatically puts the USB pad in the Idle state. The detection of a non-idle event sets WAKEUP, clears SUSP, and wakes the USB pad.

The pad goes to the Idle state if the module is disabled or if UDCON.DETACH is written to one. It returns to the Active state when USBCON.USBE is written to one and DETACH is written to zero.

32.6.1.7 *Customizing of Embedded Host timers*

It is possible to refine some timers thanks to the Timer Page (TIMPAGE) and Timer Value (TIMVALUE) fields in USBCON, as shown in Table 32-3.

Table 32-3. Customizing of Host Timers

		TIMPAGE			
		0b00 AWaitVrise Time-Out	**0b01 VbBusPulsing Time-Out**	**0b10 PdTmOutCnt Time-Out**	**0b11 SRPDetTmOut Time-Out**
TIMVALUE	**00**	20 ms	15 ms	93 ms	10 µs
	01	50 ms	23 ms	105 ms	100 µs
	10	70 ms	31 ms	118 ms	1 ms
	11	100 ms	40 ms	131 ms	11 ms

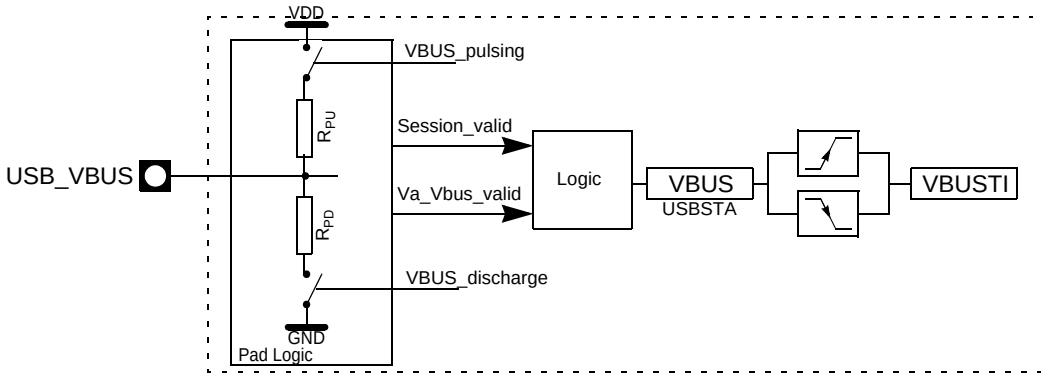
TIMPAGE is used to select the timer to be accessed while TIMVALUE indicates the time-out value of the selected timer.

TIMPAGE and TIMVALUE can be read and written. Before writing them, the user shall unlock write accesses by writing a one to the Timer Access Unlock (UNLOCK) bit in USBCON.

32.6.1.8 *Plug-in detection*

The USB connection is detected by the USB_VBUS pad. Figure 32-6 shows the architecture of the plug-in detector.

Figure 32-6. Plug-in Detection Input Block Diagram



The control logic of the USB_VBUS pad outputs two signals:

- The Session_valid signal is high when the voltage on the USB_VBUS pad is higher than or equal to 1.4V.
- The Va_Vbus_valid signal is high when the voltage on the USB_VBUS pad is higher than or equal to 4.4V.

In device mode, the USBSTA.VBUS bit follows the Session_valid comparator output:

- It is set when the voltage on the USB_VBUS pad is higher than or equal to 1.4V.
- It is cleared when the voltage on the VBUS pad is lower than 1.4V.

In host mode, the USBSTA.VBUS bit follows a hysteresis based on Session_valid and Va_Vbus_valid:

- It is set when the voltage on the USB_VBUS pad is higher than or equal to 4.4V.
- It is cleared when the voltage on the USB_VBUS pad is lower than 1.4V.

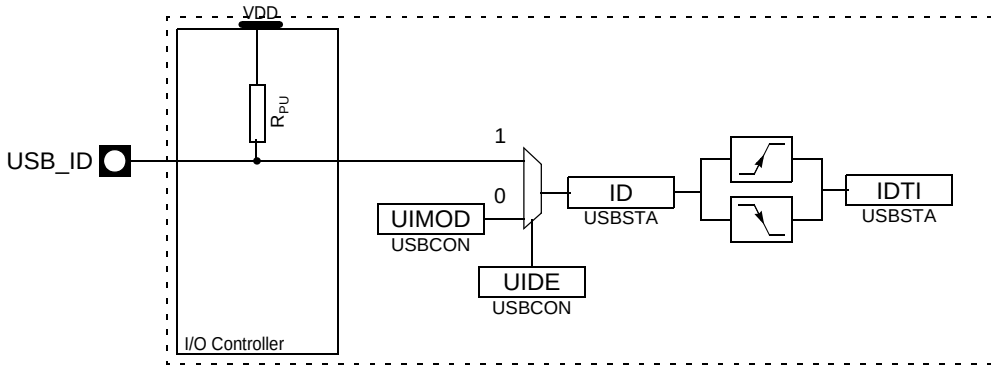
The VBUS Transition Interrupt (VBUSTI) bit in USBSTA is set on each transition of the USBSTA.VBUS bit.

The USBSTA.VBUS bit is operational regardless of whether the USBC is enabled or not.

32.6.1.9 ID detection

Figure 32-7 shows how the ID transitions are detected.

Figure 32-7. ID Detection Input Block Diagram



The USBC mode (device or host) can be detected by the USB_ID pin or selected by writing to USBCON.UIMOD. This allows the USB_ID pin to be used as a general purpose I/O pin even when the USBC interface is enabled.

The ID Transition Interrupt (IDTI) bit in USBSTA is set on each transition of the ID bit, i.e. when a Mini-A plug (host mode) is connected or disconnected. This does not occur when a Mini-B plug in device mode is connected or disconnected.

The USBSTA.ID bit is operational regardless of whether the USBC is enabled or not.

32.6.2 USBC Device Mode Operation

32.6.2.1 Device Enabling

In device mode, the USBC supports full- and low-speed data transfers.

Including the default control endpoint, a total of seven endpoints are provided. They can be configured as isochronous, bulk or interrupt types, as described in Table 32-1 on page 871

After a hardware reset, the USBC device mode is in the reset state (see Section 32.6.1.1). In this state, the endpoint banks are disabled and neither DP nor DM are pulled up (DETACH is one).

DP or DM will be pulled up according to the selected speed as soon as the DETACH bit is written to zero and VBUS is present. See "Device mode" for further details.

When the USBC is enabled (USBE is one) in device mode, it enters the Idle state, minimizing power consumption. Being in Idle state does not require the USB clocks to be activated.

The USBC device mode can be disabled or reset at any time by disabling the USBC (by writing a zero to USBE) or by enabling host mode (ID is zero).

32.6.2.2 USB reset

The USB bus reset is initiated by a connected host and managed by hardware.

When a USB reset state is detected on the USB bus, the following operations are performed by the controller:

- UDCON register is reset except for the DETACH and SPDCONF bits.
- Device Frame Number Register (UDFNUM), Endpoint n Configuration Register (UECFGn), and Endpoint n Control Register (UECONn) registers are cleared.
- The data toggle sequencing in all the endpoints are cleared.
- At the end of the reset process, the End of Reset (EORST) bit in the UDINT register is set.

32.6.2.3 Endpoint activation

When an endpoint is disabled (UERST.EPENn = 0) the data toggle sequence, Endpoint n Status Set (UESTAn), and UECONn registers will be reset. The controller ignores all transactions to this endpoint as long as it is inactive.

To complete an endpoint activation, the user should fill out the endpoint descriptor: see Figure 32-8 on page 884.

32.6.2.4 Data toggle sequence

In order to respond to a CLEAR_FEATURE USB request without disabling the endpoint, the user can clear the data toggle sequence by writing a one to the Reset Data Toggle Set bit in the Endpoint n Control Set register (UECONnSET.RSTDTS)

32.6.2.5 Busy bank enable

In order to make an endpoint bank look busy regardless of its actual state, the user can write a one to the Busy Bank Enable bit in the Endpoint n Control Register (UECONnSET.BUSY0/1ES).

If a BUSYnE bit is set, any transaction to this bank will be rejected with a NAK reply.

32.6.2.6 Address setup

The USB device address is set up according to the USB protocol.

- After all kinds of resets, the USB device address is 0.
- The host starts a SETUP transaction with a SET_ADDRESS(addr) request.
- The user writes this address to the USB Address field (UDCON.UADD), and writes a zero to the Address Enable bit (UDCON.ADDEN), resulting in the address remaining zero.
- The user sends a zero-length IN packet from the control endpoint.
- The user enables the stored USB device address by writing a one to ADDEN.

Once the USB device address is configured, the controller filters the packets to only accept those targeting the address stored in UADD.

UADD and ADDEN should not be written to simultaneously. They should be written sequentially, UADD field first.

If UADD or ADDEN is cleared, the default device address 0 is used. UADD and ADDEN are cleared:

- On a hardware reset.
- When the USBC is disabled (USBE written to zero).
- When a USB reset is detected.

32.6.2.7 Suspend and Wakeup

When an idle USB bus state has been detected for 3 ms, the controller sets the Suspend (SUSP) interrupt bit in UDINT. In this case, the transceiver is suspended, reducing power consumption.

To further reduce power consumption it is recommended to freeze the USB clock by writing a one to the Freeze USB Clock (FRZCLK) bit in USBCON when the USB bus is in suspend mode. The MCU can also enter the idle or frozen sleep mode to further lower power consumption.

To recover from the suspend mode, the user shall wait for the Wakeup (WAKEUP) interrupt bit, which is set when a non-idle event is detected, and then write a zero to FRZCLK.

As the WAKEUP interrupt bit in UDINT is set when a non-idle event is detected, it can occur regardless of whether the controller is in the suspend mode or not. The SUSP and WAKEUP interrupts are thus independent of each other except for that one bit is cleared when the other is set.

32.6.2.8 Detach

The reset value of the DETACH bit located in the UDCON register, is one.

It is possible to initiate a device re-enumeration simply by writing a one and then a zero to DETACH.

DETACH acts on the pull-up connections of the DP and DM pads. See “Device mode” for further details.

32.6.2.9 Remote wakeup

The remote wakeup request (also known as upstream resume) is the only request the device may send on its own initiative. This should be preceded by a DEVICE_REMOTE_WAKEUP request from the host.

- First, the USBC must have detected a “Suspend” state on the bus, i.e. the remote wakeup request can only be sent after a SUSP interrupt has been set.

- The user may then write a one to the remote wakeup (RMWKUP) bit in UDCON to send an Upstream Resume to the host initiating the wakeup. This will automatically be done by the controller after 5ms of inactivity on the USB bus.
- When the controller sends the Upstream Resume, the Upstream Resume (UPRSM) interrupt is set and SUSP is cleared.
- RMWKUP is cleared at the end of the transmitting Upstream Resume.
- In case of a rebroadcast resume initiated by the host, the End of Resume (EORSM) interrupt is set when the rebroadcast resume is completed.

32.6.2.10 RAM management

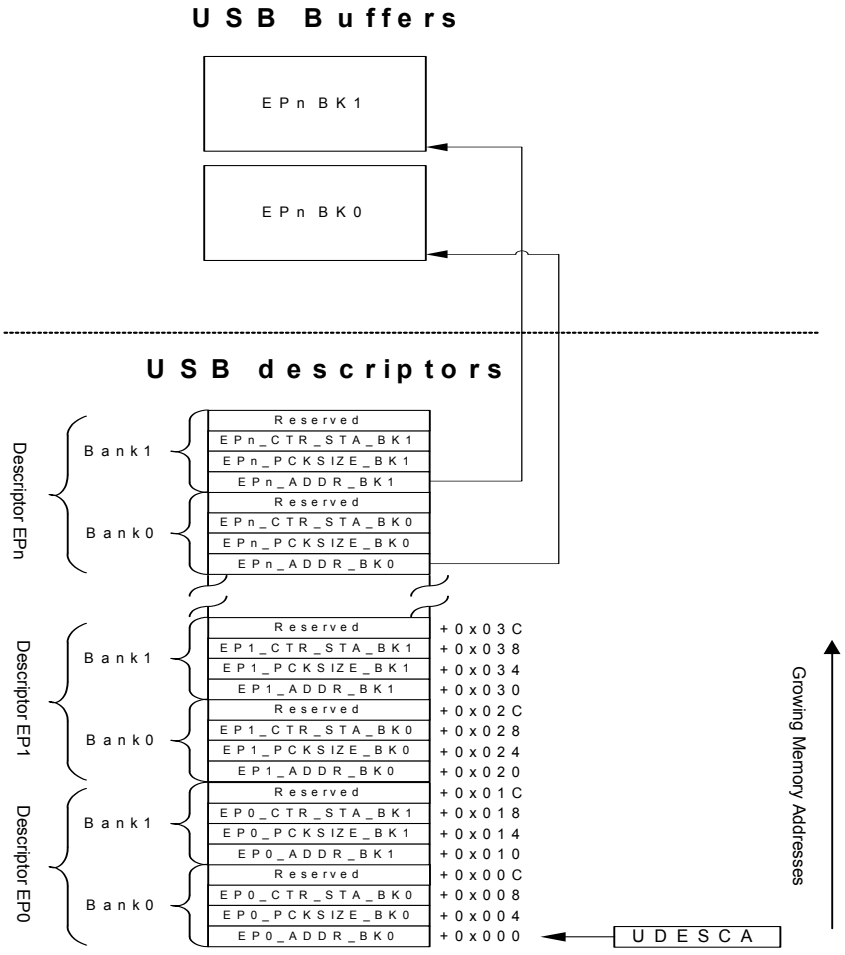
Endpoint data can be physically allocated anywhere in the embedded RAM. The USBC controller accesses these endpoints directly through the HSB master (built-in DMA).

The USBC controller reads the USBC descriptors to know where each endpoint is located. The base address of the USBC descriptor (UDESC.UDESCA) needs to be written by the user. The descriptors can also be allocated anywhere in the embedded RAM.

Before using an endpoint, the user should setup the endpoint address for each bank. Depending on the direction, the type, and the packet-mode (single or multi-packet), the user should also initialize the endpoint packet size, and the endpoint control and status fields, so that the USBC controller does not compute random values from the RAM.

When using an endpoint the user should read the UESTAX.CURRBK field to know which bank is currently being processed.

Figure 32-8. Memory organization



Each descriptor of an endpoint n consists of four words.

- The address of the endpoint and the bank used (EPn_ADDR_BK0/1).
- The packet size information for the endpoint and bank (EPn_PCKSIZE_BK0/1):

Table 32-4. EPn_PCKSIZE_BK0/1 structure

31	30:16	15	14:0
AUTO_ZLP	MULTI_PACKET_SIZE	-	BYTE_COUNT

– AUTO_ZLP: Auto zero length packet, see ”Multi packet mode for IN endpoints” on page 889.

– MULTI_PACKET_SIZE: see ”Multi packet mode and single packet mode.” on page 886.

– BYTE_COUNT: see ”Multi packet mode and single packet mode.” on page 886.

- The control and status fields for the endpoint and bank (EPn_CTR_STA_BK0/1):

Table 32-5. EPn_CTR_STA_BK0/1 structure

31:19	18	17	16	15:1	0
Status elements				Control elements	
-	UNDERF	OVERF	CRCERR	-	STALLRQ_NEXT

  - UNDERF: Underflow status for isochronous IN transfer. See "Data flow error" on page 892.
  - OVERF: Overflow status for isochronous OUT transfer. See "Data flow error" on page 892.
  - CRCERR: CRC error status for isochronous OUT transfer. See "CRC error" on page 892.
  - STALLRQ_NEXT: Stall request for the next transfer. See "STALL request" on page 885.

32.6.2.11 *STALL request*

For each endpoint, the STALL management is performed using:

- The STALL Request (STALLRQ) bit in UECONn is set to initiate a STALL request.
- The STALLed Interrupt (STALLEDI) bit in UESTAn is set when a STALL handshake has been sent.

To answer requests with a STALL handshake, STALLRQ has to be set by writing a one to the STALL Request Set (STALLRQS) bit. All following requests will be discarded (RXOUTI, etc. will not be set) and handshaked with a STALL until the STALLRQ bit is cleared, by receiving a new SETUP packet (for control endpoints) or by writing a one to the STALL Request Clear (STALLRQC) bit.

Each time a STALL handshake is sent, the STALLEDI bit is set by the USBC and the EPnINT interrupt is set.

The user can use the descriptor to manage STALL requests. The USBC controller reads the EPn_CTR_STA_BK0/1.STALLRQ_NEXT bit after successful transactions and if it is one the USBC controller will set UECON.STALLRQ. The STALL_NEXT bit will be cleared upon receiving a SETUP transaction and the USBC controller will then clear the STALLRQ bit.

- *Special considerations for control endpoints*

  If a SETUP packet is received at a control endpoint where a STALL request is active, the Received SETUP Interrupt (RXSTPI) bit in UESTAn is set, and the STALLRQ and STALLEDI bits are cleared. It allows the SETUP to be always ACKed as required by the USB standard.

  This management simplifies the enumeration process management. If a command is not supported or contains an error, the user requests a STALL and can return to the main task, waiting for the next SETUP request.

- *STALL handshake and retry mechanism*

  The retry mechanism has priority over the STALL handshake. A STALL handshake is sent if the STALLRQ bit is set and if there is no retry required.

32.6.2.12 *Multi packet mode and single packet mode.*

Single packet mode is the default mode where one USB packet is managed per bank.

The multi-packet mode allows the user to manage data exceeding the maximum endpoint size (UECFGn.EPSIZE) for an endpoint bank across multiple packets without software intervention. This mode can also be coupled with the ping-pong mode.

- For an OUT endpoint, the EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE field should be configured correctly to enable the multi-packet mode. See "Multi packet mode for OUT endpoints" on page 891. For single packet mode, the MULTI_PACKET_SIZE should be initialized to 0.
- For an IN endpoint, the EPn_PCKSIZE_BK0/1.BYTE_COUNT field should be configured correctly to enable the multi-packet mode. See"Multi packet mode for IN endpoints" on page 889. For single packet mode, the BYTE_COUNT should be less than EPSIZE.

32.6.2.13 *Management of control endpoints*

• *Overview*

A SETUP request is always ACKed. When a new SETUP packet is received, the RXSTPI is set, but not the Received OUT Data Interrupt (RXOUTI) bit.

The FIFO Control (FIFOCON) bit in UECONn is irrelevant for control endpoints. The user should therefore never use it for these endpoints. When read, this value is always zero.

Control endpoints are managed using:

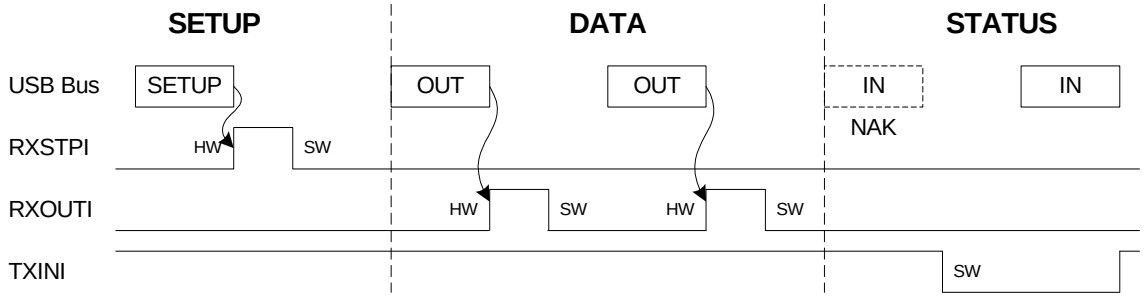
- The RXSTPI bit: is set when a new SETUP packet is received. This has to be cleared by firmware in order to acknowledge the packet and to free the bank.
- The RXOUTI bit: is set when a new OUT packet is received. This has to be cleared by firmware in order to acknowledge the packet and to free the bank.
- The Transmitted IN Data Interrupt (TXINI) bit: is set when the current bank is ready to accept a new IN packet. This has to be cleared by firmware in order to send the packet.

• *Control write*

Figure 32-9 on page 887 shows a control write transaction. During the status stage, the controller will not necessarily send a NAK on the first IN token:

- If the user knows the exact number of descriptor bytes that will be read, the status stage can be predicted, and a zero-length packet can be sent after the next IN token.
- Alternatively the bytes can be read until the NAKed IN Interrupt (NAKINI) is triggered, notifying that all bytes are sent by the host and that the transaction is now in the status stage.

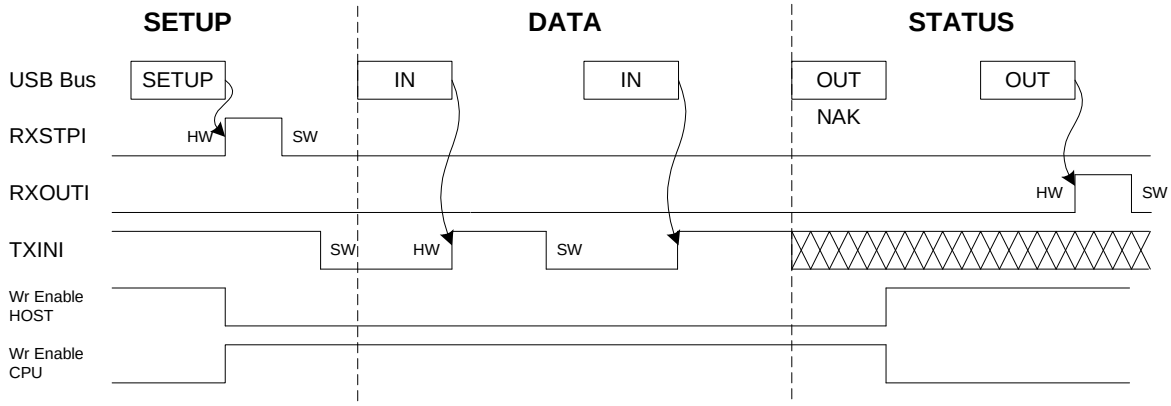
Figure 32-9. Control Write



• *Control read*

Figure 32-10 on page 887 shows a control read transaction. The USBC has to manage the simultaneous write requests from the CPU and USB host.

Figure 32-10. Control Read



A NAK handshake is always generated as the first status stage command. The UESTAn.NAKINI bit is set. It allows the user to know that the host aborts the IN data stage. As a consequence, the user should stop processing the IN data stage and should prepare to receive the OUT status stage by checking the UESTAn.RXOUTI bit.

The OUT retry is always ACKed. This OUT reception sets RXOUTI. Handle this with the following software algorithm:

```
// process the IN data stage
set TXINI
wait for RXOUTI (rising) OR TXINI (falling)
if RXOUTI is high, then process the OUT status stage
if TXINI is low, then return to process the IN data stage
```

Once the OUT status stage has been received, the USBC waits for a SETUP request. The SETUP request has priority over all other requests and will be ACKed.

32.6.2.14 Management of IN endpoints

- *Overview*

IN packets are sent by the USBC device controller upon IN requests from the host.

The endpoint and its descriptor in RAM must be pre configured (see section "RAM management" on page 883 for more details).

When the current bank is clear, the TXINI and FIFO Control (UECONn.FIFOCON) bits will be set simultaneously. This triggers an EPnINT interrupt if the Transmitted IN Data Interrupt Enable (TXINE) bit in UECONn is one.

TXINI shall be cleared by software (by writing a one to the Transmitted IN Data Interrupt Enable Clear bit in the Endpoint n Control Clear register (UECONnCLR.TXINIC)) to acknowledge the interrupt. This has no effect on the endpoint FIFO.

The user writes the IN data to the bank referenced by the EPn descriptor and allows the USBC to send the data by writing a one to the FIFO Control Clear (UECONnCLR.FIFOCONC) bit. This will also cause a switch to the next bank if the IN endpoint is composed of multiple banks. The TXINI and FIFOCON bits will be updated accordingly.

TXINI should always be cleared before clearing FIFOCON to avoid missing an TXINI event.

Figure 32-11. Example of an IN endpoint with one data bank

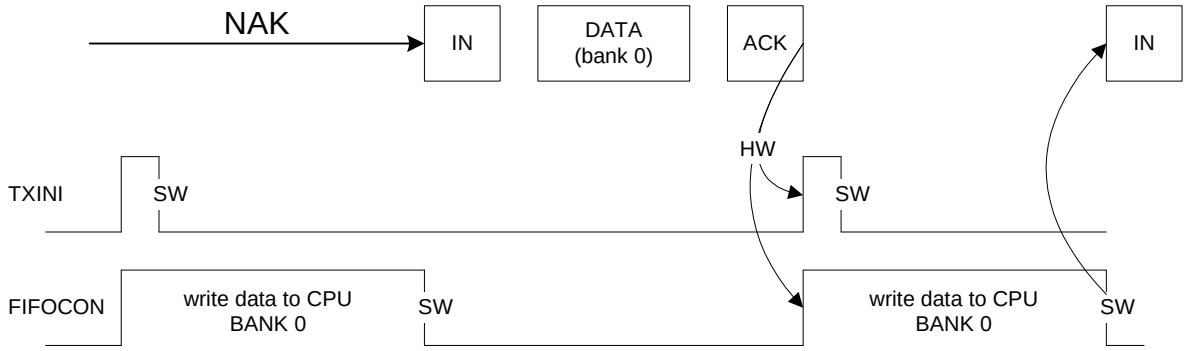
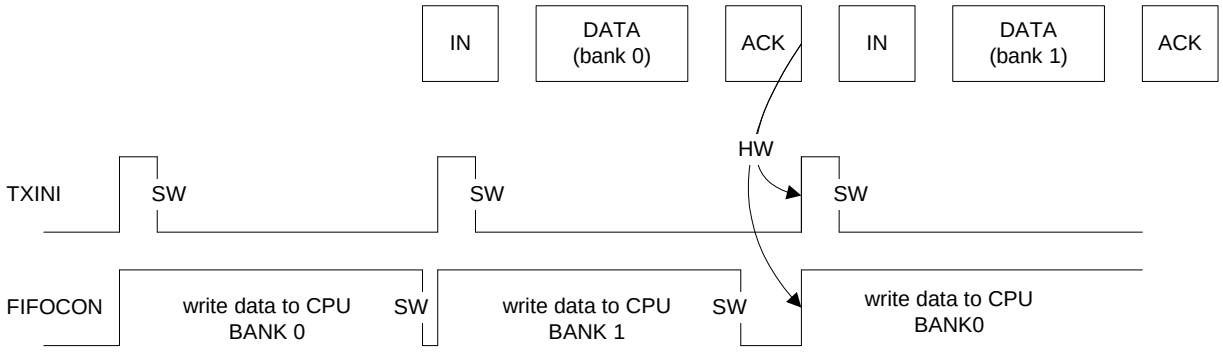


Figure 32-12. Example of an IN endpoint with two data banks



• *Detailed description*

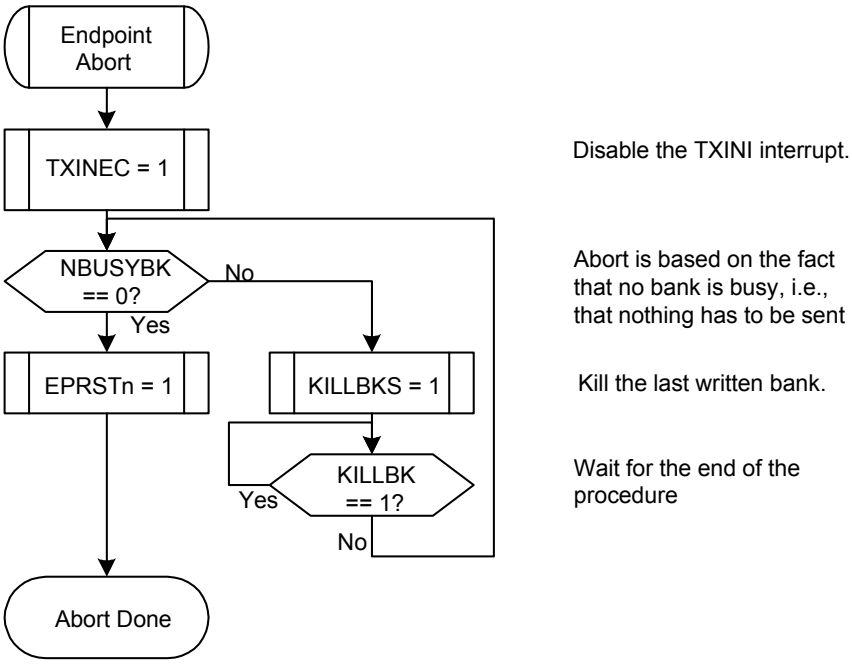
The data is written according to this sequence:

- When the bank is empty, TXINI and FIFOCON are set, which triggers an EPnINT interrupt if TXINE is one.
- The user acknowledges the interrupt by clearing TXINI.
- The user reads the UESTAX.CURRBK field to see which the current bank is.
- The user writes the data to the current bank, located in RAM as described by its descriptor: EPn_ADDR_BK0/1.
- The user should write the size of the IN packet into the USB descriptor: EPn_PCKSIZE_BK0/1.BYTE_COUNT.
- The user allows the controller to send the bank contents and switches to the next bank (if any) by clearing FIFOCON.

If the endpoint uses several banks, the current one can be written while the previous one is being read by the host. When the user clears FIFOCON, the next current bank may already be clear and TXINI is set immediately.

An “Abort” stage can be produced when a zero-length OUT packet is received during an IN stage of a control or isochronous IN transaction. The Kill IN Bank (KILLBK) bit in UECONn is used to kill the last written bank. The best way to manage this abort is to apply the algorithm represented on Figure 32-13 on page 889. See ”Endpoint n Control Register” on page 938 for more details about the KILLBK bit.

Figure 32-13. Abort Algorithm



• *Multi packet mode for IN endpoints*

In multi packet mode, the user can prepare n USB packets in the bank to be sent on a multiple IN transaction. The packet sizes will equal UECFGn.EPSIZE unless the AUTO_ZLP option is

set, or if the total byte count is not an integral multiple of EPSIZE, whereby the last packet should be short.

To enable the multi packet mode, the user should configure the endpoint descriptor (EPn_PCKSIZE_BK0/1.BYTE_COUNT) to the total size of the multi packet, which should be larger than the endpoint size (EPSIZE).

Since the EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE is incremented (by the transmitted packet size) after each successful transaction, it should be set to zero when setting up a new multi packet transfer.

The EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE is cleared by hardware when all the bank contents have been sent. The bank is considered as ready and the TX_IN flag is set when:

- A short packet (smaller than EPSIZE) has been transmitted.
- A packet has been successfully transmitted, the updated MULTI_PACKET_SIZE equals the BYTE_COUNT, and the AUTO_ZLP field is not set.
- An extra zero length packet has been automatically sent for the last transfer of the current bank, if BYTE_COUNT is a multiple of EPSIZE and AUTO_ZLP is set.

32.6.2.15 Management of OUT endpoints

- *Overview*

The endpoint and its descriptor in RAM must be pre configured, see section "RAM management" on page 883 for more details.

When the current bank is full, the RXOUTI and FIFO Control (UECONn.FIFOCON) bits will be set simultaneously. This triggers an EPnINT interrupt if the Received OUT Data Interrupt Enable (RXOUTE) bit in UECONn is one.

RXOUTI shall be cleared by software (by writing a one to the Received OUT Data Interrupt Clear (RXOUTIC) bit) to acknowledge the interrupt. This has no effect on the endpoint FIFO.

The user reads the OUT data from the RAM and clears the FIFOCON bit to free the bank. This will also cause a switch to the next bank if the OUT endpoint is composed of multiple banks.

RXOUTI should always be cleared before clearing FIFOCON to avoid missing an RXOUTI event.

Figure 32-14. Example of an OUT endpoint with one data bank

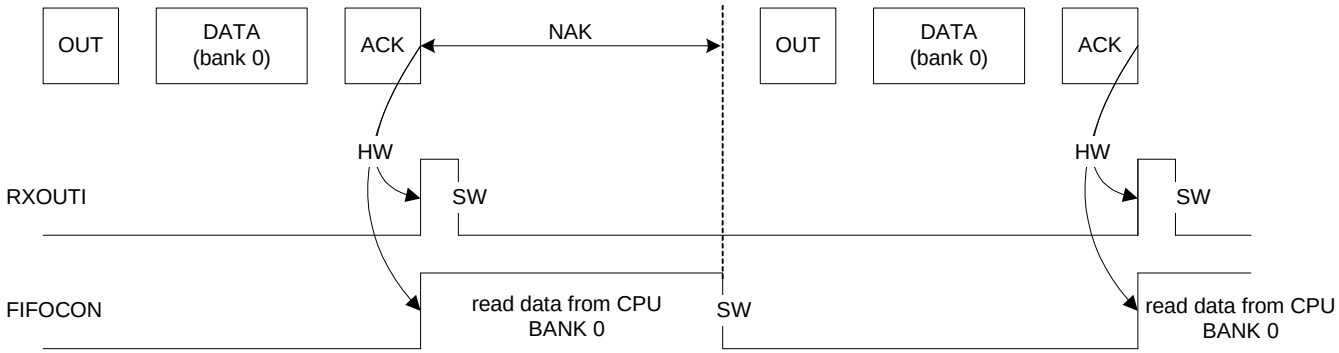
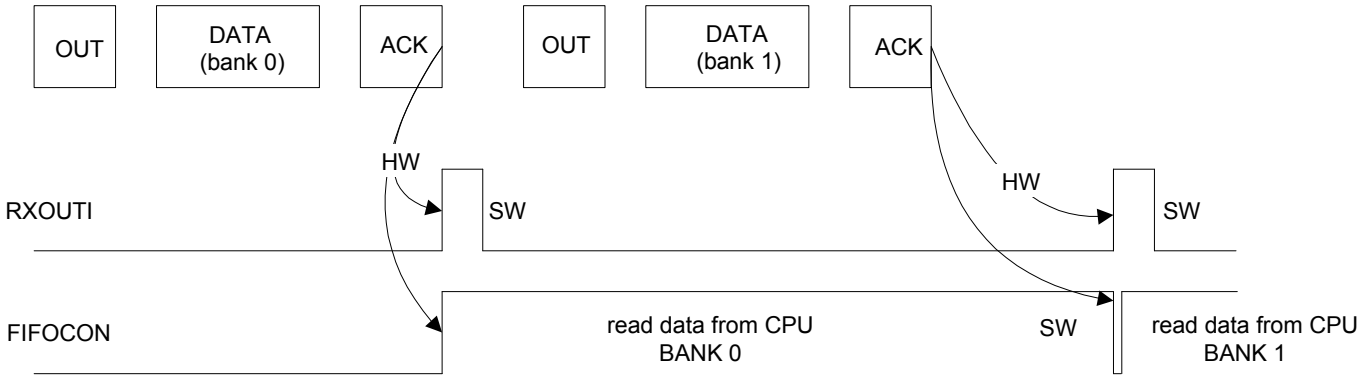


Figure 32-15. Example of an OUT endpoint with two data banks



• *Detailed description*

Before using the OUT endpoint, one should properly initialize its descriptor for each bank. See Figure 32-8 on page 884.

The data is read, according to this sequence:

- When the bank is full, RXOUTI and FIFOCON are set, which triggers an EPnINT interrupt if RXOUTE is one.
- The user acknowledges the interrupt by writing a one to RXOUTIC in order to clear RXOUTI.
- The user reads the UESTAX.CURRBK field to know the current bank number.
- The user reads the byte count of the current bank from the descriptor in RAM (EPn_PCKSIZE_BK0/1.BYTE_COUNT) to know how many bytes to read.
- The user reads the data in the current bank, located in RAM as described by its descriptor: EPn_ADDR_BK0/1.
- The user frees the bank and switches to the next bank (if any) by clearing FIFOCON.

If the endpoint uses several banks, the current one can be read while the next is being written by the host. When the user clears FIFOCON, the following bank may already be ready and RXOUTI will be immediately set.

• *Multi packet mode for OUT endpoints*

In multi packet mode, the user can extend the size of the bank allowing the storage of n USB packets in the bank.

To enable the multi packet mode, the user should configure the endpoint descriptor (EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE) to match the size of the multi packet.This value should be a multiple of the endpoint size (UECFGn.EPSIZE).

Since the EPn_PCKSIZE_BK0/1.BYTE_COUNT is incremented (by the received packet size) after each successful transaction, it should be set to zero when setting up a new multi packet transfer.

As for single packet mode, the number of received data bytes is stored in the BYTE_CNT field.

The bank is considered as “valid” and the RX_OUT flag is set when:

- A packet has been successfully received and the updated BYTE_COUNT equals the MULTI_PACKET_SIZE.
- A short packet (smaller than EPSIZE) has been received.

32.6.2.16 Data flow error

This error exists only for isochronous IN/OUT endpoints. It sets the Errorflow Interrupt (ERRORFI) bit in UESTAn, which triggers an EPnINT interrupt if the Errorflow Interrupt Enable (ERRORFE) bit is one. The user can check the EPn_CTR_STA_BK0/1.UNDERF and OVERF bits in the endpoint descriptor to see which current bank has been affected.

- An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBC. The endpoint descriptor EPn_CTR_STA_BK0/1.UNDERF points out the bank from which the IN data should have originated. If a new successful transaction occurs, the UNDERF bit is overwritten to 0 only if the UESTAn.ERRORFI is cleared.
- An overflow can occur during the OUT stage if the host tries to send a packet while the bank is full. Typically this occurs when a CPU is not fast enough. The packet data is not written to the bank and is lost. The endpoint descriptor EPn_CTR_STA_BK0/1.OVERF points out which bank the OUT data was destined to. If the UESTAn.ERRORFI bit is cleared and a new transaction is successful, the OVERF bit will be overwritten to zero.

32.6.2.17 CRC error

This error exists only for isochronous OUT endpoints. It sets the CRC Error Interrupt (CRCERRI) bit in UESTAn, which triggers an EPnINT interrupt if the CRC Error Interrupt Enable (CRCERRE) bit is one.

A CRC error can occur during an isochronous OUT stage if the USBC detects a corrupted received packet. The OUT packet is stored in the bank as if no CRC error had occurred (RXOUTI is set).

The user can also check the endpoint descriptor to see which current bank is impacted by the CRC error by reading EPn_CTR_STA_BK0/1.CRCERR.

32.6.2.18 Interrupts

There are two kinds of device interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors not related to CPU exceptions.

• Global interrupts

The processing device global interrupts are:

- The Suspend (SUSP) interrupt
- The Start of Frame (SOF) interrupt with no frame number CRC error (the Frame Number CRC Error (FNCERR) bit in the Device Frame Number (UDFNUM) register is zero)
- The End of Reset (EORST) interrupt
- The Wakeup (WAKEUP) interrupt
- The End of Resume (EORSM) interrupt
- The Upstream Resume (UPRSM) interrupt
- The Endpoint n (EPnINT) interrupt

The exception device global interrupts are:

- The Start of Frame (SOF) interrupt with a frame number CRC error (FNCERR is one)

• *Endpoint interrupts*

The processing device endpoint interrupts are:

- The Transmitted IN Data Interrupt (TXINI)
- The Received OUT Data Interrupt (RXOUTI)
- The Received SETUP Interrupt (RXSTPI)
- The Number of Busy Banks (NBUSYBK) interrupt

The exception device endpoint interrupts are:

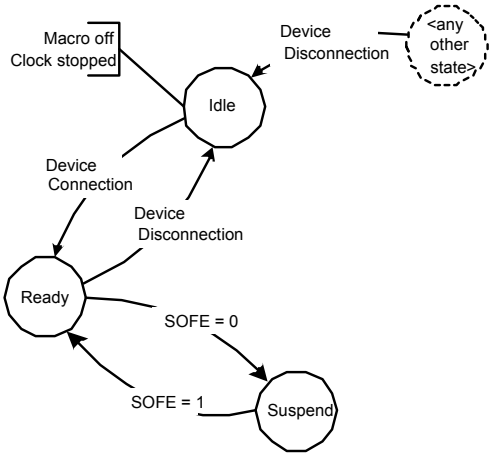
- The Errorflow Interrupt (ERRORFI)
- The NAKed OUT Interrupt (NAKOUTI)
- The NAKed IN Interrupt (NAKINI)
- The STALLed Interrupt (STALLEDI)
- The CRC Error Interrupt (CRCERRI)

32.6.3 USB Host Operation

32.6.3.1 Host Enabling

Figure 32-16 on page 894 describes the USBC host mode main states.

Figure 32-16. Host mode states



After a hardware reset, the USBC host mode is in the Reset state (see Section 32.6.1.1).

When the USBC is enabled (USBCON.USBE = 1) in host mode (USBSTA.ID = 0) it enters Idle state and waits for a device connection. Once a device is connected, the USBC enters the Ready state, which does not require the USB clock to be activated.

In host mode the USBC will suspend the USB bus by not transmitting any Start Of Frame (SOF) packets (the Start of Frame Generation Enable bit in the Host Global Interrupt register UHCON.SOFE is zero). The USBC enters the Suspend state when the USB bus is suspended, and exits when SOF generation is resumed.

32.6.3.2 Device detection

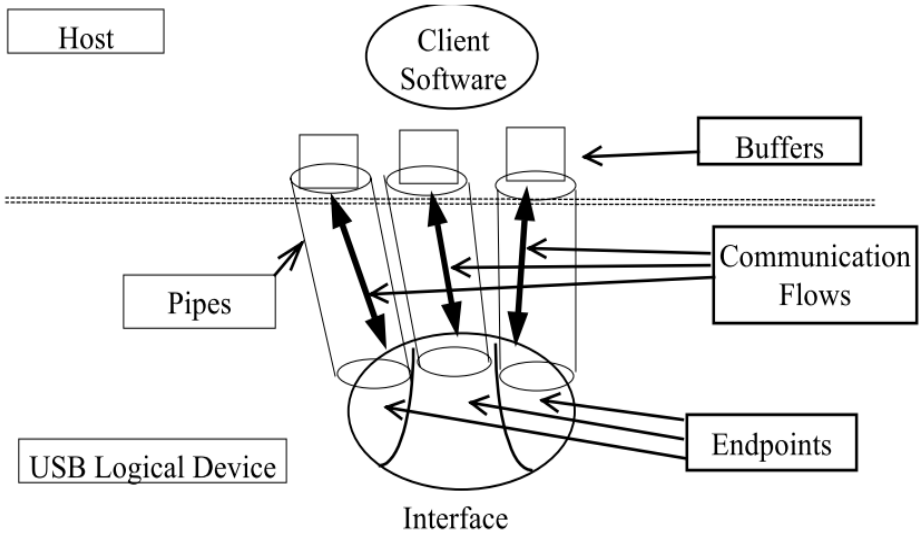
A device is detected by the USBC in host mode when DP or DM are not tied low, i.e., when a device DP or DM pull-up resistor is connected. To enable this detection, the host controller has to supply the device with VBUS power, which is done when USBSTA.VBUSRQ is one.

The device disconnection is detected by the host controller when both DP and DM are pulled down.

32.6.3.3 Description of pipes

In host mode, the term “pipe” is used instead of “endpoint”. A host pipe corresponds to a device endpoint, as illustrated by Figure 32-17 on page 895 from the USB specification.

Figure 32-17. USB Communication Flow



In host mode, the USBC associates a pipe to a device endpoint, according to the device configuration descriptors.

32.6.3.4 USB reset

The USBC sends a USB reset signal when the user writes a one to the Send USB Reset bit (UHCON.RESET). When the USB reset has been sent, the USB Reset Sent Interrupt bit in the Host Global Interrupt register (UHINT.RSTI) is set and all the pipes will be disabled.

If the bus was previously in a suspended state (UHCON.SOFE is zero) the USBC will switch it to the Resume state, causing the bus to asynchronously trigger the Host Wakeup Interrupt (UHINT.HWUPI). The SOFE bit will be set in order to generate SOF's immediately after the USB reset.

32.6.3.5 Pipe activation

A disabled pipe is inactive, and will be reset along with its context registers (UPCONn, UPSTAn, UPINRQn, and UPCFGn). Enabling a pipe is done by writing a one to the Pipe n Enable bit in the Pipe Enable/Reset Register (UPRST.PENn).

When starting an enumeration, the user gets the device descriptor by sending an GET_DESCRIPTOR USB request. This descriptor contains the maximal packet size of the device default control endpoint (bMaxPacketSize0) which the user should use to reconfigure the size of the default control pipe.

32.6.3.6 Address setup

Once the device has answered the first host requests with the default device address 0, the host assigns a new address to the device. The host controller has to send a USB reset to the device and a SET_ADDRESS(addr) SETUP request with the new address to be used by the device. Once this SETUP transaction is over, the user writes to the device address field in the "control and status 1 of endpoint n" word of the host's pipe n in the USB descriptor (Pn_CTR_STA1.PDADDR). All following requests by this pipe will be performed using this new address.

32.6.3.7 Remote wakeup

Writing UHCON.SOFE to zero when in host mode will cause the USBC to cease sending SOF's on the USB bus and enter the Suspend state. The USB device will enter the Suspend state 3ms later.

The device can awaken the host by sending an Upstream Resume (remote wakeup feature). When the host detects a non-idle state on the USB bus, it sets the Host Wakeup interrupt bit (UHINT.HWUPI). If the non-idle bus state corresponds to an Upstream Resume (K state), the Upstream Resume Received Interrupt bit (UHINT.RXRSMI) is set and the user has to generate a Downstream Resume within 1ms and for at least 20ms. It is required to first enter the Ready state by writing a one to UHCON.SOFEF and then writing a one to the Send USB Resume bit (UHCON.RESUME).

32.6.3.8 RAM management

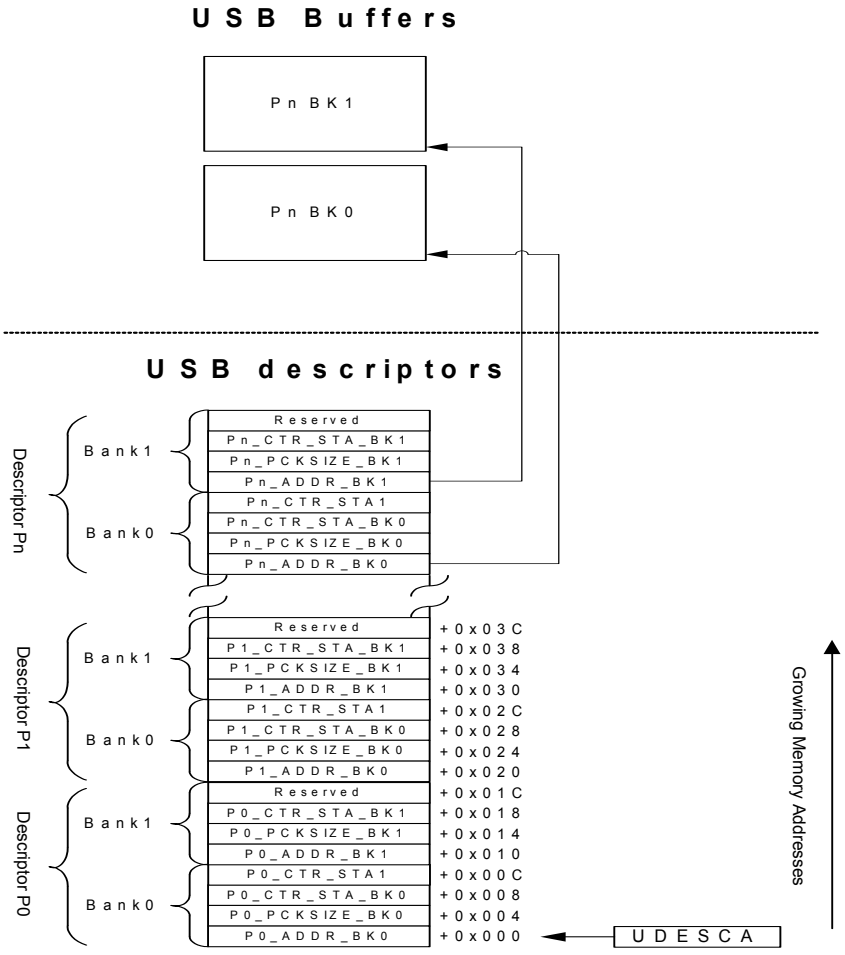
Pipe data can be physically allocated anywhere in the embedded RAM. The USBC controller accesses the pipes directly through the HSB master (built-in DMA).

The USBC controller reads the USBC descriptors to know the location of each pipe. The base address of this USBC descriptor (UDESC.UDESCA) needs to be written by the user. The descriptors can also be allocated anywhere in the embedded RAM.

Before using a pipe, the user should setup the data address for each bank. Depending on the direction, pipe type, targeted device address, targeted endpoint number, and packet-mode (single or multi-packet), the user should also initialize the pipe packet size and the pipe control and status field, so that the USB controller does not compute random values from the RAM.

When using a pipe, the user should read the UPSTAX.CURRBK field to know which bank is currently processed.

Figure 32-18. Memory organization



Each pipe n descriptor bank consists of four words.

- The address of the pipe and bank used (Pn_ADDR_BK0/1).
- The packet size information for the pipe and bank (Pn_PCKSIZE_BK0/1):

Table 32-6. Pn_PCKSIZE_BK0/1 structure

31	30:16	15	14:0
AUTO_ZLP	MULTI_PACKET_SIZE	-	BYTE_COUNT

– AUTO_ZLP: Auto zero length packet, see ”Multi packet mode for OUT pipes” on page 902.

– MULTI_PACKET_SIZE: see ”Multi packet mode and single packet mode.” on page 886.

– BYTE_COUNT: see ”Multi packet mode and single packet mode.” on page 886.

- The control and status fields for the pipe and bank (Pn_CTR_STA_BK0/1):

Table 32-7. Pn_CTR_STA_BK0/1 structure

31:19	18	17	16	15:0
Status				Control
-	UNDERF	OVERF	CRCERR	-

– UNDERF: Underflow status for isochronous/Interrupt IN transfers. This status bit is set by hardware at the current bank (where the IN packet should have been stored). When a new successful transaction occurs this bit is overwritten to zero if UPSTAX.ERRORFI has previously been cleared by software. See "Data flow error" on page 902.

– OVERF: Overflow status for isochronous/interrupt OUT transfers. This status bit is set by hardware at the current bank (where the OUT packet should have been loaded). When a new successful transaction occurs this bit is overwritten to zero if UPSTAX.ERRORFI has previously been cleared by software. See "Data flow error" on page 902.

– CRCERR: CRC error status for isochronous IN transfers. See "CRC error" on page 902.

- The control and status 1 of endpoint n (Pn_CTR_STA1):

Table 32-8. Pn_CTR_STA1 structure

31:24	23:16	15:12	11:8	7	6:0
Status		Control			
-	PERSTA	PERMAX	PEPNUM	-	PDADDR

– PERSTA: Pipe Error Status. See "PERSTA structure" table.

– PERMAX: Should be set by the user. If the Pipe Error Counter (see Figure 32-9 on page 898) is larger than PERMAX, the UPSTAX.PERRI bit is set.

– PEPNUM: Should be set by the user. Endpoint number for this pipe.

– PDADDR: Should be set by the user. Device address for this pipe.

Table 32-9. PERSTA structure

23:21	20	19	18	17	16
ERCNT	CRC16ER	TOUTER	PIDER	DAPIDER	DTGLER

This field can be cleared by software. To avoid read-modify-write issues, the user should: freeze the pipe, wait until the UPSTAX.PFREEZE is one, clear the PERSTA field in memory, and then unfreeze the pipe.

– ERCNT: Pipe Error Counter.

– CRC16ER: Is set if a CRC16 error occurs during an isochronous IN transaction.

– TOUTER: Is set if a Time-out error occurs during a USB transaction.

– PIDER: Is set if a PID error occurs during a USB transaction.

– DAPIDER: Is set if a Data PID error occurs during a USB transaction.

– DTGLER: Is set if a Data toggle error occurs during a USB transaction.

32.6.3.9 *Multi packet mode and single packet mode.*

See ”Multi packet mode and single packet mode.” on page 886 and just consider that an OUT pipe corresponds to an IN endpoint, and an IN pipe corresponds to an OUT endpoint.

32.6.3.10 *Management of control pipes*

A control transaction is composed of three stages:

- SETUP
- Data (IN or OUT)
- Status (OUT or IN)

The user has to change the pipe token according to each stage.

For control pipes only, the token is assigned a specific initial data toggle sequence:

- SETUP: Data0
- IN: Data1
- OUT: Data1

32.6.3.11 *Management of IN pipes*

- *Overview*

IN packets are sent by the USB device controller upon IN requests from the host. All the data can be read, acknowledging whether or not the bank is empty.

- *Detailed description*

The pipe and its descriptor in RAM must be pre configured.

The host can request data from the device in two modes, selected by writing to the IN Request Mode bit in the Pipe n IN Request register (UPINRQn.INMODE):

- When INMODE is written to zero, the USBC will perform INRQ IN requests before freezing the pipe.
- When INMODE is written to one, the USBC will perform IN requests as long as the pipe is not frozen by the user.

The generation of IN requests starts when the pipe is unfrozen (UPCONn.PFREEZE is zero).

When the current bank is full, the RXINI and FIFO Control (UPSTAn.FIFOCON) bits will be set simultaneously. This triggers a PnINT interrupt if the Received IN Data Interrupt Enable bit (UPCONn.RXINE) is one.

RXINI shall be cleared by software to acknowledge the interrupt. This is done by writing a one to the Received IN Data Interrupt Clear bit in the Pipe n Control Clear register (UPCONnCLR.RXINIC), which does not affect the pipe FIFO.

The user reads the byte count of the current bank from the descriptor in RAM (Pn_PCKSIZE_BK0/1.BYTE_COUNT) to know how many bytes should be read.

The user reads the IN data from the RAM and clears the FIFOCON bit to free the bank. This will also cause a switch to the next bank if the IN endpoint is composed of multiple banks. The RXINI and FIFOCON bits will be updated accordingly.

RXINI should always be cleared before clearing FIFOCON to avoid missing an RXINI event.

Figure 32-19. Example of an IN pipe with one data bank

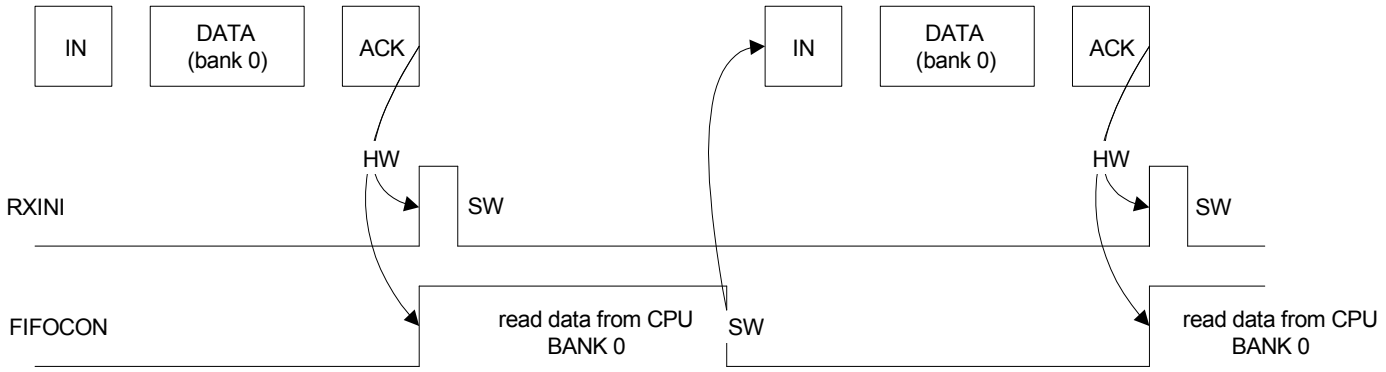
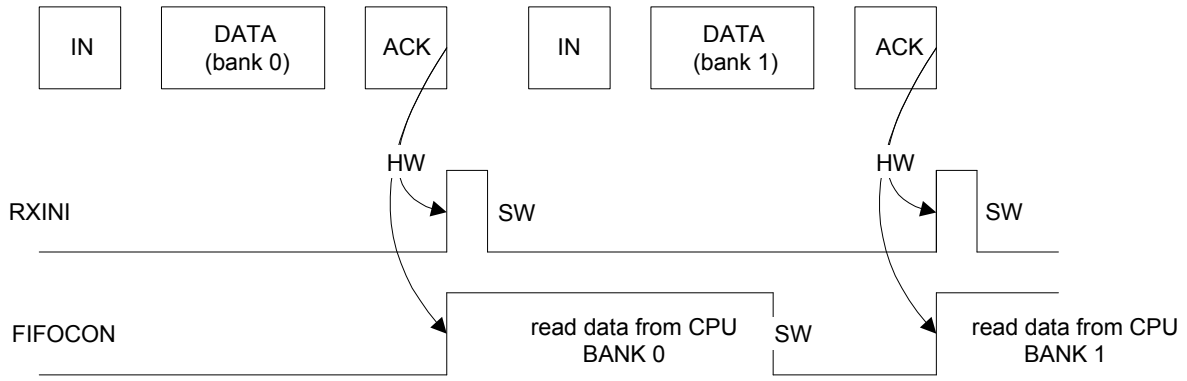


Figure 32-20. Example of an IN pipe with two data banks



- *Multi packet mode for IN pipes*

  See "Multi packet mode for OUT endpoints" on page 891 and just replace OUT endpoints with IN pipe.

32.6.3.12 *Management of OUT pipes*

- *Overview*

  OUT packets are sent by the host. All the data can be written, acknowledging whether or not the bank is full.

- *Detailed description*

  The pipe and its descriptor in RAM must be pre configured.

  When the current bank is clear, the Transmitted OUT Data Interrupt (TXOUTI) and FIFO Control (UPSTAn.FIFOCON) bits will be set simultaneously. This triggers a PnINT interrupt if the Transmitted OUT Data Interrupt Enable bit (UPCONn.TXOUTE) is one.

TXOUTI shall be cleared by software to acknowledge the interrupt. This is done by writing a one to the Transmitted OUT Data Interrupt Clear bit (UPCONnCLR.TXOUTIC), which does not affect the pipe FIFO.

The user writes the OUT data to the bank referenced to by the PEPn descriptor and allows the USBC to send the data by writing a one to the FIFO Control Clear (UPCONnCLR.FIFOCONC) bit. This will also cause a switch to the next bank if the OUT pipe is composed of multiple banks. The TXOUTI and FIFOCON bits will be updated accordingly

TXOUTI shall always be cleared before clearing FIFOCON to avoid missing an TXOUTI event.

Note that if the user decides to switch to the Suspend state (by writing a zero to UHCON.SOFE) while a bank is ready to be sent, the USBC automatically exits this state and sends the data.

Figure 32-21. Example of an OUT pipe with one data bank

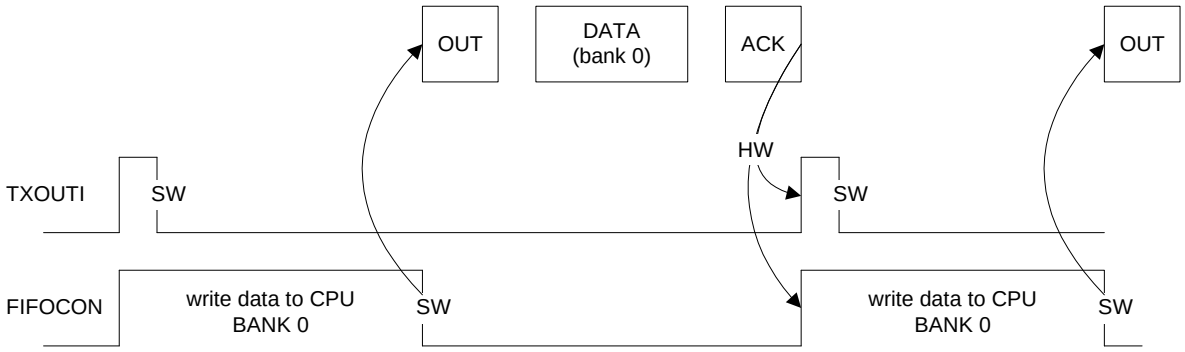


Figure 32-22. Example of an OUT pipe with two data banks and no bank switching delay

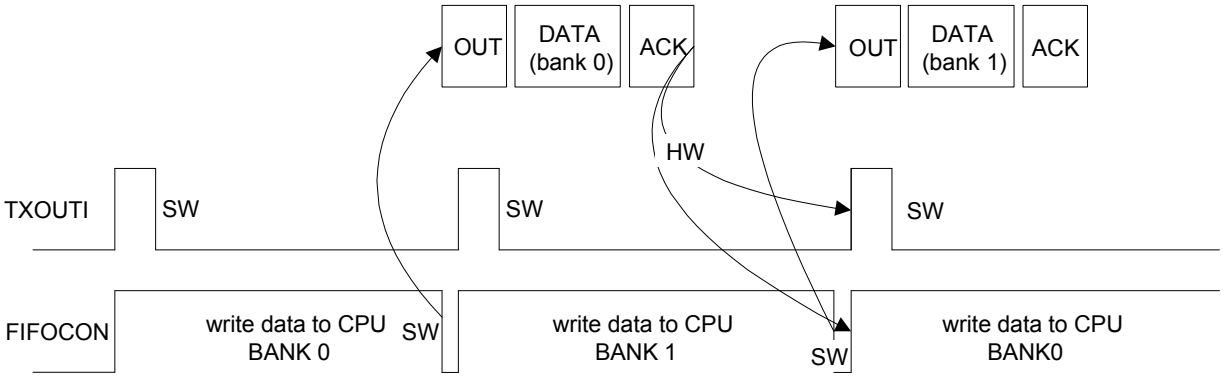
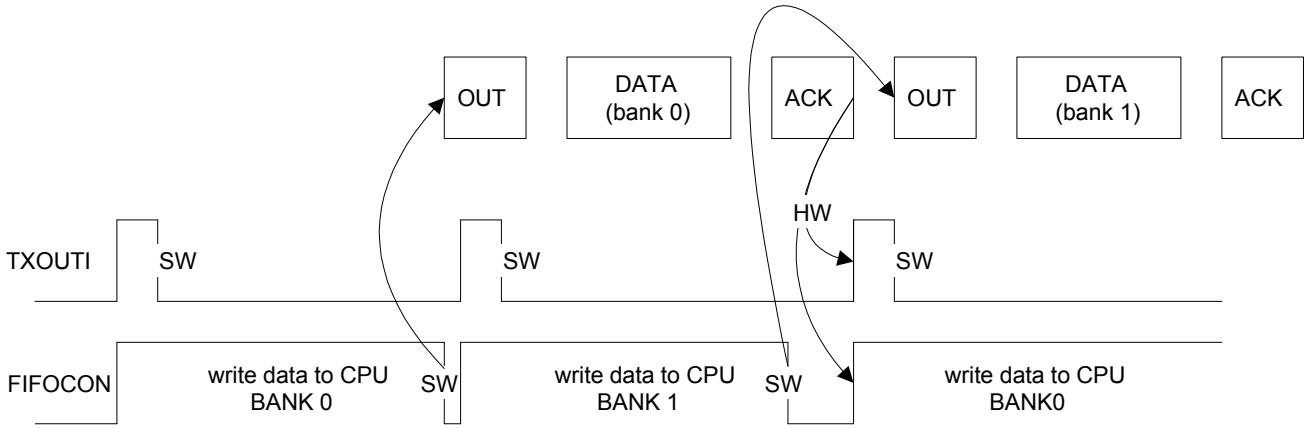


Figure 32-23. Example of an OUT pipe with two data banks and a bank switching delay



• *Multi packet mode for OUT pipes*

See section "Multi packet mode for IN endpoints" on page 889 and just replace IN endpoints with OUT pipe.

32.6.3.13 *Alternate pipe*

The user has the possibility to run sequentially several logical pipes on the same physical pipe.

Before switching pipe, the user should save the pipe context (UPCFGn, UPCONn, UPSTAn, and the pipe descriptor table).

After switching pipe, the user should restore the pipe context, current bank number, and the current data toggle by using the UPCONn.INITDTGL and UPCONn.INITBK bits.

32.6.3.14 *Data flow error*

This error exists only for isochronous and interrupt pipes for both IN and OUT directions. It sets the Errorflow Interrupt (ERRORFI) bit in UPSTAn, which triggers an PnINT interrupt if the Errorflow Interrupt Enable (ERRORFE) bit is one. The user can check the Pn_CTR_STA_BK0/1.UNDERF and OVERF bits in the pipe descriptor to see which current bank has been affected.

- An overflow can occur during an OUT stage if the host attempts to send data from an empty bank. The pipe descriptor Pn_CTR_STA_BK0/1.OVERF points out the bank from which the OUT data should have originated. If the UPSTAn.ERRORFI bit is cleared and a new transaction is successful, the Pn_CTR_STA_BK0/1.OVERF bit will be cleared.
- An underflow can occur during an IN stage if the device tries to send a packet while the bank is full. Typically this occurs when a CPU is not fast enough. The packet data is not written to the bank and is lost. The pipe descriptor Pn_CTR_STA_BK0/1.UNDERF points out which bank the OUT data was destined to. If UPSTAn.UNDERFI is zero and a new successful transaction occurs, Pn_CTR_STA_BK0/1.UNDERF will be cleared.

32.6.3.15 *CRC error*

This error exists only for isochronous IN pipes. It sets the CRC Error Interrupt bit (CRCERRI), which triggers a PnINT interrupt if the CRC Error Interrupt Enable bit (UPCONn.CRCERRE) is one.

A CRC error can occur during the IN stage if the USBC detects a corrupted packet. The IN packet will remain stored in the bank and RXINI will be set.

The user can check the Pn_CTR_STA_BK0/1.CRCERR bit in the pipe descriptor to see which current bank has been affected.

32.6.3.16 *Interrupts*

There are two kinds of host interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors not related to CPU exceptions.

- *Global interrupts*

The processing host global interrupts are:

- The Device Connection Interrupt (DCONNI)
- The Device Disconnection Interrupt (DDISCI)
- The USB Reset Sent Interrupt (RSTI)
- The Downstream Resume Sent Interrupt (RSMEDI)
- The Upstream Resume Received Interrupt (RXRSMI)
- The Host Start of Frame Interrupt (HSOFI)
- The Host Wakeup Interrupt (HWUPI)
- The Pipe n Interrupt (PnINT)

There is no exception host global interrupt.

- *Pipe interrupts*

The processing host pipe interrupts are:

- The Received IN Data Interrupt (RXINI)
- The Transmitted OUT Data Interrupt (TXOUTI)
- The Transmitted SETUP Interrupt (TXSTPI)
- The Number of Busy Banks (NBUSYBK) interrupt

The exception host pipe interrupts are:

- The Errorflow Interrupt (ERRORFI)
- The Pipe Error Interrupt (PERRI)
- The NAKed Interrupt (NAKEDI)
- The Received STALLed Interrupt (RXSTALLDI)
- The CRC Error Interrupt (CRCERRI)

32.7 User Interface

Table 32-10. USBC Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Device General Control Register	UDCON	Read/Write	0x00000100
0x0004	Device Global Interrupt Register	UDINT	Read-Only	0x00000000
0x0008	Device Global Interrupt Clear Register	UDINTCLR	Write-Only	0x00000000
0x000C	Device Global Interrupt Set Register	UDINTSET	Write-Only	0x00000000
0x0010	Device Global Interrupt Enable Register	UDINTE	Read-Only	0x00000000
0x0014	Device Global Interrupt Enable Clear Register	UDINTECLR	Write-Only	0x00000000
0x0018	Device Global Interrupt Enable Set Register	UDINTESET	Write-Only	0x00000000
0x001C	Endpoint Enable/Reset Register	UERST	Read/Write	0x00000000
0x0020	Device Frame Number Register	UDFNUM	Read-Only	0x00000000
0x0100 + n*4	Endpoint n Configuration Register	UECFGn	Read/Write	0x00000000
0x0130 + n*4	Endpoint n Status Register	UESTAn	Read-Only	0x00000100
0x0160 + n*4	Endpoint n Status Clear Register	UESTAnCLR	Write-Only	0x00000000
0x0190 + n*4	Endpoint n Status Set Register	UESTAnSET	Write-Only	0x00000000
0x01C0 + n*4	Endpoint n Control Register	UECONn	Read-Only	0x00000000
0x01F0 + n*4	Endpoint n Control Set Register	UECONnSET	Write-Only	0x00000000
0x0220 + n*4	Endpoint n Control Clear Register	UECONnCLR	Write-Only	0x00000000
0x0400	Host General Control Register	UHCON	Read/Write	0x00000000
0x0404	Host Global Interrupt Register	UHINT	Read-Only	0x00000000
0x0408	Host Global Interrupt Clear Register	UHINTCLR	Write-Only	0x00000000
0x040C	Host Global Interrupt Set Register	UHINTSET	Write-Only	0x00000000
0x0410	Host Global Interrupt Enable Register	UHINTE	Read-Only	0x00000000
0x0414	Host Global Interrupt Enable Clear Register	UHINTECLR	Write-Only	0x00000000
0x0418	Host Global Interrupt Enable Set Register	UHINTESET	Write-Only	0x00000000
0x0041C	Pipe Enable/Reset Register	UPRST	Read/Write	0x00000000
0x0420	Host Frame Number Register	UHFNUM	Read/Write	0x00000000
0x0500 + n*4	Pipe n Configuration Register	UPCFGn	Read/Write	0x00000000
0x0530 + n*4	Pipe n Status Register	UPSTAn	Read-Only	0x00000000
0x0560 + n*4	Pipe n Status Clear Register	UPSTAnCLR	Write-Only	0x00000000
0x0590 + n*4	Pipe n Status Set Register	UPSTAnSET	Write-Only	0x00000000
0x05C0 + n*4	Pipe n Control Register	UPCONn	Read-Only	0x00000000
0x05F0 +n*4	Pipe n Control Set Register	UPCONnSET	Write-Only	0x00000000
0x0620 + n*4	Pipe n Control Clear Register	UPCONnCLR	Write-Only	0x00000000
0x0650 + n*4	Pipe n IN Request Register	UPINRQn	Read/Write	0x00000001
0x0800	General Control Register	USBCON	Read/Write	0x03004000
0x0804	General Status Register	USBSTA	Read-Only	0x00000000

Table 32-10. USBC Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0808	General Status Clear Register	USBSTACLR	Write-Only	0x00000000
0x080C	General Status Set Register	USBSTASET	Write-Only	0x00000000
0x0818	IP Version Register	UVERS	Read-Only	-(1)
0x081C	IP Features Register	UFEATURES	Read-Only	-(1)
0x0820	IP PB Address Size Register	UADDRSIZE	Read-Only	-(1)
0x0824	IP Name Register 1	UNAME1	Read-Only	-(1)
0x0828	IP Name Register 2	UNAME2	Read-Only	-(1)
0x082C	USB Finite State Machine Status Register	USBFSM	Read-Only	0x00000009
0x0830	USB Descriptor address	UDESC	Read/Write	0x00000000

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

32.7.1 USB General Registers

32.7.1.1 General Control Register

Name: USBCON

Access Type: Read/Write

Offset: 0x0800

Reset Value: 0x03004000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	UIMOD	UIDE

23	22	21	20	19	18	17	16
-	UNLOCK	TIMPAGE		-	-	TIMVALUE	

15	14	13	12	11	10	9	8
USBE	FRZCLK	VBUSPO	OTGPADE	HNPREQ	SRPREQ	SRPSEL	VBUSHWC

7	6	5	4	3	2	1	0
STOE	HNPERRE	ROLEEXE	BCERRE	VBERRE	SRPE	VBUSTE	IDTE

- **UIMOD: USBC Mode**
  0: The module is in USB host mode.
  1: The module is in USB device mode.
  This bit has no effect when UIDE is one (USB_ID input pin activated).
  This bit can be written to even if USBE is zero or FRZCLK is one. Disabling the USBC (by writing a zero to the USBE bit) does not reset this bit.
- **UIDE: USB_ID Pin Enable**
  0: The USB mode (device/host) is selected from the UIMOD bit.
  1: The USB mode (device/host) is selected from the USB_ID input pin.
  This bit can be written to even if USBE is zero or FRZCLK is one. Disabling the USBC (by writing a zero to the USBE bit) does not reset this bit.
- **UNLOCK: Timer Access Unlock**
  0: The TIMPAGE and TIMVALUE fields are locked.
  1: The TIMPAGE and TIMVALUE fields are unlocked.
  The TIMPAGE and TIMVALUE fields can always be read, regardless of the UNLOCK value.
- **TIMPAGE: Timer Page**
  This field contains the page value to access a special timer register.
- **TIMVALUE: Timer Value**
  This field selects the timer value that is written to the special time register selected by TIMPAGE. See Section 32.6.1.7 for details.
- **USBE: USBC Enable**
  Writing a zero to this bit will disable the USBC, USB transceiver, and USB clock inputs. This will over-ride FRZCLK settings but not affect the value. Unless explicitly stated, all registers will become reset and read-only.
  Writing a one to this bit will enable the USBC.
  0: The USBC is disabled.

1: The USBC is enabled.
This bit can be written to even if FRZCLK is one.

- **FRZCLK: Freeze USB Clock**
  Writing a zero to this bit will enable USB clock inputs.
  Writing a one to this bit will disable USB clock inputs. The resume detection will remain active. Unless explicitly stated, all registers will become read-only.
  0: The clock inputs are enabled.
  1: The clock inputs are disabled.
  This bit can be written to even if USBE is zero.
- **VBUSPO: VBUS Polarity**
  0: The USB_VBOF output signal is in its default mode (active high).
  1: The USB_VBOF output signal is inverted (active low).
  This bit can be written even if USBE is zero or FRZCLK is one. Disabling the USBC (by writing a zero to the USBE bit) does not reset this bit.
- **OTGPADE: OTG Pad Enable**
  0: The OTG pad is disabled.
  1: The OTG pad is enabled.
  This bit can be written even if USBE is zero or FRZCLK is one. Disabling the USBC (by writing a zero to the USBE bit) does not reset this bit.
- **HNPREQ: HNP Request**
  When the controller is in device mode:
  Writing a zero to this bit has no effect.
  Writing a one to this bit will initiate a HNP (Host Negotiation Protocol).
  This bit is cleared when the controller has initiated an HNP.
  When the controller is in host mode:
  Writing a zero to this bit will reject a HNP.
  Writing a one to this bit will accept a HNP.
- **SRPREQ: SRP Request**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will initiate an SRP when the controller is in device mode.
  This bit is cleared when the controller has initiated an SRP.
- **SRPSEL: SRP Selection**
  0: Data line pulsing is selected as SRP method.
  1: VBUS pulsing is selected as SRP method.
- **VBUSHWC: VBUS Hardware Control**
  0: The hardware control over the USB_VBOF output pin is enabled. The USBC resets the USB_VBOF output pin when a VBUS problem occurs.
  1: The hardware control over the USB_VBOF output pin is disabled.
- **STOE: Suspend Time-Out Interrupt Enable**
  0: The Suspend Time-Out Interrupt (STOI) is disabled.
  1: The Suspend Time-Out Interrupt (STOI) is enabled.
- **HNPERRE: HNP Error Interrupt Enable**
  0: The HNP Error Interrupt (HNPERRI) is disabled.
  1: The HNP Error Interrupt (HNPERRI) is enabled.
- **ROLEEXE: Role Exchange Interrupt Enable**
  0: The Role Exchange Interrupt (ROLEEXI) is disabled.
  1: The Role Exchange Interrupt (ROLEEXI) is enabled.
- **BCERRE: B-Connection Error Interrupt Enable**
  0: The B-Connection Error Interrupt (BCERRI) is disabled.
  1: The B-Connection Error Interrupt (BCERRI) is enabled.
- **VBERRE: VBUS Error Interrupt Enable**
  0: The VBUS Error Interrupt (VBERRI) is disabled.

1: The VBUS Error Interrupt (VBERRI) is enabled.

- **SRPE: SRP Interrupt Enable**
  0: The SRP Interrupt (SRPI) is disabled.
  1: The SRP Interrupt (SRPI) is enabled.
- **VBUSTE: VBUS Transition Interrupt Enable**
  0: The VBUS Transition Interrupt (VBUSTI) is disabled.
  1: The VBUS Transition Interrupt (VBUSTI) is enabled.
- **IDTE: ID Transition Interrupt Enable**
  0: The ID Transition interrupt (IDTI) is disabled.
  1: The ID Transition interrupt (IDTI) is enabled.

32.7.1.2 General Status Register

Register Name: USBSTA

Access Type: Read-Only

Offset: 0x0804

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	CLKUSABLE	SPEED		VBUS	ID	VBUSRQ	-

7	6	5	4	3	2	1	0
STOI	HNPERRI	ROLEEXI	BCERRI	VBERRI	SRPI	VBUSTI	IDTI

- **CLKUSABLE: Generic Clock Usable**
  This bit is cleared when the USB generic clock is not usable.
  This bit is set when the USB generic clock (that should be 48 Mhz) is usable.
- **SPEED: Speed Status**
  This field is set according to the controller speed mode.

SPEED	Speed Status
00	full-speed mode
01	Reserved
10	low-speed mode
11	Reserved

- **VBUS: VBUS Level**
  This bit is cleared when the VBUS line level is low, even if USBE is zero.
  This bit is set when the VBUS line level is high, even if USBE is zero.
  This bit can be used in device mode to monitor the USB bus connection state of the application.
- **ID: USB_ID Pin state**
  This bit is cleared when the USB_ID level is low, even if USBE is zero.
  This bit is set when the USB_ID level is high, even if USBE is zero.
- **VBUSRQ: VBUS Request**
  0: The USB_VBOF output pin is driven low to disable the VBUS power supply generation.
  1: The USB_VBOF output pin is driven high to enable the VBUS power supply generation.
  This bit is cleared when USBSTACLR.VBUSRQC is written to one or when a VBUS error occurs while VBUSHWC is zero.
  This bit is set when USBSTASET.VBUSRQS is written to one.
  This bit should only be used in host mode.

- **STOI: Suspend Time-Out Interrupt**
  This bit is cleared when the USBSTACLR.STOIC bit is written to one.
  This bit is set when a time-out error (more than 200 ms) has been detected after a suspend. This triggers a USB interrupt if STOE is one.
  This bit should only be used in host mode.
- **HNPERRI: HNP Error Interrupt**
  This bit is cleared when the USBSTACLR.HNPERRIC bit is written to one.
  This bit is set when an error has been detected during a HNP negotiation. This triggers a USB interrupt if HNPERRE is one.
  This bit should only be used in device mode.
- **ROLEEXI: Role Exchange Interrupt**
  This bit is cleared when the USBSTACLR.ROLEEXIC bit is written to one.
  This bit is set when the USBC has successfully switched its mode because of an HNP negotiation (host to device or device to host). This triggers a USB interrupt if ROLEEXE is one.
- **BCERRI: B-Connection Error Interrupt**
  This bit is cleared when the USBSTACLR.BCERRIC bit is written to one.
  This bit is set when an error occurs during the B-connection. This triggers a USB interrupt if BCERRE is one.
  This bit should only be used in host mode.
- **VBERRI: VBUS Error Interrupt**
  This bit is cleared when the USBSTACLR.VBERRIC bit is written to one.
  This bit is set when a VBUS drop has been detected. This triggers a USB interrupt if VBERRE is one.
  This bit should only be used in host mode.
- **SRPI: SRP Interrupt**
  This bit is cleared when the USBSTACLR.SRPIC bit is written to one.
  This bit is set when an SRP has been detected. This triggers a USB interrupt if SRPE is one.
  This bit should only be used in host mode.
- **VBUSTI: VBUS Transition Interrupt**
  This bit is cleared when the USBSTACLR.VBUSTIC bit is written to one.
  This bit is set when a transition (high to low, low to high) has been detected on the USB_VBUS pad. This triggers a USB interrupt if VBUSTE is one.
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.
- **IDTI: ID Transition Interrupt**
  This bit is cleared when the USBSTACLR.IDTIC bit is written to one.
  This bit is set when a transition (high to low, low to high) has been detected on the USB_ID input pin. This triggers a USB interrupt if IDTE is one.
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.

32.7.1.3 General Status Clear Register

Register Name: USBSTACLR

Access Type: Write-Only

Offset: 0x0808

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	VBUSRQC	-

7	6	5	4	3	2	1	0
STOIC	HNPERRIC	ROLEEXIC	BCERRIC	VBERRIC	SRPIC	VBUSTIC	IDTIC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in USBSTA.
These bits always read as zero.

32.7.1.4 General Status Set Register

Register Name: USBSTASET

Access Type: Write-Only

Offset: 0x080C

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	VBUSRQS	-
7	6	5	4	3	2	1	0
STOIS	HNPERRIS	ROLEEXIS	BCERRIS	VBERRIS	SRPIS	VBUSTIS	IDTIS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in USBSTA.
These bits always read as zero.

32.7.1.5 Version Register

Register Name: UVERS

Access Type: Read-Only

Offset: 0x0818

Read Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

32.7.1.6 Features Register

Register Name: UFEATURES

Access Type: Read-Only

Offset: 0x081C

Read Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	EPTNBRMAX			

- **EPTNBRMAX: Maximal Number of pipes/endpoints**
  This field indicates the number of hardware-implemented pipes/endpoints:

32.7.1.7 Address Size Register

Register Name: UADDRSIZE

Access Type: Read-Only

Offset: 0x0820

Read Value: -

31	30	29	28	27	26	25	24
UADDRSIZE[31:24]							

23	22	21	20	19	18	17	16
UADDRSIZE[23:16]							

15	14	13	12	11	10	9	8
UADDRSIZE[15:8]							

7	6	5	4	3	2	1	0
UADDRSIZE[7:0]							

- **UADDRSIZE: IP PB Address Size**
  This field indicates the size of the PB address space reserved for the USBC IP interface.

32.7.1.8 IP Name Register 1

Register Name: UNAME1

Access Type: Read-Only

Offset: 0x0824

Read Value: -

31	30	29	28	27	26	25	24
UNAME1[31:24]							

23	22	21	20	19	18	17	16
UNAME1[23:16]							

15	14	13	12	11	10	9	8
UNAME1[15:8]							

7	6	5	4	3	2	1	0
UNAME1[7:0]							

- **UNAME1: IP Name Part One**
  This field indicates the first part of the ASCII-encoded name of the USBC IP.

32.7.1.9 IP Name Register 2

Register Name: UNAME2

Access Type: Read-Only

Offset: 0x0828

Read Value:

31	30	29	28	27	26	25	24
UNAME2[31:24]							

23	22	21	20	19	18	17	16
UNAME2[23:16]							

15	14	13	12	11	10	9	8
UNAME2[15:8]							

7	6	5	4	3	2	1	0
UNAME2[7:0]							

- **UNAME2: IP Name Part Two**
  This field indicates the second part of the ASCII-encoded name of the USBC IP.

32.7.1.10 Finite State Machine Status Register

Register Name: USBFSM

Access Type: Read-Only

Offset: 0x082C

Read Value: 0x00000009

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	DRDSTATE			

- **DRDSTATE: Dual Role Device State**
  This field indicates the state of the USBC.

DRDSTATE	Description
0	a_idle state: this is the start state for A-devices (when the ID pin is 0)
1	a_wait_vrise: In this state, the A-device waits for the voltage on VBUS to rise above the A-device VBUS Valid threshold (4.4 V).
2	a_wait_bcon: In this state, the A-device waits for the B-device to signal a connection.
3	a_host: In this state, the A-device that operates in host mode is operational.
4	a_suspend: The A-device operating as a host is in the suspend mode.
5	a_peripheral: The A-device operates as a peripheral.
6	a_wait_vfall: In this state, the A-device waits for the voltage on VBUS to drop below the A-device Session Valid threshold (1.4 V).
7	a_vbus_err: In this state, the A-device waits for recovery of the over-current condition that caused it to enter this state.
8	a_wait_discharge: In this state, the A-device waits for the data usb line to discharge (100 us).
9	b_idle: this is the start state for B-device (when the ID pin is 1). The USBC controller operates in device mode.
10	b_peripheral: In this state, the B-device acts as the peripheral.
11	b_wait_begin_hnp: In this state, the B-device is in suspend mode and waits until 3 ms before initiating the HNP protocol if requested.

DRDSTATE	Description
12	b_wait_discharge: In this state, the B-device waits for the data usb line to discharge (100 us) before becoming Host.
13	b_wait_acon: In this state, the B-device waits for the A-device to signal a connect before becoming B-Host.
14	b_host: In this state, the B-device acts as the Host.
15	b_srp_init: In this state, the B-device attempts to start a session using the SRP protocol.

32.7.1.11 USB Descriptor Address

Register Name: UDESC

Access Type: Read-Write

Offset: 0x0830

Read Value: -

31	30	29	28	27	26	25	24
UDESCA[31:24]							

23	22	21	20	19	18	17	16
UDESCA[23:16]							

15	14	13	12	11	10	9	8
UDESCA[15:8]							

7	6	5	4	3	2	1	0
UDESCA[7:0]							

- **UDESCA: USB Descriptor Address**
  This field contains the address of the USB descriptor. The three least significant bits are always zero.

32.7.2 USB Device Registers

32.7.2.1 Device General Control Register

Register Name: UDCON

Access Type: Read/Write

Offset: 0x0000

Reset Value: 0x00000100

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	GNAK	-

15	14	13	12	11	10	9	8
-	-	-	LS	-	-	RMWKUP	DETACH

7	6	5	4	3	2	1	0
ADDEN	UADD						

- **GNAK: Global NAK**
  0: Normal mode.
  1: A NAK handshake is answered for each USB transaction regardless of the current endpoint memory bank status.
- **LS: low-speed mode force**
  0: The full-speed mode is active.
  1: The low-speed mode is active.
  This bit can be written to even if USBE is zero or FRZCLK is one. Disabling the USBC (by writing a zero to the USBE bit) does not reset this bit.
- **RMWKUP: Remote wakeup**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will send an upstream resume to the host for a remote wakeup.
  This bit is cleared when the USBC receives a USB reset or once the upstream resume has been sent.
- **DETACH: Detach**
  Writing a zero to this bit will reconnect the device.
  Writing a one to this bit will physically detach the device (disconnect internal pull-up resistor from DP and DM).
- **ADDEN: Address Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will activate the UADD field (USB address).
  This bit is cleared when a USB reset is received.
- **UADD: USB Address**
  This field contains the device address.
  This field is cleared when a USB reset is received.

32.7.2.2 Device Global Interrupt Register

Register Name: UDINT

Access Type: Read-Only

Offset: 0x0004

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	EP8INT[1]	EP7INT[1]	EP6INT[1]	EP5INT[1]	EP4INT[1]
15	14	13	12	11	10	9	8
EP3INT[1]	EP2INT[1]	EP1INT[1]	EP0INT	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSM	EORSM	WAKEUP	EORST	SOF	-	SUSP

Note: 1. EPnINT bits are within the range from EP0INT to EP6INT.

- **EPnINT: Endpoint n Interrupt**
  This bit is cleared when the interrupt source is serviced.
  This bit is set when an interrupt is triggered by the endpoint n (UESTAn, UECONn). This triggers a USB interrupt if EPnINTE is one.
- **UPRSM: Upstream Resume Interrupt**
  This bit is cleared when the UDINTCLR.UPRSMC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before).
  This bit is set when the USBC sends a resume signal called “Upstream Resume”. This triggers a USB interrupt if UPRSME is one.
- **EORSM: End of Resume Interrupt**
  This bit is cleared when the UDINTCLR.EORSMC bit is written to one to acknowledge the interrupt.
  This bit is set when the USBC detects a valid “End of Resume” signal initiated by the host. This triggers a USB interrupt if EORSME is one.
- **WAKEUP: Wakeup Interrupt**
  This bit is cleared when the UDINTCLR.WAKEUPC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before) or when the Suspend (SUSP) interrupt bit is set.
  This bit is set when the USBC is reactivated by a filtered non-idle signal from the lines (not by an upstream resume). This triggers an interrupt if WAKEUPE is one.
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.
- **EORST: End of Reset Interrupt**
  This bit is cleared when the UDINTCLR.EORSTC bit is written to one to acknowledge the interrupt.
  This bit is set when a USB “End of Reset” has been detected. This triggers a USB interrupt if EORSTE is one.
- **SOF: Start of Frame Interrupt**
  This bit is cleared when the UDINTCLR.SOFC bit is written to one to acknowledge the interrupt.
  This bit is set when a USB “Start of Frame” PID (SOF) has been detected (every 1 ms). This triggers a USB interrupt if SOFE is one. The FNUM field is updated.

- **SUSP: Suspend Interrupt**

  This bit is cleared when the UDINTCLR.SUSPC bit is written to one to acknowledge the interrupt or when the Wakeup (WAKEUP) interrupt bit is set.

  This bit is set when a USB “Suspend” idle bus state has been detected for 3 frame periods (J state for 3 ms). This triggers a USB interrupt if SUSPE is one.

32.7.2.3 Device Global Interrupt Clear Register

Register Name: UDINTCLR

Access Type: Write-Only

Offset: 0x0008

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSMC	EORSMC	WAKEUPC	EORSTC	SOFC	-	SUSPC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UDINT.
These bits always read as zero.

32.7.2.4 Device Global Interrupt Set Register

Register Name: UDINTSET

Access Type: Write-Only

Offset: 0x000C

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	UPRSMS	EORSMS	WAKEUPS	EORSTS	SOFS	-	SUSPS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UDINT, which may be useful for test or debug purposes.
These bits always read as zero.

32.7.2.5 Device Global Interrupt Enable Register

Register Name: UDINTE

Access Type: Read-Only

Offset: 0x0010

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	EP8INTE[1]	EP7INTE[1]	EP6INTE[1]	EP5INTE[1]	EP4INTE[1]
15	14	13	12	11	10	9	8
EP3INTE[1]	EP2INTE[1]	EP1INTE[1]	EP0INTE	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSME	EORSME	WAKEUPE	EORSTE	SOFE	-	SUSPE

Note: 1. EPnINTE bits are within the range from EP0INTE to EP6INTE.

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared when the corresponding bit in UDINTECLR is written to one.

A bit in this register is set when the corresponding bit in UDINTESET is written to one.

32.7.2.6 Device Global Interrupt Enable Clear Register

Register Name: UDINTECLR

Access Type: Write-Only

Offset: 0x0014

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	EP8INTEC(1)	EP7INTEC(1)	EP6INTEC(1)	EP5INTEC(1)	EP4INTEC(1)

15	14	13	12	11	10	9	8
EP3INTEC(1)	EP2INTEC(1)	EP1INTEC(1)	EP0INTEC	-	-	-	-

7	6	5	4	3	2	1	0
-	UPRSMEC	EORSMEC	WAKEUPEC	EORSTEC	SOFEC	-	SUSPEC

Note: 1. EPnINTEC bits are within the range from EP0INTEC to EP6INTEC.

Writing a zero to a bit in this register has no effect.

Writing a one to a bit in this register will clear the corresponding bit in UDINTE.

These bits always read as zero.

32.7.2.7 Device Global Interrupt Enable Set Register

Register Name: UDINTESET

Access Type: Write-Only

Offset: 0x0018

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	EP8INTES(1)	EP7INTES(1)	EP6INTES(1)	EP5INTES(1)	EP4INTES(1)

15	14	13	12	11	10	9	8
EP3INTES(1)	EP2INTES(1)	EP1INTES(1)	EP0INTES	-	-	-	-

7	6	5	4	3	2	1	0
-	UPRSMES	EORSMES	WAKEUPES	EORSTES	SOFES	-	SUSPES

Note: 1. EPnINTES bits are within the range from EP0INTES to EP6INTES.
Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UDINTE.
These bits always read as zero.

32.7.2.8 Endpoint Enable/Reset Register

Register Name: UERST

Access Type: Read/Write

Offset: 0x001C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	EPEN8[1]

7	6	5	4	3	2	1	0
EPEN7[1]	EPEN6[1]	EPEN5[1]	EPEN4[1]	EPEN3[1]	EPEN2[1]	EPEN1[1]	EPEN0

- **EPENn: Endpoint n Enable**

Note: 1. EPENn bits are within the range from EPEN0 to EPEN6.

Writing a zero to this bit will disable the endpoint n (USB requests will be ignored), and resets the endpoints registers (UECFGn, UESTAn, UECONn), but not the endpoint configuration (EPBK, EPSIZE, EPDIR, EPTYPE).

Writing a one to this bit will enable the endpoint n.

0: The endpoint n is disabled.

1: The endpoint n is enabled.

32.7.2.9 Device Frame Number Register

Register Name: UDFNUM

Access Type: Read-Only

Offset: 0x0020

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
FNCERR	-	FNUM[10:5]					

7	6	5	4	3	2	1	0
FNUM[4:0]					-	-	-

- **FNCERR: Frame Number CRC Error**
  This bit is cleared upon receiving a USB reset.
  This bit is set when a corrupted frame number is received. This bit and the SOF interrupt bit are updated at the same time.
- **FNUM: Frame Number**
  This field is cleared upon receiving a USB reset.
  This field contains the 11-bit frame number information, as provided from the last SOF packet.
  FNUM is updated even if a corrupted SOF is received.

32.7.2.10 Endpoint n Configuration Register

Register Name: UECFGn, n in [0..6]

Access Type: Read/Write

Offset: 0x0100 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	EPTYPE		-	-	EPDIR

7	6	5	4	3	2	1	0
-	EPSIZE			-	EPBK	-	-

- **EPTYPE: Endpoint Type**
  This field selects the endpoint type:

EPTYPE		Endpoint Type
0	0	Control
0	1	Isochronous
1	0	Bulk
1	1	Interrupt

  This field is cleared upon receiving a USB reset.
- **EPDIR: Endpoint Direction**
  0: The endpoint direction is OUT.
  1: The endpoint direction is IN (nor for control endpoints).
  This bit is cleared upon receiving a USB reset.
- **EPSIZE: Endpoint Size**
  This field determines the size of each endpoint bank:

EPSIZE			Endpoint Size
0	0	0	8 bytes
0	0	1	16 bytes
0	1	0	32 bytes
0	1	1	64 bytes
1	0	0	128 bytes

EPSIZE			Endpoint Size
1	0	1	256 bytes
1	1	0	512 bytes
1	1	1	1024 bytes

This field is cleared upon receiving a USB reset (except for the endpoint 0).

- **EPBK: Endpoint Banks**

  This bit selects the number of banks for the endpoint:

  0: single-bank endpoint

  1: double-bank endpoint

  For control endpoints, a single-bank endpoint shall be selected.

  This field is cleared upon receiving a USB reset (except for the endpoint 0).

32.7.2.11 Endpoint n Status Register

Register Name: UESTAn, n in [0..6]

Access Type: Read-Only 0x0100

Offset: 0x0130 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	CTRLDIR	-

15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		RAMACERI	-	DTSEQ	

7	6	5	4	3	2	1	0
-	STALLEDI/ CRCERRI	-	NAKINI	NAKOUTI	RXSTPI/ ERRORFI	RXOUTI	TXINI

- **CTRLDIR: Control Direction**
  Writing a zero or a one to this bit has no effect.
  This bit is cleared after a SETUP packet to indicate that the following packet is an OUT packet.
  This bit is set after a SETUP packet to indicate that the following packet is an IN packet.
- **CURRBK: Current Bank**
  This bit is set for non-control endpoints, indicating the current bank:

CURRBK		Current Bank
0	0	Bank0
0	1	Bank1
1	0	Reserved
1	1	Reserved

This field may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

- **NBUSYBK: Number of Busy Banks**
  This field is set to indicate the number of busy banks:

NBUSYBK		Number of Busy Banks
0	0	0 (all banks free)
0	1	1
1	0	2
1	1	Reserved

For IN endpoints, this indicates the number of banks filled by the user and ready for IN transfers. When all banks are free an EPnINT interrupt will be triggered if NBUSYBKE is one.

For OUT endpoints, this indicates the number of banks filled by OUT transactions from the host. When all banks are busy an EPnINT interrupt will be triggered if NBUSYBKE is one.

- **RAMACERI: Ram Access Error Interrupt**

  This bit is cleared when the RAMACERIC bit is written to one, acknowledging the interrupt.

  This bit is set when a RAM access underflow error occurs during an IN data stage.

- **DTSEQ: Data Toggle Sequence**

  This field is set to indicate the PID of the current bank:

DTSEQ		Data Toggle Sequence
0	0	Data0
0	1	Data1
1	X	Reserved

For IN transfers, this indicates the data toggle sequence that will be used for the next packet to be sent.

For OUT transfers, this value indicates the data toggle sequence of the data received in the current bank.

- **STALLEDI: STALLed Interrupt**

  This bit is cleared when the STALLEDIC bit is written to one, acknowledging the interrupt.

  This bit is set when a STALL handshake has been sent and triggers an EPnINT interrupt if STALLEDE is one.

- **CRCERRI: CRC Error Interrupt**

  This bit is cleared when the CRCERRIC bit is written to one, acknowledging the interrupt.

  This bit is set when a CRC error has been detected in an isochronous OUT endpoint bank, and triggers an EPnINT interrupt if CRCERRE is one.

- **NAKINI: NAKed IN Interrupt**

  This bit is cleared when the NAKINIC bit is written to one, acknowledging the interrupt.

  This bit is set when a NAK handshake has been sent in response to an IN request from the host, and triggers an EPnINT interrupt if NAKINE is one.

- **NAKOUTI: NAKed OUT Interrupt**

  This bit is cleared when the NAKOUTIC bit is written to one, acknowledging the interrupt.

  This bit is set when a NAK handshake has been sent in response to an OUT request from the host, and triggers an EPnINT interrupt if NAKOUTE is one.

- **ERRORFI: Isochronous Error flow Interrupt**

  This bit is cleared when the ERRORFIC bit is written to one, acknowledging the interrupt.

  This bit is set, for isochronous IN/OUT endpoints, when an errorflow (underflow or overflow) error occurs, and triggers an EPnINT interrupt if ERRORFE is one.

  An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBC.

  An overflow can also occur during OUT stage if the host sends a packet while the bank is already full, resulting in the packet being lost. This is typically due to a CPU not being fast enough.

  This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints and it means RXSTPI for control endpoints.

- **RXSTPI: Received SETUP Interrupt**

  This bit is cleared when the RXSTPIC bit is written to one, acknowledging the interrupt and freeing the bank.

  This bit is set, for control endpoints, to signal that the current bank contains a new valid SETUP packet, and triggers an EPnINT interrupt if RXSTPE is one.

  This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints and it means UNDERFI for isochronous IN/OUT endpoints.

- **RXOUTI: Received OUT Data Interrupt**

  This bit is cleared when the RXOUTIC bit is written to one, acknowledging the interrupt. For control endpoints, it releases the bank. For other endpoint types, the user should clear the FIFOCON bit to free the bank. RXOUTI shall always be cleared before clearing FIFOCON to avoid missing an interrupt.

This bit is set, for control endpoints, when the current bank contains a bulk OUT packet (data or status stage). This triggers an EPnINT interrupt if RXOUTE is one.

This bit is set for isochronous, bulk and, interrupt OUT endpoints, at the same time as FIFOCON when the current bank is full. This triggers an EPnINT interrupt if RXOUTE is one.

This bit is inactive (cleared) for isochronous, bulk and interrupt IN endpoints.

- **TXINI: Transmitted IN Data Interrupt**

  This bit is cleared when the TXINIC bit is written to one, acknowledging the interrupt. For control endpoints, this will send the packet. For other endpoint types, the user should clear the FIFOCON to allow the USBC to send the data. TXINI shall always be cleared before clearing FIFOCON to avoid missing an interrupt.

  This bit is set for control endpoints, when the current bank is ready to accept a new IN packet. This triggers an EPnINT interrupt if TXINE is one.

  This bit is set for isochronous, bulk and interrupt IN endpoints, at the same time as FIFOCON when the current bank is free. This triggers an EPnINT interrupt if TXINE is one.

  This bit is inactive (cleared) for isochronous, bulk and interrupt OUT endpoints.

32.7.2.12 Endpoint n Status Clear Register

Register Name: UESTAnCLR, n in [0..6]

Access Type: Write-Only

Offset: 0x0160 + (n * 0x04)

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	RAMACERIC	-	-	-

7	6	5	4	3	2	1	0
-	STALLEDIC/ CRCERRIC	-	NAKINIC	NAKOUTIC	RXSTPIC/ ERRORFIC	RXOUTIC	TXINIC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UESTA.
These bits always read as zero.

32.7.2.13 Endpoint n Status Set Register

Register Name: UESTAnSET, n in [0..6]

Access Type: Write-Only

Offset: 0x0190 + (n * 0x04)

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	NBUSYBKS	RAMACERIS	-		-

7	6	5	4	3	2	1	0
-	STALLEDIS/ CRCERRIS	-	NAKINIS	NAKOUTIS	RXSTPIS/ ERRORFIS	RXOUTIS	TXINIS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UESTA.
These bits always read as zero.

32.7.2.14 Endpoint n Control Register

Register Name: UECONn, n in [0..6]

Access Type: Read-Only

Offset: 0x01C0 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	BUSY1E	BUSY0E
23	22	21	20	19	18	17	16
-	-	-	-	STALLRQ	RSTDT	-	-
15	14	13	12	11	10	9	8
-	FIFOCON	KILLBK	NBUSYBKE	RAMACERE	-	-	
7	6	5	4	3	2	1	0
-	STALLEDE/ CRCERRE	-	NAKINE	NAKOUTE	RXSTPE/ ERRORFE	RXOUTE	TXINE

- **BUSY0E: Busy Bank0 Enable**
  This bit is cleared when the BUSY0C bit is written to one.
  This bit is set when the BUSY0ES bit is written to one. This will set the bank 0 as “busy”. All transactions, except SETUP, destined to this bank will be rejected (i.e: NAK token will be answered).
- **BUSY1E: Busy Bank1 Enable**
  This bit is cleared when the BUSY1C bit is written to one.
  This bit is set when the BUSY1ES bit is written to one. This will set the bank 1 as “busy”. All transactions, except SETUP, destined to this bank will be rejected (i.e: NAK token will be answered).
- **STALLRQ: STALL Request**
  This bit is cleared when a new SETUP packet is received or when the STALLRQC bit is written to zero.
  This bit is set when the STALLRQS bit is written to one, requesting a STALL handshake to be sent to the host.
- **RSTDT: Reset Data Toggle**
  The data toggle sequence is cleared when the RSTDTS bit is written to one (i.e., Data0 data toggle sequence will be selected for the next sent (IN endpoints) or received (OUT endpoints) packet.
  This bit is always read as zero.
- **FIFOCON: FIFO Control**
  For control endpoints:
  The FIFOCON and RWALL bits are irrelevant. The software shall therefore never use them for these endpoints. When read, their value is always 0.
  For IN endpoints:
  This bit is cleared when the FIFOCONC bit is written to one, sending the FIFO data and switching to the next bank.
  This bit is set simultaneously to TXINI, when the current bank is free.
  For OUT endpoints:
  This bit is cleared when the FIFOCONC bit is written to one, freeing the current bank and switching to the next.
  This bit is set simultaneously to RXINI, when the current bank is full.

- **KILLBK: Kill IN Bank**
  This bit is cleared by hardware after the completion of the “kill packet procedure”.
  This bit is set when the KILLBKS bit is written to one, killing the last written bank.
  The user shall wait for this bit to be cleared before trying to process another IN packet.
  Caution: The bank is cleared when the “kill packet” procedure is completed by the USBC core:
  If the bank is really killed, the NBUSYBK field is decremented.
  If the bank sent instead of killed (IN transfer), the NBUSYBK field is decremented and the TXINI flag is set. This specific case can occur if an IN token comes while the user tries to kill the bank.
  Note: If two banks are ready to be sent, the above specific case will not occur, since the first bank is sent (IN transfer) while the last bank is killed.
- **NBUSYBKE: Number of Busy Banks Interrupt Enable**
  This bit is cleared when the NBUSYBKEC bit is written to zero, disabling the Number of Busy Banks interrupt (NBUSYBK).
  This bit is set when the NBUSYBKES bit is written to one, enabling the Number of Busy Banks interrupt (NBUSYBK).
- **RAMACERE: RAMACER Interrupt Enable**
  This bit is cleared when the RAMACEREC bit is written to one, disabling the RAMACER interrupt (RAMACERI).
  This bit is set when the RAMACERES bit is written to one, enabling the RAMACER interrupt (RAMACERI).
- **STALLEDE: STALLed Interrupt Enable**
  This bit is cleared when the STALLEDEC bit is written to one, disabling the STALLed interrupt (STALLEDI).
  This bit is set when the STALLEDES bit is written to one, enabling the STALLed interrupt (STALLEDI).
- **CRCERRE: CRC Error Interrupt Enable**
  This bit is cleared when the CRCERREC bit is written to one, disabling the CRC Error interrupt (CRCERRI).
  This bit is set when the CRCERRES bit is written to one, enabling the CRC Error interrupt (CRCERRI).
- **NAKINE: NAKed IN Interrupt Enable**
  This bit is cleared when the NAKINEC bit is written to one, disabling the NAKed IN interrupt (NAKINI).
  This bit is set when the NAKINES bit is written to one, enabling the NAKed IN interrupt (NAKINI).
- **NAKOUTE: NAKed OUT Interrupt Enable**
  This bit is cleared when the NAKOUTEC bit is written to one, disabling the NAKed OUT interrupt (NAKOUTI).
  This bit is set when the NAKOUTES bit is written to one, enabling the NAKed OUT interrupt (NAKOUTI).
- **RXSTPE: Received SETUP Interrupt Enable**
  This bit is cleared when the RXSTPEC bit is written to one, disabling the Received SETUP interrupt (RXSTPI).
  This bit is set when the RXSTPES bit is written to one, enabling the Received SETUP interrupt (RXSTPI).
- **ERRORFE: Errorflow Interrupt Enable**
  This bit is cleared when the ERRORFEC bit is written to one, disabling the Underflow interrupt (ERRORFI).
  This bit is set when the ERRORFES bit is written to one, enabling the Underflow interrupt (ERRORFI).
- **RXOUTE: Received OUT Data Interrupt Enable**
  This bit is cleared when the RXOUTEC bit is written to one, disabling the Received OUT Data interrupt (RXOUT).
  This bit is set when the RXOUTES bit is written to one, enabling the Received OUT Data interrupt (RXOUT).
- **TXINE: Transmitted IN Data Interrupt Enable**
  This bit is cleared when the TXINEC bit is written to one, disabling the Transmitted IN Data interrupt (TXINI).
  This bit is set when the TXINES bit is written to one, enabling the Transmitted IN Data interrupt (TXINI).

32.7.2.15 Endpoint n Control Clear Register

Register Name: UECONnCLR, n in [0..6]

Access Type: Write-Only

Offset: 0x0220 + (n * 0x04)

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	BUSY1EC	BUSY0EC
23	22	21	20	19	18	17	16
-	-	-	-	STALLRQC	-	-	-
15	14	13	12	11	10	9	8
-	FIFOCONC	-	NBUSYBKEC	RAMACEREC	-	-	-
7	6	5	4	3	2	1	0
-	STALLEDEC/ CRCERREC	-	NAKINEC	NAKOUTEC	RXSTPEC/ ERRORFEC	RXOUTEC	TXINEC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UECONn.
These bits always read as zero.

32.7.2.16 Endpoint n Control Set Register

Register Name: UECONnSET, n in [0..6]

Access Type: Write-Only

Offset: 0x01F0 + (n * 0x04)

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	BUSY1ES	BUSY0ES
23	22	21	20	19	18	17	16
-	-	-	-	STALLRQS	RSTDTS	-	-
15	14	13	12	11	10	9	8
-	-	KILLBKS	NBUSYBKES	RAMACERES	-	-	-
7	6	5	4	3	2	1	0
-	STALLEDES/ CRCERRES	-	NAKINES	NAKOUTES	RXSTPES/ ERRORFES	RXOUTES	TXINES

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UECONn.
These bits always read as zero.

32.7.3 USB Host Registers

32.7.3.1 Host General Control Register

Register Name: UHCON

Access Type: Read/Write

Offset: 0x0400

Reset Value: 0x00000000

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

-
-
- **RESUME: Send USB Resume**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will generate a USB Resume on the USB bus. This bit should only be done when the start of frame generation is enabled (SOFE bit is one).
  This bit is cleared when the USB Resume has been sent or when a USB reset is requested.
- **RESET: Send USB Reset**
  Writing a zero to this bit might be useful when a device disconnection is detected (UHINT.DDISCI is one) while a USB Reset is being sent.
  Writing a one to this bit will generate a USB Reset on the USB bus.
  This bit is cleared when the USB Reset has been sent.
- **SOFE: Start of Frame Generation Enable**
  Writing a zero to this bit will disable the SOF generation and to leave the USB bus in idle state.
  Writing a one to this bit will generate SOF on the USB bus in full speed mode and keep it alive in low speed mode.
  This bit is set when a USB reset is requested or an upstream resume interrupt is detected (UHINT.RXRSMI).

32.7.3.2 Host Global Interrupt Register

Register Name: UHINT

Access Type: Read-Only

Offset: 0x0404

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	P8INT(1)
15	14	13	12	11	10	9	8
P7INT(1)	P6INT(1)	P5INT(1)	P4INT(1)	P3INT(1)	P2INT(1)	P1INT(1)	P0INT
7	6	5	4	3	2	1	0
-	HWUPI	HSOFI	RXRSMI	RSMEDI	RSTI	DDISCI	DCONNI

Note: 1. PnINT bits are within the range from P0INT to P6INT.

- **PnINT: Pipe n Interrupt**
  This bit is cleared when the interrupt source is served.
  This bit is set when an interrupt is triggered by the endpoint n (UPSTAn). This triggers a USB interrupt if the corresponding pipe interrupt enable bit is one (UHINTE register).
- **HWUPI: Host Wakeup Interrupt**
  This bit is cleared when the HWUPIC bit is written to one.
  This bit is set when:
  - the host controller is in the suspend mode (SOFE is zero) and an upstream resume from the peripheral is detected.
  - the host controller is in the suspend mode (SOFE is zero) and a peripheral disconnection is detected.
  - the host controller is in the Idle state (VBUSRQ is zero, no VBUS is generated), and an SRP event initiated by the peripheral is detected (USBSTA.SRPI is one).
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.
- **HSOFI: Host Start of Frame Interrupt**
  This bit is cleared when the HSOFIC bit is written to one.
  This bit is set when a SOF is issued by the Host controller. This triggers a USB interrupt when HSOFE is one. When using the host controller in low speed mode, this bit is also set when a keep-alive is sent.
- **RXRSMI: Upstream Resume Received Interrupt**
  This bit is set when an Upstream Resume has been received from the Device.
  This bit is cleared when the RXRSMIC is written to one.
- **RSMEDI: Downstream Resume Sent Interrupt**
  This bit is cleared when the RSMEDIC bit is written to one.
  This bit set when a Downstream Resume has been sent to the Device.
- **RSTI: USB Reset Sent Interrupt**
  This bit is cleared when the RSTIC bit is written to one.
  This bit is set when a USB Reset has been sent to the device.

- **DDISCI: Device Disconnection Interrupt**
  This bit is cleared when the DDISCIC bit is written to one.
  This bit is set when the device has been removed from the USB bus.
- **DCONNI: Device Connection Interrupt**
  This bit is cleared when the DCONNIC bit is written to one.
  This bit is set when a new device has been connected to the USB bus.

32.7.3.3 Host Global Interrupt Clear Register

Register Name: UHINTCLR

Access Type: Write-Only

Offset: 0x0408

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	HWUPIC	HSOFIC	RXRSMIC	RSMEDIC	RSTIC	DDISCIC	DCONNIC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UHINT.
These bits always read as zero.

32.7.3.4 Host Global Interrupt Set Register

Register Name: UHINTSET

Access Type: Write-Only

Offset: 0x040C

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-		-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	HWUPIS	HSOFIS	RXRSMIS	RSMEDIS	RSTIS	DDISCIS	DCONNIS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UHINT.
These bits always read as zero.

32.7.3.5 Host Global Interrupt Enable Register

Register Name: UHINTE

Access Type: Read-Only

Offset: 0x0410

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	P8INTE[1]

15	14	13	12	11	10	9	8
P7INTE[1]	P6INTE[1]	P5INTE[1]	P4INTE[1]	P3INTE[1]	P2INTE[1]	P1INTE[1]	P0INTE

7	6	5	4	3	2	1	0
-	HWUPIE	HSOFIE	RXRSMIE	RSMEDIE	RSTIE	DDISCIE	DCONNIE

Note: 1. PnINTE bits are within the range from P0INTE to P6INTE.

- **PnINTE: Pipe n Interrupt Enable**
  This bit is cleared when the PnINTEC bit is written to one. This will disable the Pipe n Interrupt (PnINT).
  This bit is set when the PnINTES bit is written to one. This will enable the Pipe n Interrupt (PnINT).
- **HWUPIE: Host Wakeup Interrupt Enable**
  This bit is cleared when the HWUPIEC bit is written to one. This will disable the Host Wakeup Interrupt (HWUPI).
  This bit is set when the HWUPIES bit is written to one. This will enable the Host Wakeup Interrupt (HWUPI).
- **HSOFIE: Host Start of Frame Interrupt Enable**
  This bit is cleared when the HSOFIEC bit is written to one. This will disable the Host Start of Frame interrupt (HSOFI).
  This bit is set when the HSOFIES bit is written to one. This will enable the Host Start of Frame interrupt (HSOFI).
- **RXRSMIE: Upstream Resume Received Interrupt Enable**
  This bit is cleared when the RXRSMIEC bit is written to one. This will disable the Downstream Resume interrupt (RXRSMI).
  This bit is set when the RXRSMIES bit is written to one. This will enable the Upstream Resume Received interrupt (RXRSMI).
- **RSMEDIE: Downstream Resume Sent Interrupt Enable**
  This bit is cleared when the RSMEDIEC bit is written to one. This will disable the Downstream Resume interrupt (RSMEDI).
  This bit is set when the RSMEDIES bit is written to one. This will enable the Downstream Resume interrupt (RSMEDI).
- **RSTIE: USB Reset Sent Interrupt Enable**
  This bit is cleared when the RSTIEC bit is written to one. This will disable the USB Reset Sent interrupt (RSTI).
  This bit is set when the RSTIES bit is written to one. This will enable the USB Reset Sent interrupt (RSTI).
- **DDISCIE: Device Disconnection Interrupt Enable**
  This bit is cleared when the DDISCIEC bit is written to one. This will disable the Device Disconnection interrupt (DDISCI).
  This bit is set when the DDISCIES bit is written to one. This will enable the Device Disconnection interrupt (DDISCI).
- **DCONNIE: Device Connection Interrupt Enable**
  This bit is cleared when the DCONNIEC bit is written to one. This will disable the Device Connection interrupt (DCONNI).
  This bit is set when the DCONNIES bit is written to one. This will enable the Device Connection interrupt (DCONNI).

32.7.3.6 Host Global Interrupt Enable Clear Register

Register Name: UHINTECLR

Access Type: Write-Only

Offset: 0x0414

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	P8INTEC[1]

15	14	13	12	11	10	9	8
P7INTEC[1]	P6INTEC[1]	P5INTEC[1]	P4INTEC[1]	P3INTEC[1]	P2INTEC[1]	P1INTEC[1]	P0INTEC

7	6	5	4	3	2	1	0
-	HWUPIEC	HSOFIEC	RXRSMIEC	RSMEDIEC	RSTIEC	DDISCIEC	DCONNIEC

Note: 1. PnINTEC bits are within the range from P0INTEC to P6INTEC.

Writing a zero to a bit in this register has no effect.

Writing a one to a bit in this register will clear the corresponding bit in UHINTE.

These bits always read as zero.

32.7.3.7 Host Global Interrupt Enable Set Register

Register Name: UHINTESET

Access Type: Write-Only

Offset: 0x0418

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	P8INTES[1]

15	14	13	12	11	10	9	8
P7INTES[1]	P6INTES[1]	P5INTES[1]	P4INTES[1]	P3INTES[1]	P2INTES[1]	P1INTES[1]	P0INTES

7	6	5	4	3	2	1	0
-	HWUPIES	HSOFIES	RXRSMIES	RSMEDIES	RSTIES	DDISCIES	DCONNIES

Note: 1. PnINTES bits are within the range from P0INTES to P6INTES.
Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UHINT.
These bits always read as zero.

32.7.3.8 Pipe Enable/Reset Register

Register Name: UPRST

Access Type: Read/Write

Offset: 0x0041C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	PEN8[1]

7	6	5	4	3	2	1	0
PEN7[1]	PEN6[1]	PEN5[1]	PEN4[1]	PEN3[1]	PEN2[1]	PEN1[1]	PEN0

Note: 1. PENn bits are within the range from PEN0 to PEN6.

- **PENn: Pipe n Enable**

  Writing a zero to this bit will disable the pipe n, forcing the pipe to an inactive state and resetting the pipe registers (UPCFGn, UPSTAn, and UPCONn).

  Writing a one to this bit will enable the pipe n.

32.7.3.9 Host Frame Number Register

Register Name: UHFNUM

Access Type: Read/Write

Offset: 0x0420

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
FLENHIGH							

15	14	13	12	11	10	9	8
-	-	FNUM[10:5]					

7	6	5	4	3	2	1	0
FNUM[4:0]					-	-	-

- **FLENHIGH: Frame Length**
  This field contains the 8 high-order bits of the 14-bits internal frame counter (frame counter at 12 MHz, counter length is 12000 to ensure a SOF generation every 1 ms).
- **FNUM: Frame Number**
  This field contains the current SOF number.
  This field can be written by software to initialize a new frame number value. In this case, at the next SOF, the FNUM field takes its new value

32.7.3.10 Pipe n Configuration Register

Register Name: UPCFGn, n in [0..6]

Access Type: Read/Write

Offset: 0x0500 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
BINTERVAL							
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	PTYPE		-	-	PTOKEN	
7	6	5	4	3	2	1	0
-	PSIZE			-	PBK	-	-

- **BINTERVAL: bInterval parameter**
  This field corresponds to the bus access period of the pipe.
  For Interrupt pipe, this field corresponds to the desired period from 1 ms to 255 ms.
  For isochronous pipe, this field corresponds to the desired period calculated as this: $2^{(BInterval)} * 1$ ms.
  For bulk or control pipe, this field corresponds to the desired period from 1 ms to 255 ms.
  This field is cleared upon sending a USB reset.
- **PTYPE: Pipe Type**
  This field contains the pipe type.
  This field is cleared upon sending a USB reset.

PTYPE		Pipe Type
0	0	Control
0	1	Isochronous
1	0	Bulk
1	1	Interrupt

- **PTOKEN: Pipe Token**
  This field contains the endpoint token.

PTOKEN	Endpoint Direction
00	SETUP
01	IN
10	OUT
11	reserved

- **PSIZE: Pipe Size**

  This field contains the size of each pipe bank.

  This field is cleared upon sending a USB reset.

PSIZE			Endpoint Size
0	0	0	8 bytes
0	0	1	16 bytes
0	1	0	32 bytes
0	1	1	64 bytes
1	0	0	128 bytes
1	0	1	256 bytes
1	1	0	512 bytes
1	1	1	1024 bytes

- **PBK: Pipe Banks**

  This bit selects the number of banks for the pipe.

  0: single-bank pipe

  1: double bank pipe

  For control endpoints, a single-bank pipe should be selected.

  This field is cleared upon sending a USB reset.

32.7.3.11 Pipe n Status Register

Register Name: UPSTAn, n in [0..6]

Access Type: Read-Only

Offset: 0x0530 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	CURRBK	NBUSYBK		-	RAMACERI	DTSEQ	

7	6	5	4	3	2	1	0
-	RXSTALLDI/ CRCERRI	ERRORFI	NAKEDI	PERRI	TXSTPI	TXOUTI	RXINI

- **CURRBK: Current Bank**

  For non-control pipe, this field indicates the number of the current bank.

  This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll this field for an interrupt.

CURRBK		Current Bank
0	0	Bank0
0	1	Bank1

- **NBUSYBK: Number of Busy Banks**

  This field indicates the number of busy bank.

  For OUT pipe, this field indicates the number of busy bank(s), filled by the user, ready for OUT transfer. When all banks are busy, this triggers an PnINT interrupt if UPCONn.NBUSYBKE is one.

  For IN pipe, this field indicates the number of busy bank(s) filled by IN transaction from the Device. When all banks are free, this triggers an PnINT interrupt if UPCONn.NBUSYBKE is one.

NBUSYBK		Number of busy bank
0	0	All banks are free.
0	1	1 busy bank
1	0	2 busy banks
1	1	reserved

- **RAMACERI: Ram Access Error Interrupt**

  This bit is cleared when the RAMACERIC bit is written to one.

  This bit is set when a RAM access underflow error occurs during IN data stage.

- **DTSEQ: Data Toggle Sequence**
  This field indicates the data PID of the current bank.
  For OUT pipes, this field indicates the data toggle of the next packet that will be sent.
  For IN pipes, this field indicates the data toggle of the received packet stored in the current bank.

DTSEQ		Data toggle sequence
0	0	Data0
0	1	Data1
1	0	reserved
1	1	reserved

- **RXSTALLDI: Received STALLed Interrupt**
  This bit is cleared when the RXSTALLDIC bit is written to one.
  This bit is set, for all endpoints (except isochronous), when a STALL handshake has been received on the current bank of the pipe. The pipe is automatically frozen. This triggers an interrupt if the RXSTALLE bit is one.
- **CRCERRI: CRC Error Interrupt**
  This bit is cleared when the CRCERRIC bit is written to one.
  This bit is set, for isochronous endpoint, when a CRC error occurs on the current bank of the pipe. This triggers an interrupt if the TXSTPE bit is one.
- **ERRORFI: Errorflow Interrupt**
  This bit is cleared when the ERRORFIC bit is written to one.
  This bit is set:
  - for isochronous and interrupt IN/OUT pipes, when an error flow occurs. This triggers an interrupt if the ERRORFIE bit is one.
  - for isochronous or interrupt OUT pipes, when a transaction underflow occurs in the current pipe. i.e, the pipe can't send the OUT data packet in time because the current bank is not ready.
  - for isochronous or interrupt IN pipes, when a transaction flow error occurs in the current pipe. i.e, the current bank of the pipe is not free when a new IN USB packet is received. This packet is not stored in the bank. For interrupt pipes, the overflowed packet is ACKed to respect the USB standard.
- **NAKEDI: NAKed Interrupt**
  This bit is cleared when the NAKEDIC bit is written to one.
  This bit is set when a NAK has been received on the current bank of the pipe. This triggers an interrupt if the NAKEDE bit is one.
- **PERRI: Pipe Error Interrupt**
  This bit is cleared when the PERRIC bit is written to one.
  This bit is set when an error occurs on the current bank of the pipe. This triggers an interrupt if the PERRE bit is set. Refers to the PERSTA structure of the pipe descriptor (Figure 32-9) to determine the source of the error.
- **TXSTPI: Transmitted SETUP Interrupt**
  This bit is cleared when the TXSTPIC bit is written to one.
  This bit is set, for Control endpoints, when the current SETUP bank is free and can be filled. This triggers an interrupt if the TXSTPE bit is one.
- **TXOUTI: Transmitted OUT Data Interrupt**
  This bit is cleared when the TXOUTIC bit is written to one.
  This bit is set when the current OUT bank is free and can be filled. This triggers an interrupt if the TXOUTE bit is one.
- **RXINI: Received IN Data Interrupt**
  This bit is cleared when the RXINIC bit is written to one.
  This bit is set when a new USB message is stored in the current bank of the pipe. This triggers an interrupt if the RXINE bit is one.

32.7.3.12 Pipe n Status Clear Register

Register Name: UPSTAnCLR, n in [0..6]

Access Type: Write-Only

Offset: 0x0560 + (n * 0x04)

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	RAMACERIC	-	-

7	6	5	4	3	2	1	0
-	RXSTALLDIC/ CRCERRIC	ERRORFIC	NAKEDIC	PERRIC	TXSTPIC	TXOUTIC	RXINIC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UPSTAn.
These bits always read as zero.

32.7.3.13 *Pipe n Status Set Register*

Register Name: UPSTAnSET, n in [0..6]

Access Type: Write-Only

Offset: 0x0590 + (n * 0x04)

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	RAMACERIS	-	-

7	6	5	4	3	2	1	0
-	RXSTALLDIS/ CRCERRIS	ERRORFIC	NAKEDIS	PERRIS	TXSTPIS	TXOUTIS	RXINIS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UPSTAn.
These bits always read as zero.

32.7.3.14 Pipe n Control Register

Register Name: UPCONn, n in [0..6]

Access Type: Read-Only

Offset: 0x05C0 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	INITBK	INITDTGL	PFREEZE	-
15	14	13	12	11	10	9	8
-	FIFOCON	-	NBUSYBKE	-	RAMACERE	-	-
7	6	5	4	3	2	1	0
-	RXSTALLDE/ CRCERRE	ERRORFIE	NAKEDE	PERRE	TXSTPE	TXOUTE	RXINE

- **INITBK: Bank Initialization**
  This bit is always read as zero.
  If the user writes a one to the INITBKC bit, this will set the current bank to Bank0 value for the current pipe.
  If the user writes a one to the INITBKS bit, this will set the current bank to Bank1 value for the current pipe.
  This may be useful to restore a pipe to manage alternate pipes on the same physical pipe.
- **INITTGL: Data Toggle Initialization**
  This bit is always read as zero.
  If the user writes a one to the INITTGLC bit, this will set the Data toggle to Data0 value for the current pipe.
  If the user writes a one to the INITTGLS bit, this will set the Data toggle to Data1 value for the current pipe.
  This may be useful to restore a pipe to manage alternate pipes on the same physical pipe.
- **PFREEZE: Pipe Freeze**
  This bit is cleared when the PFREEZEC bit is written to one. This will enable the pipe request generation.
  This bit is set when the PFREEZES bit is written to one or when the pipe is not configured or when a STALL handshake has been received on this pipe, or when INRQ In requests have been processed, or after a pipe Enable (UPRST.PEN rising). This will freeze the pipe requests generation.
  If the user clears the PFRFEEZEC bit while a transaction is on going on the USB bus, the transaction will be properly completed and then the PFREEZE bit will be cleared.
- **FIFOCON: FIFO Control**
  For OUT and SETUP pipes:
  This bit is cleared when the FIFOCONC bit is written to one. This will send the FIFO data and switch the bank.
  This bit is set when the current bank is free, at the same time than TXOUTI or TXSTPI.
  For IN pipes:
  This bit is cleared when the FIFOCONC bit is written to one. This will free the current bank and switch to the next bank.
  This bit is set when a new IN message is stored in the current bank, at the same time than RXINI.
- **NBUSYBKE: Number of Busy Banks Interrupt Enable**
  This bit is cleared when the NBUSYBKEC bit is written to one. This will disable the Transmitted IN Data interrupt (NBUSYBKE).

This bit is set when the NBUSYBKES bit is written to one.This will enable the Transmitted IN Data interrupt (NBUSYBKE).

- **RAMACERE: Ram Access Error Interrupt Enable**
  This bit is cleared when the NBUSYBKEC bit is written to one. This will disable the Transmitted IN Data interrupt (NBUSYBKE).
  This bit is set when the NBUSYBKES bit is written to one.This will enable the Transmitted IN Data interrupt (NBUSYBKE).
- **RXSTALLDE: Received STALLed Interrupt Enable**
  This bit is cleared when the RXSTALLDEC bit is written to one. This will disable the Transmitted IN Data interrupt (RXSTALLDE).
  This bit is set when the RXSTALLDES bit is written to one. This will enable the Transmitted IN Data interrupt (RXSTALLDE).
- **CRCERRE: CRC Error Interrupt Enable**
  This bit is cleared when the CRCERREC bit is written to one. This will disable the Transmitted IN Data interrupt (CRCERRE).
  This bit is set when the CRCERRES bit is written to one. This will enable the Transmitted IN Data interrupt (CRCERRE).
- **ERRORFIE: Errorflow Interrupt Enable**
  This bit is cleared when the ERRORFIEC bit is written to one. This will disable the Transmitted IN Data interrupt (OVERFIE).
  This bit is set when the ERRORFIES bit is written to one. This will enable the Transmitted IN Data interrupt (OVERFIE).
- **NAKEDE: NAKed Interrupt Enable**
  This bit is cleared when the NAKEDEC bit is written to one. This will disable the Transmitted IN Data interrupt (NAKEDE).
  This bit is set when the NAKEDES bit is written to one. This will enable the Transmitted IN Data interrupt (NAKEDE).
- **PERRE: Pipe Error Interrupt Enable**
  This bit is cleared when the PERREC bit is written to one. This will disable the Transmitted IN Data interrupt (PERRE).
  This bit is set when the PERRES bit is written to one. This will enable the Transmitted IN Data interrupt (PERRE).
- **TXSTPE: Transmitted SETUP Interrupt Enable**
  This bit is cleared when the TXSTPEC bit is written to one. This will disable the Transmitted IN Data interrupt (TXSTPE).
  This bit is set when the TXSTPES bit is written to one. This will enable the Transmitted IN Data interrupt (TXSTPE).
- **TXOUTE: Transmitted OUT Data Interrupt Enable**
  This bit is cleared when the TXOUTEC bit is written to one. This will disable the Transmitted IN Data interrupt (TXOUTE).
  This bit is set when the TXOUTES bit is written to one. This will enable the Transmitted IN Data interrupt (TXOUTE).
- **RXINE: Received IN Data Interrupt Enable**
  This bit is cleared when the RXINEC bit is written to one. This will disable the Transmitted IN Data interrupt (RXINE).
  This bit is set when the RXINES bit is written to one. This will enable the Transmitted IN Data interrupt (RXINE).

32.7.3.15 Pipe n Control Set Register

Register Name: UPCONnSET, n in [0..6]

Access Type: Write-Only

Offset: 0x05F0 + (n * 0x04)

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	INITBKS	INITDTGLS	PFREEZES	-

15	14	13	12	11	10	9	8
-	-	-	NBUSYBKES	-	-	-	-

7	6	5	4	3	2	1	0
-	RXSTALLDES/ CRCERRES	ERRORFIES	NAKEDES	PERRES	TXSTPES	TXOUTES	RXINES

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UPCONn.
These bits always read as zero.

32.7.3.16 Pipe n Control Clear Register

Register Name: UPCONnCLR, n in [0..6]

Access Type: Write-Only

Offset: 0x0620 + (n * 0x04)

Read Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	INITBKC	INITDTGLC	PFREEZEC	-
15	14	13	12	11	10	9	8
-	FIFOCONC	-	NBUSYBKEC	-	-	-	-
7	6	5	4	3	2	1	0
-	RXSTALLDEC/ CRCERREC	ERRORFIEC	NAKEDEC	PERREC	TXSTPEC	TXOUTEC	RXINEC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UPCONn.
These bits always read as zero.

32.7.3.17 Pipe n IN Request Register

Register Name: UPINRQn, n in [0..6]

Access Type: Read/Write

Offset: 0x0650 + (n * 0x04)

Reset Value: 0x00000001

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	INMODE

7	6	5	4	3	2	1	0
INRQ							

- **INMODE: IN Request Mode**
  Writing a zero to this bit will perform a pre-defined number of IN requests. This number is the INRQ field.
  Writing a one to this bit will allow the USBC to perform infinite IN requests when the pipe is not frozen.
- **INRQ: IN Request Number before Freeze**
  This field contains the number of IN transactions before the USBC freezes the pipe. The USBC will perform INRQ IN requests before freezing the pipe. This counter is automatically decreased by 1 each time an IN request has been successfully performed.
  This register has no effect when the INMODE bit is 0 (infinite IN requests generation till the pipe is not frozen).

32.8 Module Configuration

The specific configuration for each USBC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 32-11. MODULE Clock Name

Module name	Clock Name	Description
USBC	CLK_USBC_HSB	HSB clock
	CLK_USBC_PB	Peripheral Bus clock from the PBB clock domain
	GCLK_USBC	The generic clock used for the USBC is GCLK0

Table 32-12. Register Reset Values

Register	Reset Value
UVERS	0x00000210
UFEATURES	7
UADDRSIZE	0x1000
UNAME1	0x48555342
UNAME2	0x004F5447

33. Pulse Width Modulation Controller (PWM)

Rev. 5.0.1.0

33.1 Features

- 4 channels
- Common clock generator providing thirteen different clocks
  - A modulo n counter providing eleven clocks
  - Two independent linear dividers working on modulo n counter outputs
  - High frequency asynchronous clocking mode
- Independent channels
  - Independent 20-bit counter for each channel
  - Independent complementary outputs with 16-bit dead-time generator (also called dead-band or non-overlapping time) for each channel
  - Independent enable disable command for each channel
  - Independent clock selection for each channel
  - Independent period, duty-cycle and dead-time for each channel
  - Independent double buffering of period, duty-cycle and dead-times for each channel
  - Independent programmable selection of the output waveform polarity for each channel
  - Independent programmable center or left aligned output waveform for each channel
  - Independent output override for each channel
- 2 2-bit Gray up/down channels for stepper motor control
- Synchronous channel mode
  - Synchronous channels share the same counter
  - Mode to update the synchronous channels registers after a programmable number of periods
  - Synchronous channels supports connection with peripheral DMA controller which offers buffer transfer without processor intervention to update duty-cycle values
- 2 independent events lines intended to synchonize ADC conversions
- 8 comparison units intended to generate interrupts, pulses on event lines and PDC tranfer requests
- 5 programmable fault inputs providing an asynchronous protection of PWM outputs
- Write-Protect registers

33.2 Overview

The PWM Controller (PWM) controls 4 channels independently. Each channel controls two complementary square output waveforms. Characteristics of the output waveforms such as period, duty-cycle, polarity and dead-times (also called dead-bands or non-overlapping times) are configured through the user interface. Each channel selects and uses one of the clocks provided by the clock generator. The clock generator provides several clocks resulting from the division of the PWM internal clock (CCK). This internal clock can be driven either by the master clock (CLK_PWM) or by the generic clock (GCLK).

All PWM accesses are made through registers mapped on the peripheral bus. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period, the duty-cycle or the dead-times.

Channels can be linked together as synchronous channels to be able to update their duty-cycle or dead-times at the same time.

The update of duty-cycles of synchronous channels can be performed by the Peripheral DMA Controller Channel (PDCA) which offers buffer transfer without processor Intervention.

The PWM provides 8 independent comparison units capable to compare a programmed value to the counter of the synchronous channels (counter of channel 0). These comparisons are intended to generate software interrupts, to trigger pulses on the 2 independent event lines (in order to synchronize ADC conversions with a lot of flexibility independently of the PWM outputs), and to trigger PDCA transfer requests.

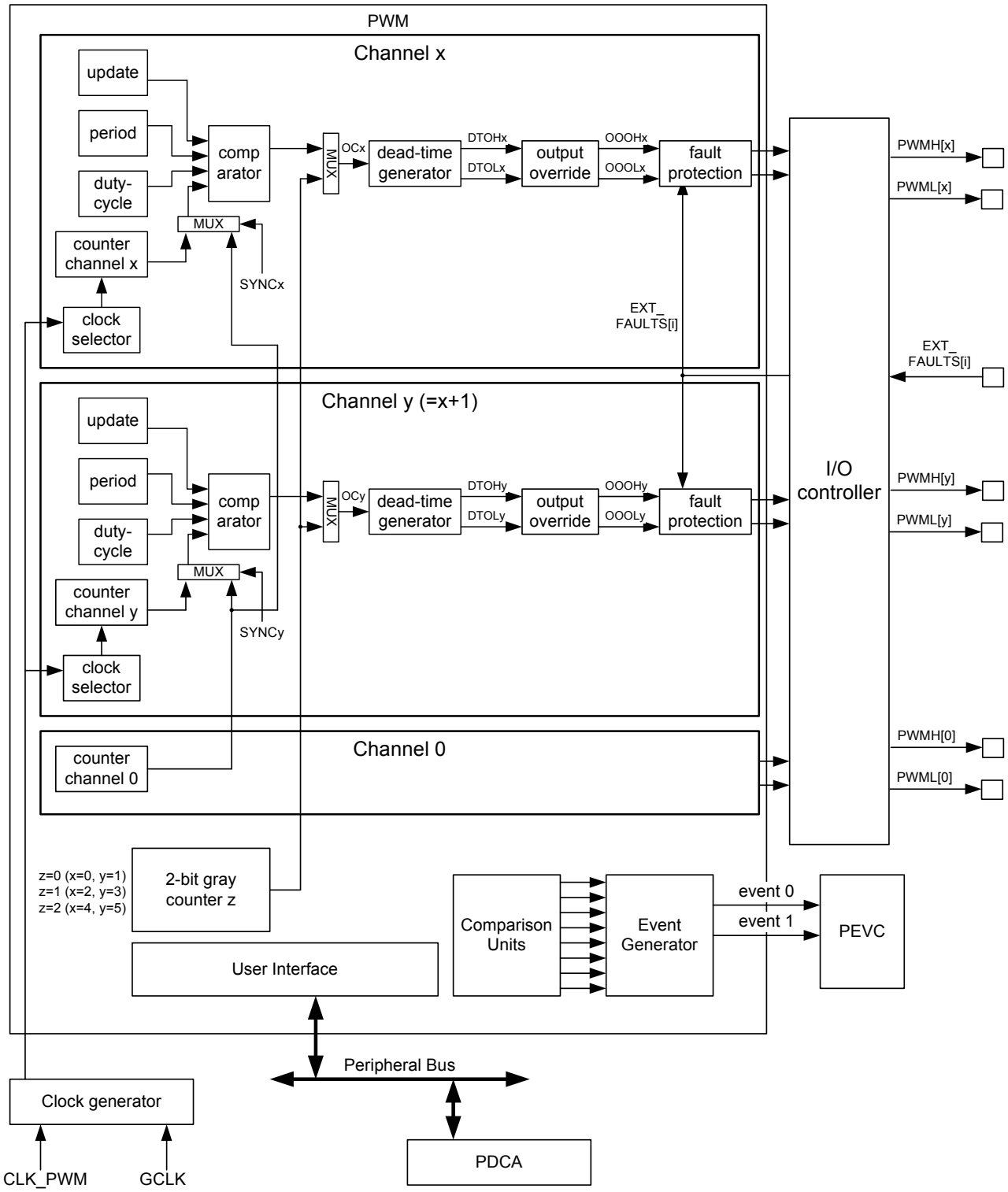
The PWM outputs can be overridden synchronously or asynchronously to their channel counter.

The PWM Controller provide a fault protection mechanism with 5 fault inputs, capable to detect a fault condition and to override the PWM outputs asynchronously.

For safety usage, some control registers are write-protected.

33.3 Block Diagram

Figure 33-1. Pulse Width Modulation Controller Block Diagram



33.4 I/O Lines Description

Each channel outputs two complementary external I/O lines.

Table 33-1. I/O Line Description

Name	Description	Type
PWMHx	PWM Waveform Output High for channel x	Output
PWMLx	PWM Waveform Output Low for channel x	Output
EXT_FAULTSx	PWM Fault Input x	Input

33.5 Product Dependencies

33.5.1 I/O Lines

The pins used for interfacing the PWM may be multiplexed with the I/O Controller lines. The programmer must first program the I/O controller to assign the desired PWM pins to their peripheral function. If I/O lines of the PWM are not used by the application, they can be used for other purposes by the I/O controller.

33.5.2 Clocks

The clock of the PWM (CLK_PWM) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the PWM before disabling the clock, to avoid freezing the PWM in an undefined state.

The PWM counters can be fed by a Generic Clock (GCLK). This is a high frequency clock which is asynchronous to CLK_PWM.

33.5.3 Interrupts

The PWM interrupt line is connected to the Interrupt Controller. Using the PWM interrupt requires the interrupt controller to be programmed first.

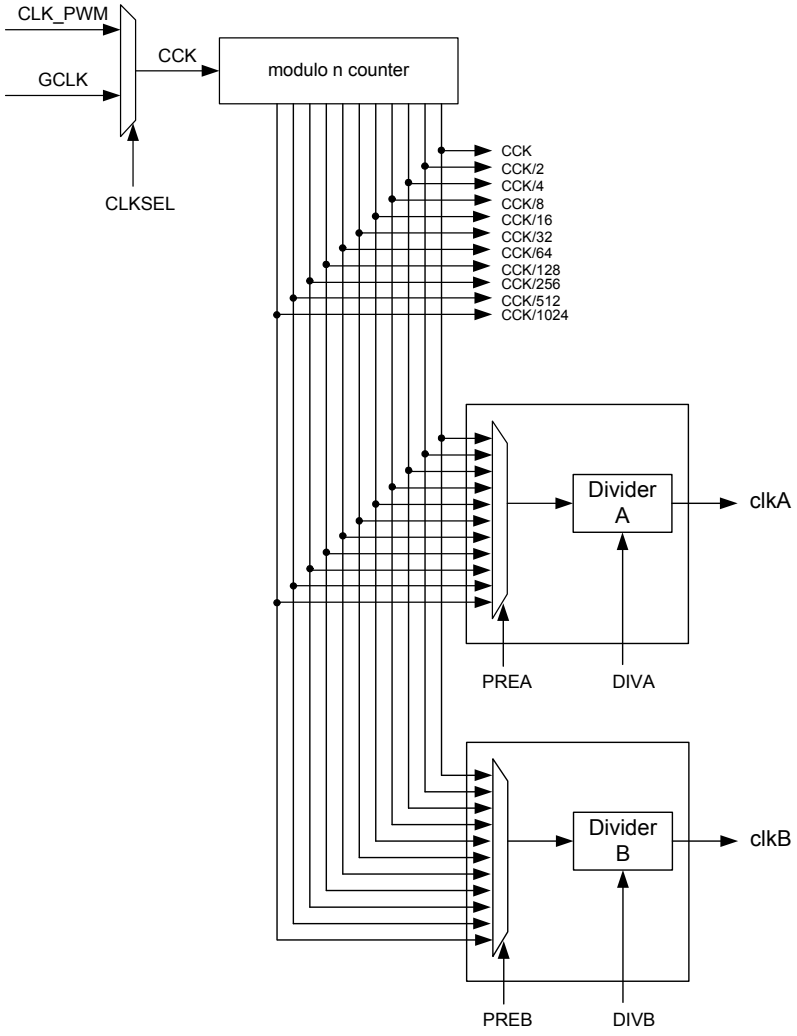
33.6 Functional Description

The PWM Controller is primarily composed of a clock generator module and 4 channels.

- The clock generator module provides 13 clocks. Its source clock is chosen according to the CLKSEL bit in the Clock Register (CLK). It allows to select:
  - CLK_PWM: the master clock (clock of the peripheral bus to which the PWM is connected)
  - GCLK: the generic clock (high frequency clock which is asynchronous to CLK_PWM)
- Each channel can independently choose one of the clock generator outputs.
- Each channel generates an output waveform with attributes that can be defined independently for each channel through the user interface registers.

33.6.1 PWM Clock Generator

Figure 33-2. Functional View of the Clock Generator Block Diagram



The PWM internal clock (named CCK and driven either by CLK_PWM or by GCLK) is divided in the clock generator module to provide different clocks available for all channels. Each channel can independently select one of the divided clocks.

The selection of the source clock of the PWM counters is made by the CLKSEL bit in the CLK Register. In asynchronous clocking mode (CLKSEL=1, GCLK selected), the PWM counters and the prescaler allow running the CPU from any clock source while the prescaler is operating on a faster clock (GCLK).

The clock generator is divided in three blocks:

- a modulo n counter which provides 11 clocks: F_{CCK}, $F_{CCK}/2$, $F_{CCK}/4$, $F_{CCK}/8$, $F_{CCK}/16$, $F_{CCK}/32$, $F_{CCK}/64$, $F_{CCK}/128$, $F_{CCK}/256$, $F_{CCK}/512$, $F_{CCK}/1024$
- two linear dividers (1, 1/2, 1/3,... 1/255) that provide two separate clocks: clkA and clkB

Each linear divider can independently divide one of the clocks of the modulo n counter. The selection of the clock to be divided is made according to the PREA (PREB) field of the PWM Clock register (PWM_CLK). The resulting clock clkA (clkB) is the clock selected divided by DIVA (DIVB) field value.

After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) are set to 0. This implies that after reset clkA (clkB) are turned off.

At reset, all clocks provided by the modulo n counter are turned off except clock “CCK”. This situation is also true when the PWM master clock is turned off through the Power Management Controller.

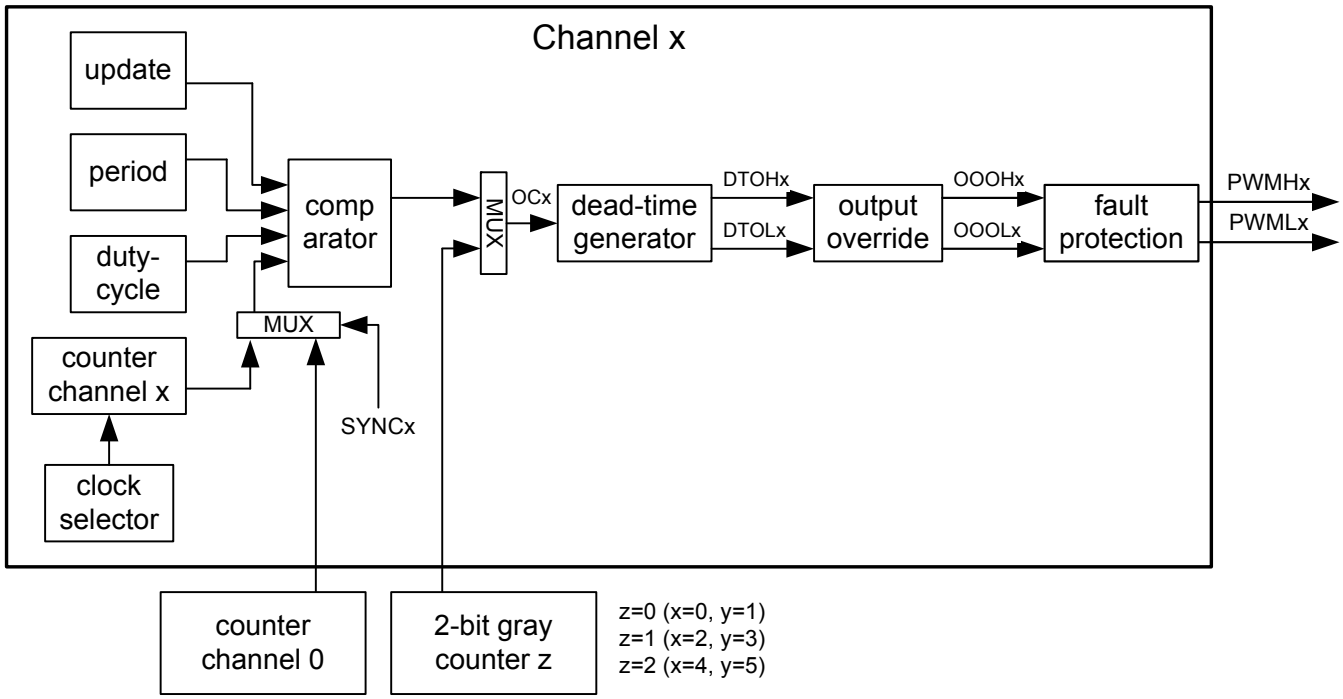
CAUTION:

- Before using the PWM, the programmer must first enable the PWM clock in the Power Manager (PM).
- The master clock frequency (CLK_PWM) must be lower than half of the generic clock frequency (GCLK) due to the synchronization mechanism between both clock domains.
- After selecting a new PWM input clock (written CLKSEL to a new value), no write in any PWM registers must be attempted before a delay of 2 master clock periods (CLK_PWM). This is the time needed by the PWM to switch the source of the internal clock (CCK).

33.6.2 PWM Channel

33.6.2.1 Block Diagram

Figure 33-3. Functional View of the Channel Block Diagram



Each of the 4 channels is composed of six blocks:

- A clock selector which selects one of the clocks provided by the clock generator (described in Section 33.6.1 on page 969).
- A counter clocked by the output of the clock selector. This counter is incremented or decremented according to the channel configuration and comparators matches. The size of the counter is 20 bits.
- A comparator used to compute the OCx output waveform according to the counter value and the configuration. The counter value can be the one of the channel counter or the one of the channel 0 counter according to SYNCx bit in the "Sync Channels Mode Register" on page 1007 (SCM).
- A 2-bit configurable gray counter enables the stepper motor driver. One gray counter drives 2 channels.
- A dead-time generator providing two complementary outputs (DTOHx/DTOLx) which allows to drive external power control switches safely.
- An output override block that can force the two complementary outputs to a programmed value (OOOHx/OOOLx).
- An asynchronous fault protection mechanism that has the highest priority to override the two complementary outputs in case of fault detection (PWMHx/PWMLx).

33.6.2.2 Comparator

The comparator continuously compares its counter value with the channel period defined by CPRD in the "Channel Period Register" on page 1041 (CPRDx) and the duty-cycle defined by CDTY in the "Channel Duty Cycle Register" on page 1039 (CDTYx) to generate an output signal OCx accordingly.

The different properties of the waveform of the output OCx are:

- the ***clock selection***. The channel counter is clocked by one of the clocks provided by the clock generator described in the previous section. This channel parameter is defined in the CPRE field of the "Channel Mode Register" on page 1037 (CMRx). This field is reset at 0.
- the ***waveform period***. This channel parameter is defined in the CPRD field of the CPRDx register.
  - If the waveform is left aligned, then the output waveform period depends on the counter source clock and can be calculated:
  By using the PWM internal clock (CCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024), the resulting period formula will be:

  $$\frac{(X \times CPRD)}{CCK}$$

  By using the PWM internal clock (CCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

  $$\frac{(CRPD \times DIVA)}{CCK} \text{ or } \frac{(CRPD \times DIVB)}{CCK}$$

  If the waveform is center aligned then the output waveform period depends on the counter source clock and can be calculated:
  By using the PWM internal clock (CCK) divided by an X given prescaler value
  (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

  $$\frac{(2 \times X \times CPRD)}{CCK}$$

  By using the PWM internal clock (CCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

  $$\frac{(2 \times CPRD \times DIVA)}{CCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{CCK}$$

- the ***waveform duty-cycle***. This channel parameter is defined in the CDTY field of the CDTYx register.
  If the waveform is left aligned then:

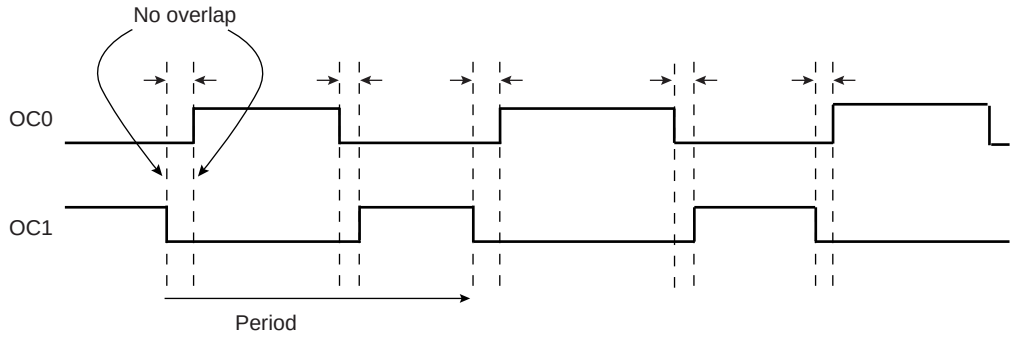
  $$\text{duty cycle} = \frac{(period - 1/\text{fchannel\_x\_clock} \times CDTY)}{period}$$

  If the waveform is center aligned, then:

  $$\text{duty cycle} = \frac{((period/2) - 1/\text{fchannel\_x\_clock} \times CDTY))}{(period/2)}$$

- the ***waveform polarity.*** At the beginning of the period, the signal can be at high or low level. This property is defined in the CPOL field of the CMRx register. By default the signal starts by a low level.
- the ***waveform alignment***. The output waveform can be left or center aligned. Center aligned waveforms can be used to generate non overlapped waveforms. This property is defined in the CALG field of the CMRx register. The default mode is left aligned.

Figure 33-4. Non Overlapped Center Aligned Waveforms



Note: 1. See Figure 33-5 on page 974 for a detailed description of center aligned waveforms.

When center aligned, the channel counter increases up to CPRD and decreases down to 0. This ends the period.

When left aligned, the channel counter increases up to CPRD and is reset. This ends the period.

Thus, for the same CPRD value, the period for a center aligned channel is twice the period for a left aligned channel.

Waveforms are fixed at 0 when:

- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

Waveforms are fixed at 1 (once the channel is enabled) when:

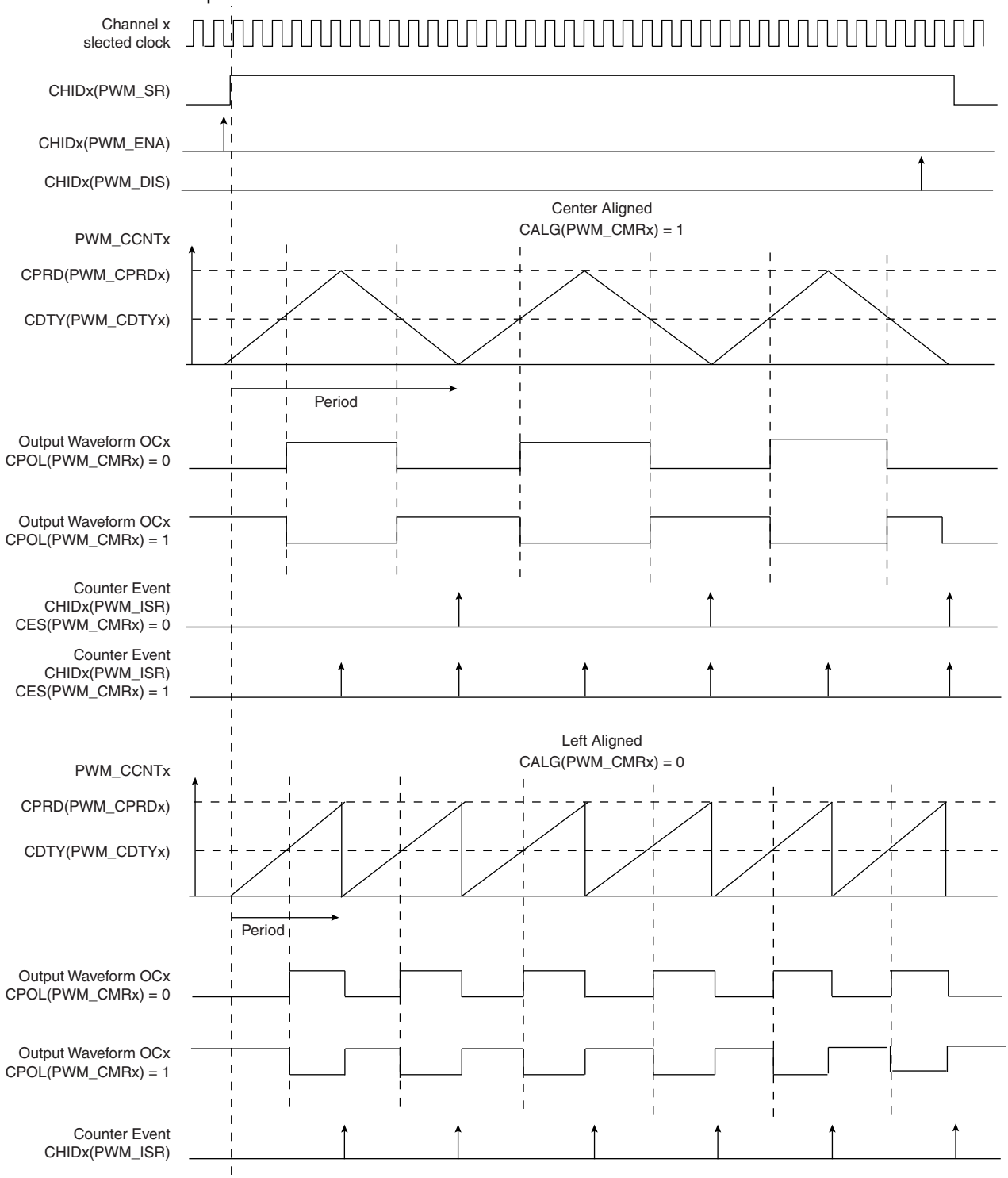
- CDTY = 0 and CPOL = 0
- CDTY = CPRD and CPOL = 1

The waveform polarity must be written before enabling the channel. This immediately affects the channel output level. Changes on channel polarity are not taken into account while the channel is enabled.

Besides generating output signals OCx, the comparator generates interrupts in function of the counter value. When the output waveform is left aligned, the interrupt occurs at the end of the counter period. When the output waveform is center aligned, the CES bit of the CMRx register defines when the channel counter interrupt occurs. If CES is set to 0, the interrupt occurs at the end of the counter period. If CES is set to 1, the interrupt occurs at the end of the counter period and at half of the counter period.

Figure 33-5 on page 974 illustrates the counter interrupts in function of the configuration.

Figure 33-5. Waveform Properties



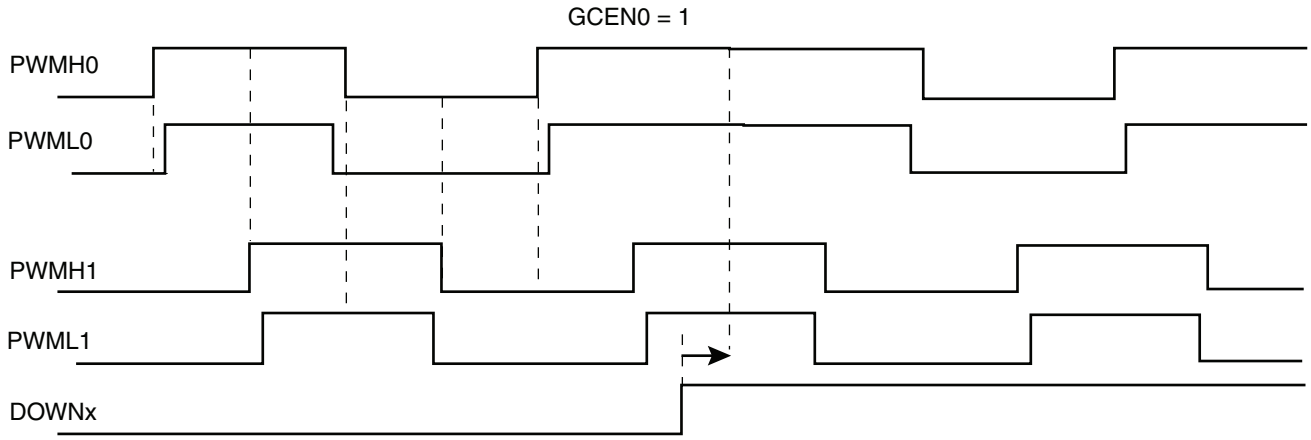
33.6.2.3 2-bit Gray Up/Down Counter for Stepper Motor

It is possible to configure a couple of channels to provide a 2-bit gray count waveform on 2 outputs. Dead-Time generator and other downstream logic can be configured on these channels.

Up or down count mode can be configured on-the-fly by the SMMR register.

When GCEN0 is written to 1, channels 0 and 1 outputs are driven with gray counter.

Figure 33-6. 2-bit Gray Up/Down Counter



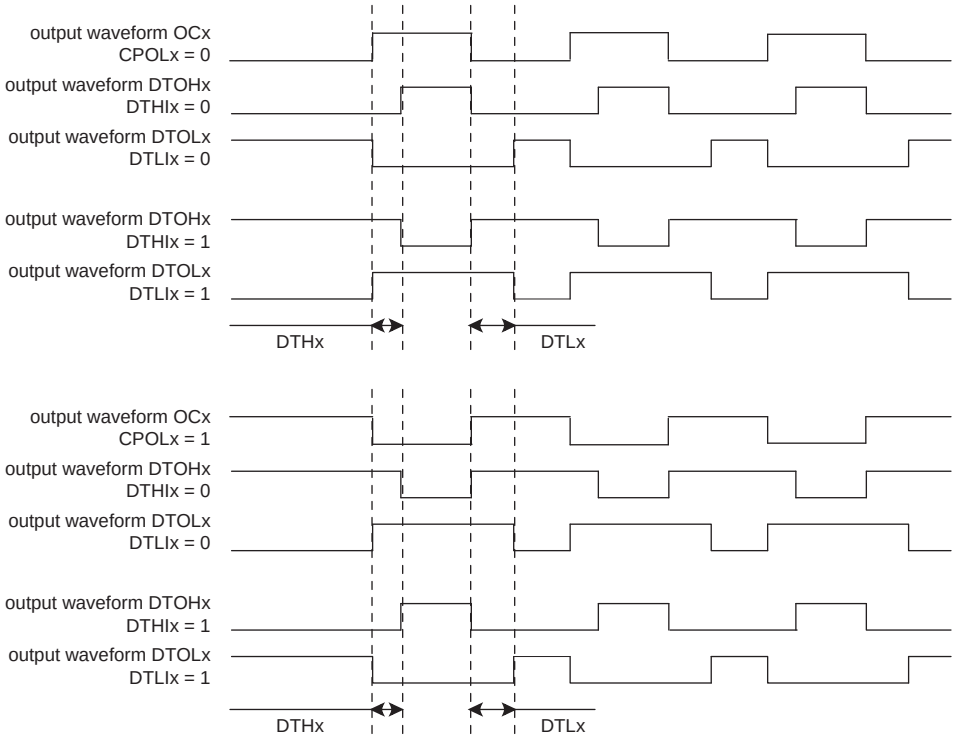
33.6.2.4 Dead-Time Generator

The dead-time generator uses the comparator output OCx to provide the two complementary outputs DTOHx and DTOLx, which allows the PWM to drive external power control switches safely. When the dead-time generator is enabled by writing a one to the DTE bit in the "Channel Mode Register" on page 1037 (CMRx), dead-times (also called dead-bands or non-overlapping times) are inserted between the edges of the two complementary outputs DTOHx and DTOLx. Note that enabling or disabling the dead-time generator is allowed only if the channel is disabled.

The dead-time is adjustable by the "Channel Dead Time Register" on page 1046 (DTx), both outputs of the dead-time generator can be adjusted separately by DTH and DTL. The dead-time values can be updated synchronously to the PWM period by using the "Channel Dead Time Update Register" on page 1047 (DTUPDx).

The dead-time is based on a specific counter which uses the same selected clock that feeds the channel counter of the comparator. Depending on the edge and the configuration of the dead-time, DTOHx and DTOLx are delayed until the counter has reached the value defined by DTH or DTL. An inverted configuration bit (DTHI and DTLI bits in the CMRx register) is provided for each outputs to invert the dead-times outputs. The following figure shows the waveform of the dead-time generator.

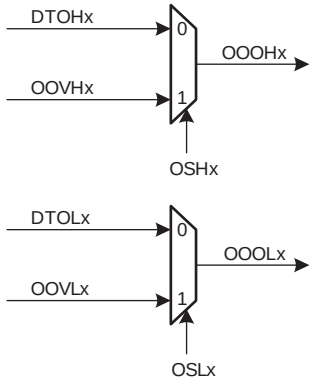
Figure 33-7. Complementary Output Waveforms



33.6.2.5 *Output Override*

The two complementary outputs DTOHx and DTOLx of the dead-time generator can be forced to a value defined by the software.

Figure 33-8. Override Output Selection



The OSHx and OSLx fields in the "Output Selection Register" on page 1017 (OS) allow the outputs of the dead-time generator DTOHx and DTOLx to be overridden by the value defined in the OOVHx and OOVLx fields in the"Output Override Value Register" on page 1016 (OOV).

The set registers "Output Selection Set Register" on page 1018 and "Output Selection Set Update Register" on page 1020 (OSS and OSSUPD) enable the override of the outputs of a channel regardless of other channels. In the same way, the clear registers "Output Selection Clear Register" on page 1019 and "Output Selection Clear Update Register" on page 1021 (OSC and OSCUPD) disable the override of the outputs of a channel regardless of other channels.

By using buffer OSSUPD and OSCUPD registers, the output selection of PWM outputs is done synchronously to the channel counter, at the beginning of the next PWM period.

By using OSS and OSC registers, the output selection of PWM outputs is done asynchronously to the channel counter, as soon as the register is written.

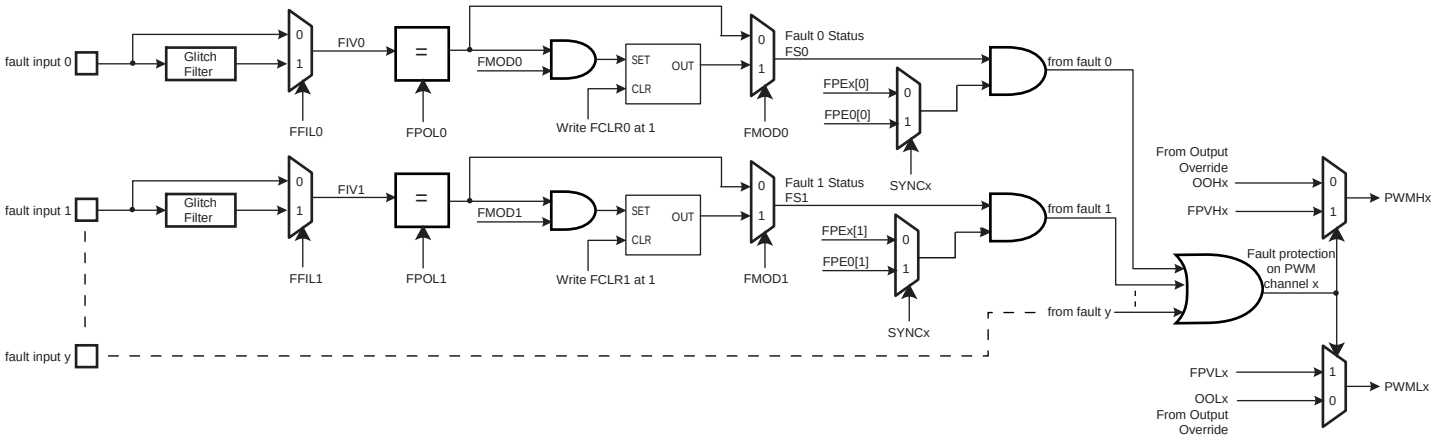
The value of the current output selection can be read in OS.

While overriding PWM outputs, the channel counters continue to run, only the PWM outputs are forced to user defined values.

33.6.2.6 *Fault Protection*

5 inputs provide fault protection which can force any of the PWM output pair to a programmable value. This mechanism has priority over output overriding.

Figure 33-9. Fault Protection



The polarity level of the faults inputs are configured by the FPOL field in the "Fault Mode Register" on page 1022 (FMR).

The fault inputs can be glitch filtered or not in function of the FFIL field in the FMR register. When the filter is enabled, glitches on fault inputs with a width inferior to the PWM internal clock (CCK) period are rejected.

A fault becomes active as soon as its corresponding fault input has a transition to the programmed polarity level. If the corresponding FMOD bit is written to zero in the FMR register, the fault remains active as long as the fault input is at this polarity level. If the corresponding bit FMOD is written to one, the fault remains active until the fault input is not at this polarity level anymore AND until it is cleared by writing the corresponding FCLR bit in the "Fault Clear Register" on page 1024 (FSCR). By reading the "Fault Status ReSister" on page 1023 (FSR), the user can read the current level of the fault inputs thanks to the FIV field, and can know which fault is currently active thanks to the FS field.

Each fault can be taken into account or not by the fault protection mechanism in each channel. To be taken into account in the channel x, the fault y must be enabled by the FPEx[y] bit in the "PWM Fault Protection Enable Registers" (FPE1). However the synchronous channels (see Section 33.6.2.7 on page 980) don't use their own fault enable bits, but those of the channel 0 (FPE0[y] bits).

The fault protection on a channel is triggered when this channel is enabled AND when any one of the faults that are enabled for this channel is active. It can be triggered even if the PWM internal clock (CCK) is not running but only by a fault input that is not glitch filtered.

When the fault protection is triggered on a channel, the fault protection mechanism forces the channel outputs to the values defined by the FPVHx and FPVLx fields in the "Fault Protection Value Register" on page 1025 (FPV) and leads to a reset of the counter of this channel. The output forcing is made asynchronously to the channel counter.

CAUTION:

- To prevent an unexpected activation of the status FSy bit in the FSR register, the FMODy bit can be written to one only if the FPOLy bit has been previously configured to its final value.
- To prevent an unexpected activation of the Fault Protection on the channel x, the FPEx[y] bit can be written to one only if the FPOLy bit has been previously configured to its final value.

If a comparison unit is enabled (see Section 33.6.3 on page 988) and if a fault is triggered in the channel 0, in this case the comparison cannot match.

As soon as the fault protection is triggered on a channel, an interrupt (different from the interrupt generated at the end of the PWM period) can be generated but only if it is enabled and not masked. The interrupt is reset by reading the interrupt status register, even if the fault which has caused the trigger of the fault protection is kept active.

33.6.2.7 Synchronous Channels

Some channels can be linked together as synchronous channels. They have the same source clock, the same period, the same alignment and are started together. In this way, their counters are synchronized together.

The synchronous channels are defined by the SYNCx bits in the "Sync Channels Mode Register" on page 1007 (SCM). Only one group of synchronous channels is allowed.

When a channel is defined as a synchronous channel, the channel 0 is automatically defined as a synchronous channel too, because the channel 0 counter configuration is used by all the synchronous channels.

If a channel x is defined as a synchronous channel, it uses the following configuration fields of the channel 0 instead of its own:

- CPRE0 field in CMR0 register instead of CPREx field in CMRx register (same source clock)
- CPRD0 field in CMR0 register instead of CPRDx field in CMRx register (same period)
- CALG0 field in CMR0 register instead of CALGx field in CMRx register (same alignment)

Thus writing these fields of a synchronous channel has no effect on the output waveform of this channel (except channel 0 of course).

Because counters of synchronous channels must start at the same time, they are all enabled together by enabling the channel 0 (by the CHID0 bit in the ENA register). In the same way, they are all disabled together by disabling the channel 0 (by the CHID0 bit in the DIS register). However, a synchronous channel x different from channel 0 can be enabled or disabled independently from others (by the CHIDx bit in the ENA and DIS registers).

Defining a channel as a synchronous channel while it is an asynchronous channel (by writing the SYNCx bit to one while it was at zero) is allowed only if the channel is disabled at this time (CHIDx=0 in SR register). In the same way, defining a channel as an asynchronous channel while it is a synchronous channel (by writing the SYNCx bit to zero while it was at one) is allowed only if the channel is disabled at this time.

The UPDM (Update Mode) field in the SCM register allow to select one of the three methods to update the registers of the synchronous channels:

- Method 1 (UPDM=0): the period value, the duty-cycle values and the dead-time values must be written by the CPU in their respective update registers (respectively CPRDUPDx, CDTYUPDx and DTUPDx).The update is triggered at the next PWM period as soon as the UPDULOCK bit in the "Sync Channels Update Control Register" on page 1009 (SCUC) is set to 1 (see Section 33.6.2.8 on page 982).
- Method 2 (UPDM=1): the period value, the duty-cycle values, the dead-time values and the update period value must be written by the CPU in their respective update registers (respectively CPRDUPDx, CDTYUPDx and DTUPD). The update of the period value and of the dead-time values is triggered at the next PWM period as soon as the UPDULOCK bit in the "Sync Channels Update Control Register" on page 1009 (SCUC) is set to 1. The update of the duty-cycle values and the update period value is triggered automatically after an update period defined by the UPR field in the "Sync Channels Update Period Register" on page 1010 (SCUP) (see Section 33.6.2.9 on page 983).
- Method 3 (UPDM=2): same as Method 2 apart from the fact that the duty-cycle values of ALL synchronous channels are written by the Peripheral DMA Controller (PDCA) (see Section

33.6.2.10 on page 985). The user can choose to synchronize the PDCA transfer request with a comparison match (see Section 33.6.3 on page 988), by the PTRM and PTRCS fields in the SCM register.

Table 33-2. Summary of the update of registers of Synchronous Channels

	UPDM=0	UPDM=1	UPDM=2
Period Value (CPRDUPDx)	Write by the CPU		
	Update is triggered at the next PWM period as soon as the UPDULOCK bit is set to 1		
Dead-Time Values (DTUPDx)	Write by the CPU		
	Update is triggered at the next PWM period as soon as the UPDULOCK bit is set to 1		
Duty-Cycle Values (CDTYUPDx)	Write by the CPU	Write by the CPU	Write by the PDCA
	Update is triggered at the next PWM period as soon as the UPDULOCK bit is set to 1	Update is triggered at the next PWM period as soon as the update period counter has reached the value UPR	
Update Period Value (SCUPUPD)	Not applicable	Write by the CPU	
	Not applicable	Update is triggered at the next PWM period as soon as the update period counter has reached the value UPR	

33.6.2.8 *Method 1: Manual write of duty-cycle values and manual trigger of the update*

In this mode, the update of the period value, the duty-cycle values and the dead-time values must be made by writing in their respective update registers with the CPU (respectively CPRDUPDx, CDTYUPDx and DTUPDx).

To trigger the update, the user must use the UPDULOCK bit of the "Sync Channels Update Control Register" on page 1009 (SCUC) which allows to update synchronously (at the same PWM period) the synchronous channels:

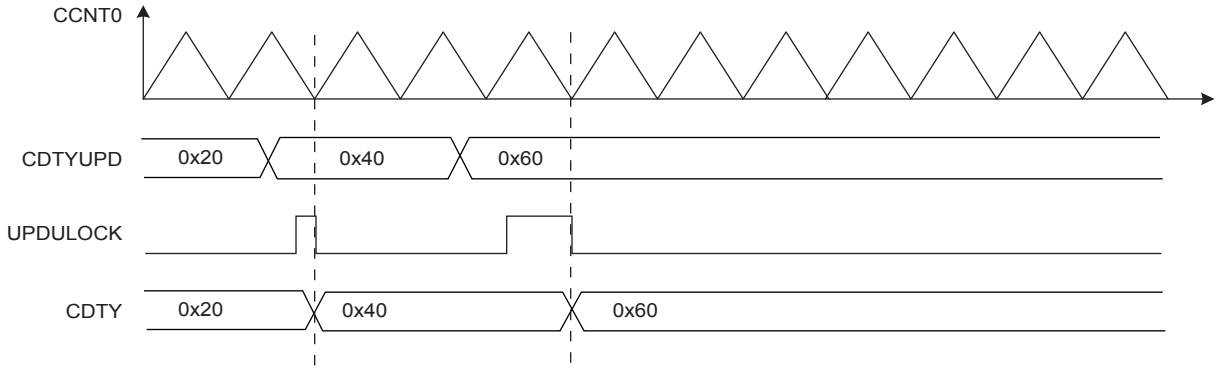
- If the UPDULOCK bit is set to 1, the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to 1, the update is locked and cannot be performed.

After writing the UPDULOCK bit to 1, it is held at this value until the update occurs, then it is read 0.

Sequence for the Method 1:

1. Select the manual write of duty-cycle values and the manual update by writing the UPDM field to zero in the SCM register
2. Define the synchronous channels by the SYNCx bits in the SCM register.
3. Enable the synchronous channels by writing CHID0 in the ENA register.
4. If an update of the period value and/or the duty-cycle values and/or the dead-time values is required, write registers that need to be updated (CPRDUPDx, CDTYUPDx and DTUPDx).
5. Write UPDULOCK to one in SCUC.
6. The update of the registers will occur at the beginning of the next PWM period. At this time the UPDULOCK bit is reset, go to step 4) for new values.

Figure 33-10. Method 1 (UPDM=0)



33.6.2.9 *Method 2: Manual write of duty-cycle values and automatic trigger of the update*

In this mode, the update of the period value, the duty-cycle values, the dead-time values and the update period value must be made by writing in their respective update registers with the CPU (respectively CPRDUPDx, CDTYUPDx, DTUPDx and SCUPUPD).

To trigger the update of the period value and the dead-time values, the user must use the UPDULOCK bit of the "Sync Channels Update Control Register" on page 1009 (SCUC) which allows to update synchronously (at the same PWM period) the synchronous channels:

- If the UPDULOCK bit is set to 1, the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to 1, the update is locked and cannot be performed.

After writing the UPDULOCK bit to one, it is held at this value until the update occurs, then it is read 0.

The update of the duty-cycle values and the update period is triggered automatically after an update period.

To configure the automatic update, the user must define a value for the Update Period by the UPR field in the "Sync Channels Update Period Register" on page 1010 (SCUP). The PWM controller waits UPR+1 periods of synchronous channels before updating automatically the duty values and the update period value.

The status of the duty-cycle value write is reported in the "Interrupt Status Register 2" on page 1015 (ISR2) by the following bits:

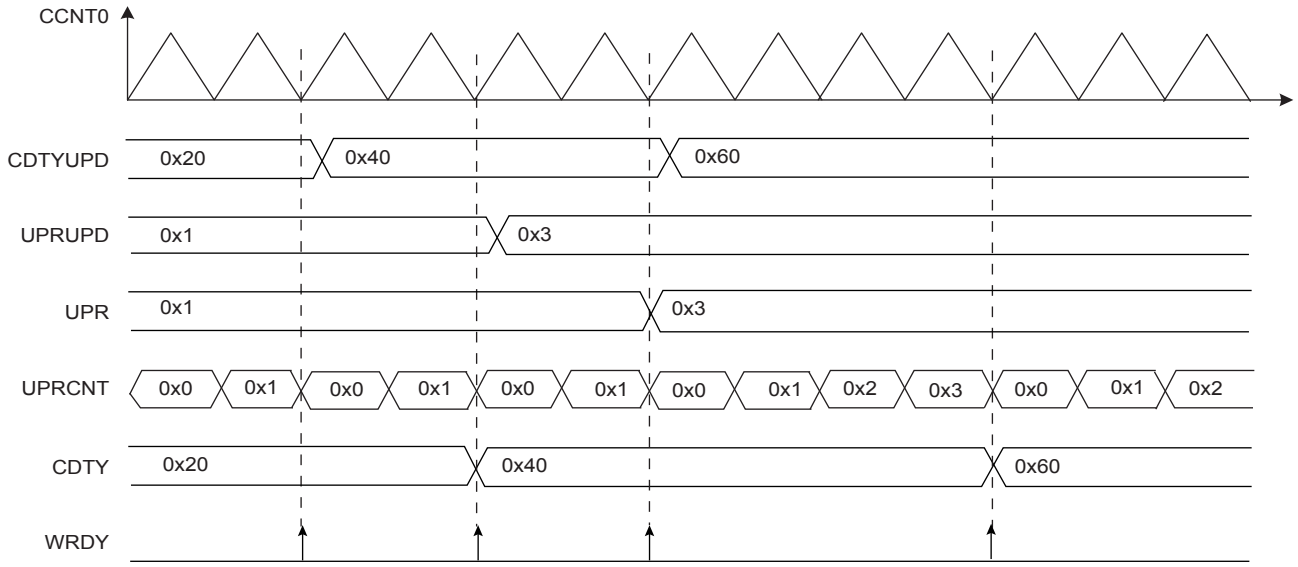
- WRDY: this bit is set to 1 when the PWM Controller is ready to receive new duty-cycle values and a new update period value. It is reset to 0 when the ISR2 register is read.

Depending on the interrupt mask in the IMR2 register, an interrupt can be generated by these bits.

Sequence for the Method 2:

1. Select the manual write of duty-cycle values and the automatic update by writing the UPDM field to one in the SCM register
2. Define the synchronous channels by the SYNCx bits in the SCM register.
3. Define the update period by the UPR field in the SCUP register.
4. Enable the synchronous channels by writing CHID0 in the ENA register.
5. If an update of the period value and/or of the dead-time values is required, write registers that need to be updated (CPRDUPDx, DTUPDx), else go to Step 8.
6. Write UPDULOCK to one in SCUC.
7. The update of these registers will occur at the beginning of the next PWM period. At this moment the UPDULOCK bit is reset, go to Step 5. for new values.
8. If an update of the duty-cycle values and/or the update period is required, check first that write of new update values is possible by polling the WRDY bit (or by waiting for the corresponding interrupt) in the ISR2 register.
9. Write registers that need to be updated (CDTYUPDx, SCUPUPD).
10. The update of these registers will occur at the next PWM period of the synchronous channels when the Update Period is elapsed. Go to Step 8. for new values.

Figure 33-11. Method 2 (UPDM=1)



33.6.2.10 *Method 3: Automatic write of duty-cycle values and automatic trigger of the update*

In this mode, the update of the duty cycle values is made automatically by the Peripheral DMA Controller (PDCA). The update of the period value, the dead-time values and the update period value must be made by writing in their respective update registers with the CPU (respectively CPRDUPDx, DTUPDx and SCUPUPD).

To trigger the update of the period value and the dead-time values, the user must use the UPDULOCK bit which allows to update synchronously (at the same PWM period) the synchronous channels:

- If the UPDULOCK bit is set to 1, the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to 1, the update is locked and cannot be performed.

After writing the UPDULOCK bit to one, it is held at this value until the update occurs, then it is read 0.

The update of the duty-cycle values and the update period value is triggered automatically after an update period.

To configure the automatic update, the user must define a value for the Update Period by the UPR field in the "Sync Channels Update Period Register" on page 1010 (SCUP). The PWM controller waits UPR+1 periods of synchronous channels before updating automatically the duty values and the update period value.

Using the PDCA removes processor overhead by reducing its intervention during the transfer. This significantly reduces the number of clock cycles required for a data transfer, which improves micro controller performance.

The PDCA must write the duty-cycle values in the synchronous channels index order. For example if the channels 0, 1 and 3 are synchronous channels, the PDCA must write the duty-cycle of the channel 0 first, then the duty-cycle of the channel 1, and finally the duty-cycle of the channel 3.

The following status are reported in the "Interrupt Status Register 2" on page 1015 (ISR2):

- WRDY: this bit is set to 1 when the PWM Controller is ready to receive new duty-cycle values and a new update period value. It is reset to 0 when the ISR2 register is read. The user can choose to synchronize the WRDY bit and the PDCA transfer request with a comparison match (see Section 33.6.3 on page 988), by the PTRM and PTRCS fields in the SCM register.
- UNRE: this bit is set to 1 when the update period defined by the UPR field is elapsed while the whole data has not been written by the PDCA. It is reset to 0 when the ISR2 register is read.

Depending on the interrupt mask in the IMR2 register, an interrupt can be generated by these bits.

Sequence for Method 3:

1. Select the automatic write of duty-cycle values and automatically update by setting the UPDM field to 2 in the SCM register.
2. Define the synchronous channels by the SYNCx bits in the SCM register.
3. Define the update period by the UPR field in the SCUP register.
4. Define when the WRDY bit and the corresponding PDCA transfer request must be set in the update period by the PTRM bit and the PTRCS field in the SCM register (at the end of the update period or when a comparison matches).
5. Define the PDCA transfer settings for the duty-cycle values and enable it in the PDCA registers
6. Enable the synchronous channels by writing CHID0 in the ENA register.
7. If an update of the period value and/or of the dead-time values is required, write registers that need to be updated (CPRDUPDx, DTUPDx), else go to Step 10.
8. Write UPDULOCK to one in SCUC.
9. The update of these registers will occur at the beginning of the next PWM period. At this moment the UPDULOCK bit is reset, go to Step 7. for new values.
10. If an update of the update period value is required, check first that write of a new update value is possible by polling the WRDY bit (or by waiting for the corresponding interrupt) in the ISR2 register, else go to Step 13.
11. Write register that need to be updated (SCUPUPD).
12. The update of this registers will occur at the next PWM period of the synchronous channels when the Update Period is elapsed. Go to Step 10. for new values.
13. Check the end of the PDCA transfer with the Transfer Complete bit in the PDCA status register. If the transfer is ended define a new PDCA transfer in the PDCA registers, for new duty-cycle values. Go to Step 5.

Figure 33-12. Method 3 (UPDM=2 and PTRM=0)

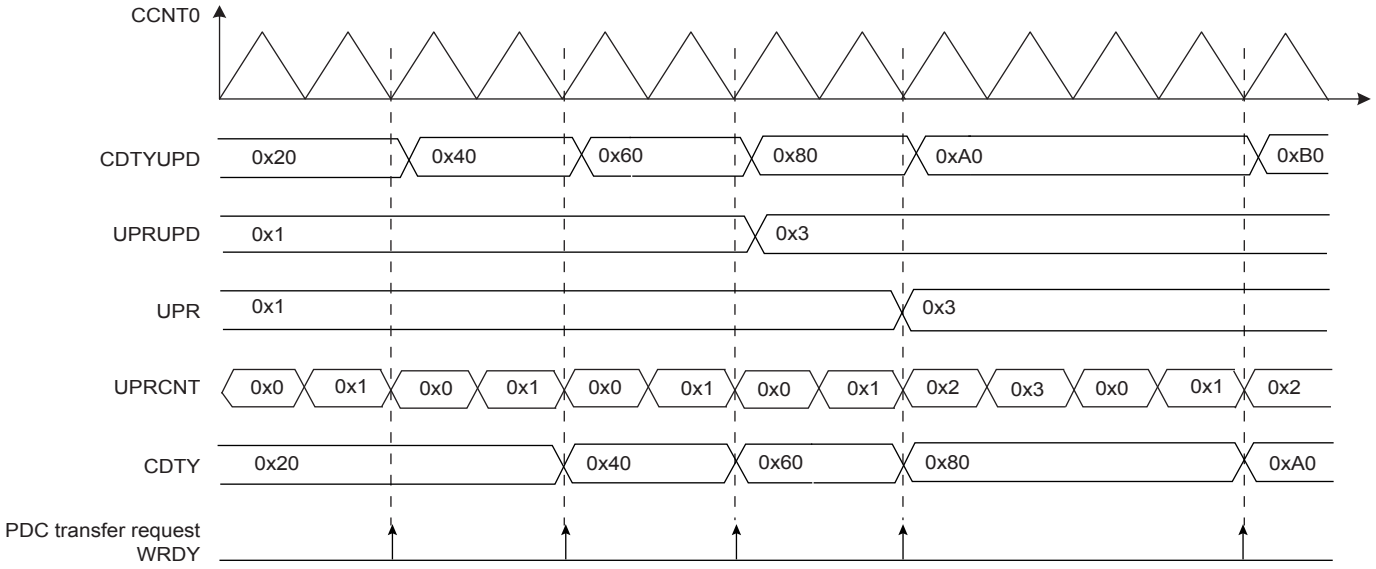
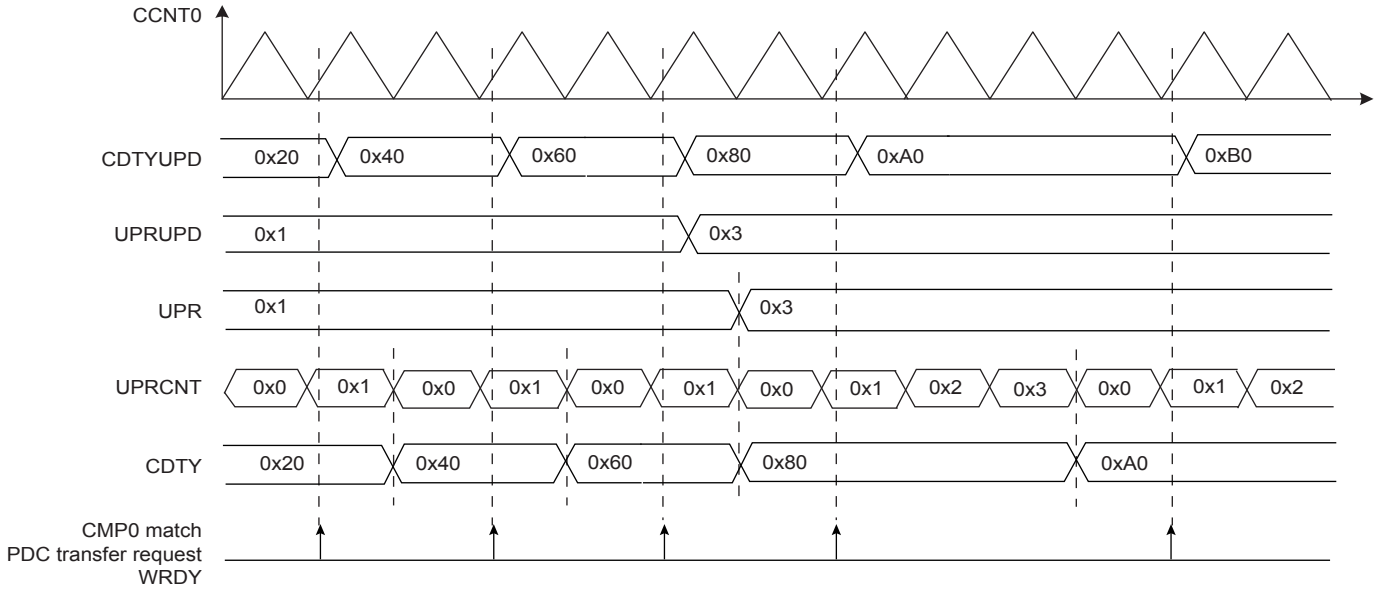


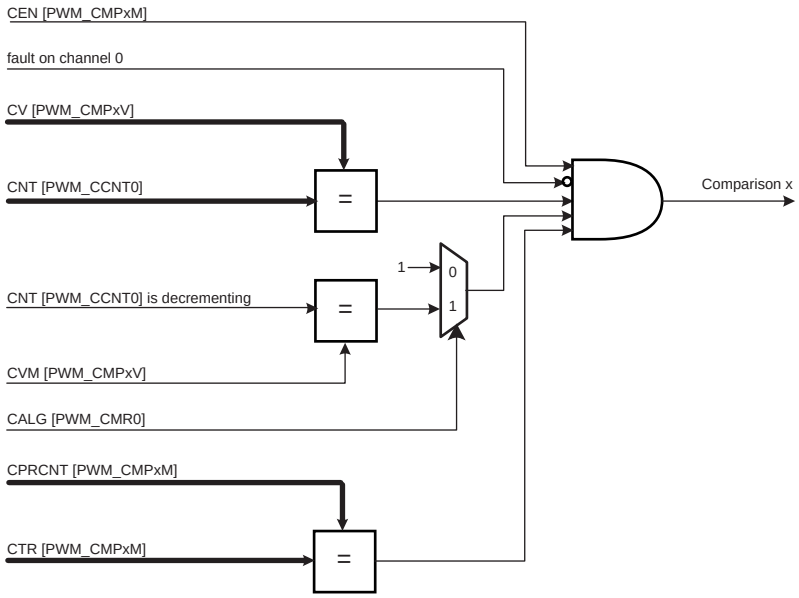
Figure 33-13. Method 3 (UPDM=2 and PTRM=1 and PTRCS=0)



33.6.3 PWM Comparison Units

The PWM provides 8 independent comparison units able to compare a programmed value with the current value of the channel 0 counter (which is the channel counter of all synchronous channels, Section 33.6.2.7 on page 980). These comparisons are intended to generate pulses on the event lines (used to synchronize ADC, see Section 33.6.4 on page 990), to generate software interrupts and to trigger PDCA transfer requests for the synchronous channels (see Section 33.6.2.10 on page 985).

Figure 33-14. Comparison Unit Block Diagram



The comparison x matches when it is enabled by the CEN bit in the "Comparison x Mode Register" on page 1035 (CMPxM for the comparison x) and when the counter of the channel 0 reaches the comparison value defined by the CV field in "Comparison x Value Register" on page 1033 (CMPxV for the comparison x). If the counter of the channel 0 is center aligned (CALG=1 in "Channel Mode Register" on page 1037), the CVM bit (in CMPxV) defines if the comparison is made when the counter is counting up or counting down (in left alignment mode CALG=0, this bit is useless).

If a fault is active on the channel 0, the comparison is disabled and cannot match (see Section 33.6.2.6 on page 978).

The user can define the periodicity of the comparison x by the CTR and CPR fields (in CMPxV). The comparison is performed periodically once every CPR+1 periods of the counter of the channel 0, when the value of the comparison period counter CPRCNT (in CMPxM) reaches the value defined by CTR. CPR is the maximum value of the comparison period counter CPRCNT. If CPR=CTR=0, the comparison is performed at each period of the counter of the channel 0.

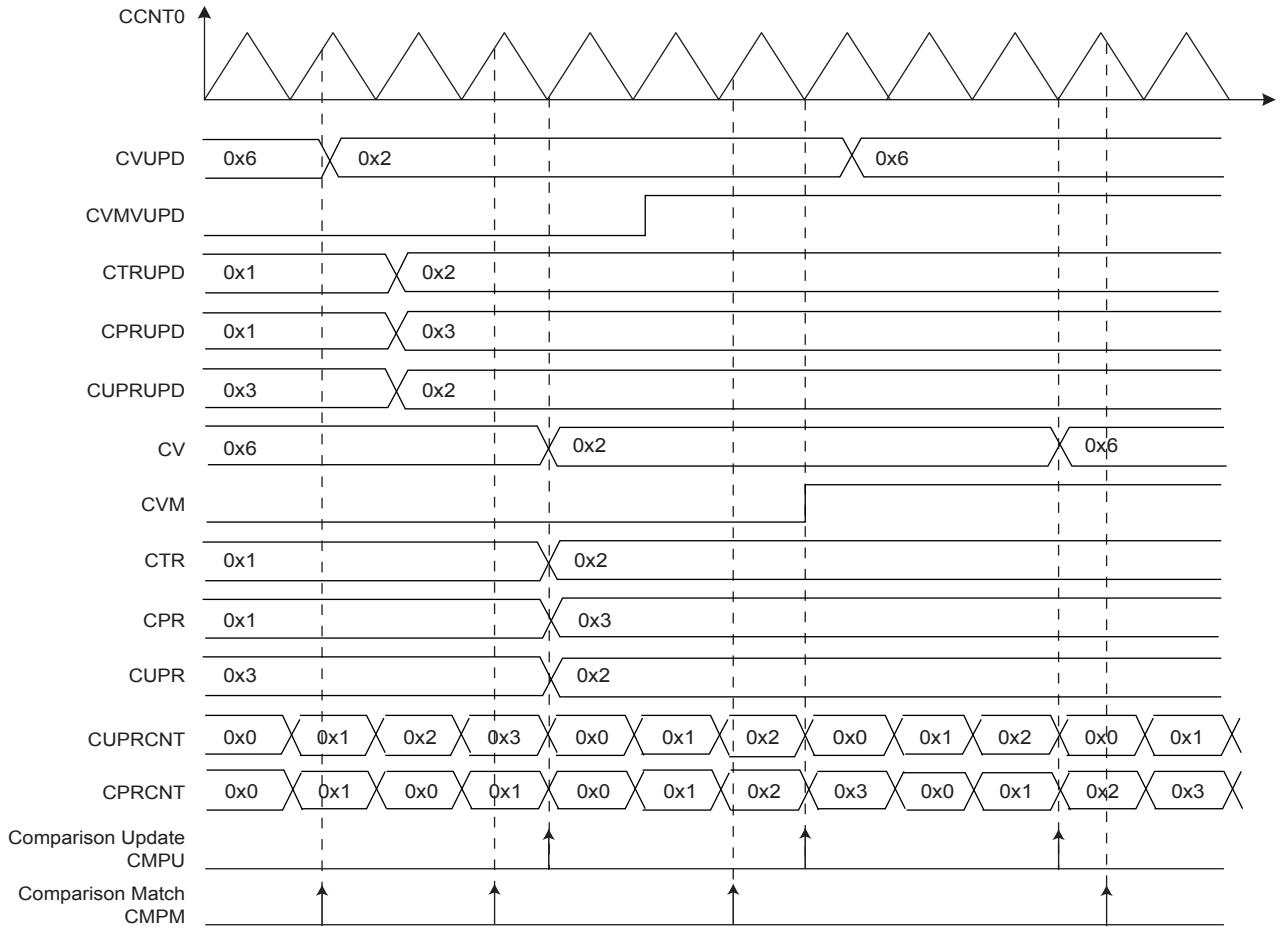
The comparison x configuration can be modified while the channel 0 is enabled by using the "PWM Comparison x Mode Update Register" on page 1036 (CMPxMUPD registers for the comparison x). In the same way, the comparison x value can be modified while the channel 0 is enabled by using the "Comparison x Value Update Register" on page 1034 (CMPxVUPD registers for the comparison x).

The update of the comparison x configuration and the comparison x value is triggered periodically after the comparison x update period. It is defined by the CUPR field in the CMPxM. The comparison unit has an update period counter independent from the period counter to trigger this update. When the value of the comparison update period counter CUPRCNT (in CMPxM) reaches the value defined by CUPR, the update is triggered. The comparison x update period CUPR itself can be updated while the channel 0 is enabled by using the CMPxMUPD register.

CAUTION: to be taken into account, writing in the CMPxVUPD register must be followed by a write in the CMPxMUPD register.

The comparison match and the comparison update can be a source of an interrupt, but only if it is enabled and not masked. These interrupts can be enabled by the "Interrupt Enable Register 2" on page 1012 and disabled by the "Interrupt Disable Register 2" on page 1013. The comparison match interrupt and the comparison update interrupt are reset by reading the "Interrupt Status Register 2" on page 1015.

Figure 33-15. Comparison Waveform

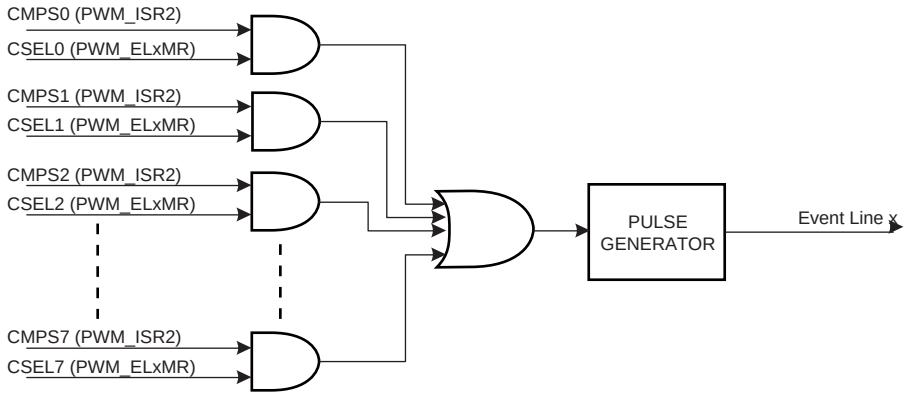


33.6.4 PWM Event Lines

The PWM provides 2 independent event lines intended to trigger actions in other peripherals (in particular for ADC (Analog to Digital Converter)).

A pulse (one cycle of the master clock (CLK_PWM))is generated on an event line, when at least one of the selected comparisons is matching. The comparisons can be selected independently by the CSEL bits in the "Event Line x Register" on page 1027 (ELxMR for the Event Line x).

Figure 33-16. Event Line Block Diagram



33.6.5 PWM Controller Operations

33.6.5.1 Initialization

Before enabling the channels, they must have been configured by the software application:

- Unlock user interface by writing the WPCMD field in the WPCR Register.
- Configuration of the clock generator (DIVA, PREA, DIVB, PREB, CLKSEL in the CLK register if required). After writing CLKSEL to a new value, no write in any PWM registers must be attempted before a delay of 2 master clock periods (CLK_PWM). This is the time needed by the PWM to switch the internal clock (CCK).
- Selection of the clock for each channel (CPRE field in the CMRx register)
- Configuration of the waveform alignment for each channel (CALG field in the CMRx register)
- Selection of the counter event selection (if CALG=1) for each channel (CES field in the CMRx register)
- Configuration of the output waveform polarity for each channel (CPOL in the CMRx register)
- Configuration of the period for each channel (CPRD in the CPRDx register). Writing in CPRDx register is possible while the channel is disabled. After validation of the channel, the user must use CPRDUPDx register to update CPRDx as explained below.
- Configuration of the duty-cycle for each channel (CDTY in the CDTYx register). Writing in CDTYx register is possible while the channel is disabled. After validation of the channel, the user must use CDTYUPDx register to update CDTYx as explained below.
- Configuration of the dead-time generator for each channel (DTH and DTL in DTx) if enabled (DTE bit in the CMRx register). Writing in the DTx register is possible while the channel is disabled. After validation of the channel, the user must use DTUPDx register to update DTx
- Selection of the synchronous channels (SYNCx in the SCM register)
- Selection of the moment when the WRDY bit and the corresponding PDCA transfer request are set (PTRM and PTRCS in the SCM register)
- Configuration of the update mode (UPDM in the SCM register)
- Configuration of the update period (UPR in the SCUP register) if needed.
- Configuration of the comparisons (CMPxV and CMPxM).
- Configuration of the event lines (ELxMR).
- Configuration of the fault inputs polarity (FPOL in FMR)
- Configuration of the fault protection (FMOD and FFIL in FMR, FPV and FPE1)
- Enable of the interrupts (writing CHIDx and FCHIDx in IER1 register, and writing WRDYE, UNRE, CMPMx and CMPUx in IER2 register)
- Enable of the channels (writing CHIDx in the ENA register)

33.6.5.2 Source Clock Selection Criteria

The large number of source clocks can make selection difficult. The relationship between the value in the "Channel Period Register" on page 1041 (CPRDx) and the "Channel Duty Cycle Register" on page 1039 (CDTYx) can help the user. The event number written in the Period Register gives the PWM accuracy. The Duty-Cycle quantum cannot be lower than *1/CPRDx* value. The higher the value of CPRDx, the greater the PWM accuracy.

For example, if the user writes 15 (in decimal) in CPRDx, the user is able to write a value between 1 up to 14 in CDTYx Register. The resulting duty-cycle quantum cannot be lower than 1/15 of the PWM period.

33.6.5.3 Changing the Duty-Cycle, the Period and the Dead-Times

It is possible to modulate the output waveform duty-cycle, period and dead-times.

To prevent unexpected output waveform, the user must use the "Channel Duty Cycle Update Register" on page 1040, the "Channel Period Update Register" on page 1043 and the "Channel Dead Time Update Register" on page 1047 (CDTYUPDx, CPRDUPDx and DTUPDx) to change waveform parameters while the channel is still enabled.

- If the channel is an asynchronous channel (SYNCx=0 in "Sync Channels Mode Register" on page 1007 (SCM)), these registers hold the new period, duty-cycle and dead-times values until the end of the current PWM period and update the values for the next period.
- If the channel is a synchronous channel and update method 0 is selected (SYNCx=1 and UPDM=0 in SCM register), these registers hold the new period, duty-cycle and dead-times values until the UPDULOCK bit is written to one (in "Sync Channels Update Control Register" on page 1009 (SCUC)) and the end of the current PWM period, then update the values for the next period.
- If the channel is a synchronous channel and update method 1 or 2 is selected (SYNCx=1 and UPDM=1 or 2 in SCM register):
  - these CPRDUPDx and DTUPDx registers hold the new period and dead-times values until the UPDULOCK bit is written to one (in SCUC register) and the end of the current PWM period, then update the values for the next period.
  - the CDTYUPDx register holds the new duty-cycle value until the end of the update period of synchronous channels (when UPRCNT is equal to UPR in "Sync Channels Update Period Register" on page 1010 (SCUP)) and the end of the current PWM period, then updates the value for the next period

Note: If the update registers (CDTYUPDx, CPRDUPDx and DTUPDx) are written several times between two updates, only the last written value is taken into account.

33.6.5.4 Changing the Synchronous Channels Update Period

It is possible to change the update period of synchronous channels (see Section 33.6.2.9 on page 983 and Section 33.6.2.10 on page 985) while they are enabled.

To prevent an unexpected update of the synchronous channels registers, the user must use the "Sync Channels Update Period Update Register" on page 1011 (SCUPUPD) to change the update period of synchronous channels while they are still enabled. This register holds the new value until the end of the update period of synchronous channels (when UPRCNT is equal to

UPR in "Sync Channels Update Period Register" on page 1010 (SCUP)) and the end of the current PWM period, then updates the value for the next period.

Note: If the SCUPUPD update register is written several times between two updates, only the last written value is taken into account.

Note: Changing the update period does make sense only if there is one or more synchronous channels and if the update method 1 or 2 is selected (UPDM=1 or 2 in "Sync Channels Mode Register" on page 1007).

33.6.5.5 *Changing the Comparison Value and the Comparison Configuration*

It is possible to change the comparison values and the comparison configurations while the channel 0 is enabled (see Section 33.6.3 on page 988).

To prevent unexpected comparison match, the user must use the "Comparison x Value Update Register" on page 1034 and the "PWM Comparison x Mode Update Register" on page 1036 (CMPxVUPD and CMPxMUPD) to change respectively the comparison values and the comparison configurations while the channel 0 is still enabled. These registers hold the new values until the end of the comparison update period (when CUPRCNT is equal to CUPR in "Comparison x Mode Register" on page 1035 (CMPxM)) and the end of the current PWM period, then update the values for the next period.

CAUTION: to be taken into account, the write of the CMPxVUPD register must be followed by a write of the CMPxMUPD register.

Note: If the update registers CMPxVUPD and CMPxMUPD are written several times between two updates, only the last written value are taken into account.

33.6.5.6 *Interrupts*

Depending on the interrupt mask in the IMR1 and IMR2 registers, an interrupt can be generated at the end of the corresponding channel period (CHIDx in the ISR1 register), after a fault event (FCHIDx in the ISR1 register), after a comparison match (CMPMx in the ISR2 register), after a comparison update (CMPUx in the ISR2 register) or according to the transfer mode of the synchronous channels (WRDY and UNRE in the ISR2 register).

If the interrupt is generated by the CHIDx or FCHIDx bits, the interrupt remains active until a read operation in the ISR1 register occurs.

If the interrupt is generated by the WRDY, UNRE, CMPMx or CMPUx bits, the interrupt remains active until a read operation in the ISR2 register occurs.

A channel interrupt is enabled by setting the corresponding bit in the IER1 and IER2 registers. A channel interrupt is disabled by setting the corresponding bit in the IDR1 and IDR2 registers.

33.6.5.7 Write Protect Registers

To prevent any single software error that may corrupt PWM behavior, the registers listed below can be write-protected by writing the WPCMD field in the "Write Protect Control Register" on page 1029 (WPCR). They are divided into 6 groups:

- Register group 0:
  - "Clock Register" on page 998
- Register group 1:
  - "Disable Register" on page 1001
- Register group 2:
  - "Sync Channels Mode Register" on page 1007
  - "Channel Mode Register" on page 1037
  - "Stepper Motor Mode Register" on page 1028
- Register group 3:
  - "Channel Period Register" on page 1041
  - "Channel Period Update Register" on page 1043
- Register group 4:
  - "Channel Dead Time Register" on page 1046
  - "Channel Dead Time Update Register" on page 1047
- Register group 5:
  - "Fault Mode Register" on page 1022
  - "Fault Protection Value Register" on page 1025

There are two types of Write Protect:

- the Write Protect SW, which can be enabled or disabled.
- the Write Protect HW, which can just be enabled, only a hardware reset of the PWM controller can disable it.

Both Write Protect can be applied independently to a particular register group thanks to the WPCMD and WPRG fields in WPCR register. If at least one of the Write Protect is active, the register group is write-protected. The WPCMD field allows to perform the following actions depending on its value:

- 0: Disabling the Write Protect SW of the register groups of which the WPRG bit is at 1.
- 1: Enabling the Write Protect SW of the register groups of which the WPRG bit is at 1.
- 2: Enabling the Write Protect HW of the register groups of which the WPRG bit is at 1.

At any time, the user can know which Write Protect is active in which register group by the WPSWS and WPHWS fields in the "Write Protect Status Register" on page 1031 (WPSR).

If a write access in a write-protected register is detected, then the WPVS bit in the WPSR register is set and the WPVSRC field indicates in which register the write access has been attempted, through its address offset without the two LSBs.

The WPVS and WPSR fields are automatically reset after reading the WPSR register.

33.7 User Interface

Table 33-3. PWM Register Memory Map[2]

Offset	Register	Register Name	Access	Reset
0x000	Clock Register	CLK	Read/write	0x00000000
0x004	Enable Register	ENA	Write-only	–
0x008	Disable Register	DIS	Write-only	–
0x00C	Status Register	SR	Read-only	0x00000000
0x010	Interrupt Enable Register 1	IER1	Write-only	–
0x014	Interrupt Disable Register 1	IDR1	Write-only	–
0x018	Interrupt Mask Register 1	IMR1	Read-only	0x00000000
0x01C	Interrupt Status Register 1	ISR1	Read-only	0x00000000
0x020	Sync Channels Mode Register	SCM	Read/write	0x00000000
0x024	Reserved	–	–	–
0x028	Sync Channels Update Control Register	SCUC	Read/Write	0x00000000
0x02C	Sync Channels Update Period Register	SCUP	Read/Write	0x00000000
0x030	Sync Channels Update Period Update Register	SCUPUPD	Write-only	0x00000000
0x034	Interrupt Enable Register 2	IER2	Write-only	–
0x038	Interrupt Disable Register 2	IDR2	Write-only	–
0x03C	Interrupt Mask Register 2	IMR2	Read-only	0x00000000
0x040	Interrupt Status Register 2	ISR2	Read-only	0x00000000
0x044	Output Override Value Register	OOV	Read/Write	0x00000000
0x048	Output Selection Register	OS	Read/Write	0x00000000
0x04C	Output Selection Set Register	OSS	Write-only	–
0x050	Output Selection Clear Register	OSC	Write-only	–
0x054	Output Selection Set Update Register	OSSUPD	Write-only	–
0x058	Output Selection Clear Update Register	OSCUPD	Write-only	–
0x05C	Fault Mode Register	FMR	Read/Write	0x00000000
0x060	Fault Status Register	FSR	Read-only	0x00000000
0x064	Fault Clear Register	FCR	Write-only	–
0x068	Fault Protection Value Register	FPV	Read/Write	0x00000000
0x6C	Fault Protection Enable Register	FPE	Read/Write	0x00000000
0x070-0x078	Reserved	–	–	–
0x07C	Event Line 0 Mode Register	EL0MR	Read/Write	0x00000000
0x080	Event Line 1 Mode Register	EL1MR	Read/Write	0x00000000
0x084 - 0x0E0	Reserved	–	–	–
0x0B0	PWM Stepper Motor Mode Register	SMMR	Read/Write	0x000000000
0x0B4 - 0x0E0	Reserved	–	–	–

Table 33-3. PWM Register Memory Map[2]

Offset	Register	Register Name	Access	Reset
0x0E4	Write Protect Control Register	WPCR	Write-only	–
0x0E8	Write Protect Status Register	WPSR	Read-only	0x000000000
0x0EC - 0x0F8	Reserved	–	–	–
0x0FC	Version Register	VERSION	Read-only	0x- [1]
0x100 - 0x12C	Reserved	–	–	–
0x130	Comparison 0 Value Register	CMP0V	Read/Write	0x000000000
0x134	Comparison 0 Value Update Register	CMP0VUPD	Write-only	–
0x138	Comparison 0 Mode Register	CMP0M	Read/Write	0x000000000
0x13C	Comparison 0 Mode Update Register	CMP0MUPD	Write-only	–
0x140	Comparison 1 Value Register	CMP1V	Read/Write	0x000000000
0x144	Comparison 1 Value Update Register	CMP1VUPD	Write-only	–
0x148	Comparison 1 Mode Register	CMP1M	Read/Write	0x000000000
0x14C	Comparison 1 Mode Update Register	CMP1MUPD	Write-only	–
0x150	Comparison 2 Value Register	CMP2V	Read/Write	0x000000000
0x154	Comparison 2 Value Update Register	CMP2VUPD	Write-only	–
0x158	Comparison 2 Mode Register	CMP2M	Read/Write	0x000000000
0x15C	Comparison 2 Mode Update Register	CMP2MUPD	Write-only	–
0x160	Comparison 3 Value Register	CMP3V	Read/Write	0x000000000
0x164	Comparison 3 Value Update Register	CMP3VUPD	Write-only	–
0x168	Comparison 3 Mode Register	CMP3M	Read/Write	0x000000000
0x16C	Comparison 3 Mode Update Register	CMP3MUPD	Write-only	–
0x170	Comparison 4 Value Register	CMP4V	Read/Write	0x000000000
0x174	Comparison 4 Value Update Register	CMP4VUPD	Write-only	–
0x178	Comparison 4 Mode Register	CMP4M	Read/Write	0x000000000
0x17C	Comparison 4 Mode Update Register	CMP4MUPD	Write-only	–
0x180	Comparison 5 Value Register	CMP5V	Read/Write	0x000000000
0x184	Comparison 5 Value Update Register	CMP5VUPD	Write-only	–
0x188	Comparison 5 Mode Register	CMP5M	Read/Write	0x000000000
0x18C	Comparison 5 Mode Update Register	CMP5MUPD	Write-only	–
0x190	Comparison 6 Value Register	CMP6V	Read/Write	0x000000000
0x194	Comparison 6 Value Update Register	CMP6VUPD	Write-only	–
0x198	Comparison 6 Mode Register	CMP6M	Read/Write	0x000000000
0x19C	Comparison 6 Mode Update Register	CMP6MUPD	Write-only	–
0x1A0	Comparison 7 Value Register	CMP7V	Read/Write	0x000000000
0x1A4	Comparison 7 Value Update Register	CMP7VUPD	Write-only	–

Table 33-3. PWM Register Memory Map[2]

Offset	Register	Register Name	Access	Reset
0x1A8	Comparison 7 Mode Register	CMP7M	Read/Write	0x000000000
0x1AC	Comparison 7 Mode Update Register	CMP7MUPD	Write-only	–
0x1B0 - 0x1FC	Reserved	–	–	–
0x200 + ch_num * 0x20 + 0x00	Channel Mode Register	CMR	Read/Write	0x000000000
0x200 + ch_num * 0x20 + 0x04	Channel Duty Cycle Register	CDTY	Read/Write	0x000000000
0x200 + ch_num * 0x20 + 0x08	Channel Duty Cycle Update Register	CDTYUPD	Write-only	–
0x200 + ch_num * 0x20 + 0x0C	Channel Period Register	CPRD	Read/Write	0x000000000
0x200 + ch_num * 0x20 + 0x10	Channel Period Update Register	CPRDUPD	Write-only	–
0x200 + ch_num * 0x20 + 0x14	Channel Counter Register	CCNT	Read-only	0x000000000
0x200 + ch_num * 0x20 + 0x18	Channel Dead Time Register	DT	Read/Write	0x000000000
0x200 + ch_num * 0x20 + 0x1C	Channel Dead Time Update Register	DTUPD	Write-only	–

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

2. Some registers are indexed with “ch_num” index ranging from 0 to 3.

33.7.1 Clock Register

Name: CLK

Access Type: Read/Write

Offset: 0x000

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CLKSEL	-	-	-	PREB			
23	22	21	20	19	18	17	16
DIVB							
15	14	13	12	11	10	9	8
-	-	-	-	PREA			
7	6	5	4	3	2	1	0
DIVA							

This register can only be written if the WPSWS0 and WPHWS0 bits are cleared in ”Write Protect Status Register” on page 1031.

- **CLKSEL: CCK Source Clock Selection**
  0: The PWM internal clock CCK is driven by the master clock CLK_PWM.
  1: The PWM internal clock CCK is driven by the generic clock GCLK.

CAUTION: After writing CLKSEL to a new value, no write to any PWM registers must be attempted before a delay of 2 master clock periods (CLK_PWM). This is the time needed by the PWM to switch the internal clock CCK.

- **PREA, PREB: CLKA, CLKB Source Clock Selection**

Table 33-4. Source Clock Selection

PREA, PREB	Divider Input Clock
0	CCK
1	CCK/2
2	CCK/4
3	CCK/8
4	CCK/16
5	CCK/32
6	CCK/64
7	CCK/128
8	CCK/256
9	CCK/512
10	CCK/1024
Other	Reserved

- **DIVA, DIVB: CLKA, CLKB Divide Factor**

Table 33-5. Divide Factor

DIVA/DIVB	CLKA/CLKB
0	CLKA/CLKB clock is turned off
1	CLKA/CLKB clock is selected by PREA/PREB
2 - 255	CLKA/CLKB clock is selected by PREA/PREB divided by DIVA/DIVB factor

33.7.2 Enable Register

Name: ENA

Access Type: Write-only

Offset: 0x004

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will enable the PWM output for channel x.

33.7.3 Disable Register

Name: DIS

Access Type: Write-only

Offset: 0x008

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

This register can only be written if the WPSWS1 and WPHWS1 bits are cleared in ”Write Protect Status Register” on page 1031.

- **CHIDx: Channel ID**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will disable the PWM output for channel x.

33.7.4 Status Register

Name: SR

Access Type: Read-only

Offset: 0x00C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**
  0: PWM output for channel x is disabled.
  1: PWM output for channel x is enabled.

33.7.5 Interrupt Enable Register 1

Name: IER1

Access Type: Write-only

Offset: 0x010

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

33.7.6 Interrupt Disable Register 1

Name: IDR1

Access Type: Write-only

Offset: 0x014

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

33.7.7 Interrupt Mask Register 1

Name: IMR1

Access Type: Read-only

Offset: 0x018

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

33.7.8 Interrupt Status Register 1

Name: ISR1

Access Type: Read-only

Offset: 0x01C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **FCHIDx: Fault Protection Trigger on Channel x**
  0: No new trigger of the fault protection since the last read of the ISR1 register.
  1: At least one trigger of the fault protection since the last read of the ISR1 register.
- **CHIDx: Counter Event on Channel x**
  0: No new counter event has occurred since the last read of the ISR1 register.
  1: At least one counter event has occurred since the last read of the ISR1 register.

Note: Reading ISR1 automatically clears CHIDx and FCHIDx.

33.7.9 Sync Channels Mode Register

Name: SCM

Access Type: Read/Write

Offset: 0x020

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
PTRCS			PTRM	-	-	UPDM	
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	SYNC3	SYNC2	SYNC1	SYNC0

This register can only be written if the WPSWS2 and WPHWS2 bits are cleared in ”Write Protect Status Register” on page 1031.

- **PTRCS: PDCA Transfer Request Comparison Selection**
  Selection of the comparison used to set the WRDY bit and the corresponding PDCA transfer request.
- **PTRM: PDCA Transfer Request Mode**

Table 33-6. WRDY bit and PDCA Transfer Request

UPDM	PTRM	WRDY bit and PDCA Transfer Request
0	x	The WRDY bit in ”Interrupt Status Register 2” on page 1015 and the PDCA transfer request are never set to 1.
1	x	The WRDY bit in ”Interrupt Status Register 2” on page 1015 is set to 1 as soon as the update period is elapsed, the PDCA transfer is never requested.
2	0	The WRDY bit in ”Interrupt Status Register 2” on page 1015 and the PDCA transfer is requested as soon as the update period is elapsed.
	1	The WRDY bit in ”Interrupt Status Register 2” on page 1015 and the PDCA transfer is requested as soon as the selected comparison matches.

- **UPDM: Synchronous Channels Update Mode**
  0: Manual write of double buffer registers and manual update of synchronous channels. The update occurs at the beginning of the next PWM period, when the UPDULOCK bit in ”Sync Channels Update Control Register” on page 1009 is set.
  1: Manual write of double buffer registers and automatic update of synchronous channels. The update occurs when the Update Period is elapsed.
  2: Automatic write of duty-cycle update registers by the PDCA and automatic update of synchronous channels. The update occurs when the Update Period is elapsed.
  3: Reserved.

- **SYNCx: Synchronous Channel x**
  0: Channel x is not a synchronous channel.
  1: Channel x is a synchronous channel.

33.7.10 Sync Channels Update Control Register

Name: SCUC

Access Type: Read/Write

Offset: 0x028

Reset Value: 0x00000000

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

- **UPDULOCK: Synchronous Channels Update Unlock**

  0: No effect

  1: If the UPDM field is set to “0” in ”Sync Channels Mode Register” on page 1007, writing the UPDULOCK bit to one will trigger the update of the period value, the duty-cycle and the dead-time values of synchronous channels at the beginning of the next PWM period. If the UPDM field is set to “1” or “2”, writing the UPDULOCK bit to one will trigger only the update of the period value and the dead-time values of synchronous channels.

This bit is automatically reset when the update is done.

33.7.11 Sync Channels Update Period Register

Name: SCUP

Access Type: Read/Write

Offset: 0x02C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
UPRCNT				UPR			

- **UPRCNT: Update Period Counter**
  Reports the value of the Update Period Counter.
- **UPR: Update Period**
  Defines the time between each update of the synchronous channels if automatic trigger of the update is activated (UPDM=1 of UPDM=2 in "Sync Channels Mode Register" on page 1007). This time is equal to UPR+1 periods of the synchronous channels.

33.7.12 Sync Channels Update Period Update Register

Name: SCUPUPD

Access Type: Write-only

Offset: 0x030

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	UPRUPD			

This register acts as a double buffer for the UPR value. This prevents an unexpected automatic trigger of the update of synchronous channels.

- **UPRUPD: Update Period Update**

  Defines the requested time between each update of the synchronous channels if automatic trigger of the update is activated (UPDM=1 of UPDM=2 in "Sync Channels Mode Register" on page 1007). This time is equal to UPR+1 periods of the synchronous channels.

33.7.13 Interrupt Enable Register 2

Name: IER2

Access Type: Write-only

Offset: 0x034

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	6	5	4	3	2	1	0
-	-	-	-	UNRE	-	-	WRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

33.7.14 Interrupt Disable Register 2

Name: IDR2

Access Type: Write-only

Offset: 0x038

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	6	5	4	3	2	1	0
-	-	-	-	UNRE	-	-	WRDY

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

33.7.15 Interrupt Mask Register 2

Name: IMR2

Access Type: Read-only

Offset: 0x03C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	6	5	4	3	2	1	0
-	-	-	-	UNRE	-	-	WRDY

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

33.7.16 Interrupt Status Register 2

Name: ISR2

Access Type: Read-only

Offset: 0x040

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	6	5	4	3	2	1	0
-	-	-	-	UNRE	-	-	WRDY

- **CMPUx: Comparison x Update**
  0: The comparison x has not been updated since the last read of the ISR2 register.
  1: The comparison x has been updated at least one time since the last read of the ISR2 register.
- **CMPMx: Comparison x Match**
  0: The comparison x has not matched since the last read of the ISR2 register.
  1: The comparison x has matched at least one time since the last read of the ISR2 register.
- **UNRE: Synchronous Channels Update Underrun Error**
  0: No Synchronous Channels Update Underrun has occurred since the last read of the ISR2 register.
  1: At least one Synchronous Channels Update Underrun has occurred since the last read of the ISR2 register.
- **WRDY: Write Ready for Synchronous Channels Update**
  0: New duty-cycle and dead-time values for the synchronous channels cannot be written.
  1: New duty-cycle and dead-time values for the synchronous channels can be written.

Note: Reading ISR2 automatically clears WRDY, UNRE and CMPSx.

33.7.17 Output Override Value Register

Name: OOV

Access Type: Read/Write

Offset: 0x044

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OOVL3	OOVL2	OOVL1	OOVL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OOVH3	OOVH2	OOVH1	OOVH0

- **OOVLx: Output Override Value for PWML output of the channel x**
  0: Override value is 0 for PWML output of channel x.
  1: Override value is 1 for PWML output of channel x.
- **OOVHx: Output Override Value for PWMH output of the channel x**
  0: Override value is 0 for PWMH output of channel x.
  1: Override value is 1 for PWMH output of channel x.

33.7.18 Output Selection Register

Name: OS

Access Type: Read/Write

Offset: 0x048

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSL3	OSL2	OSL1	OSL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSH3	OSH2	OSH1	OSH0

- **OSLx: Output Selection for PWML output of the channel x**
  0: Dead-time generator output DTOLx selected as PWML output of channel x.
  1: Output override value OOVLx selected as PWML output of channel x.
- **OSHx: Output Selection for PWMH output of the channel x**
  0: Dead-time generator output DTOHx selected as PWMH output of channel x.
  1: Output override value OOVHx selected as PWMH output of channel x.

33.7.19 Output Selection Set Register

Name: OSS

Access Type: Write-only

Offset: 0x04C

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSSL3	OSSL2	OSSL1	OSSL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSSH3	OSSH2	OSSH1	OSSH0

- **OSSLx: Output Selection Set for PWML output of the channel x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will override the PWML output of channel x with the OOVLx value.
- **OSSHx: Output Selection Set for PWMH output of the channel x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will override the PWMH output of channel x with the OOVHx value.

33.7.20 Output Selection Clear Register

Name: OSC

Access Type: Write-only

Offset: 0x050

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSCL3	OSCL2	OSCL1	OSCL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCH3	OSCH2	OSCH1	OSCH0

- **OSCLx: Output Selection Clear for PWML output of the channel x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will override the PWML output of channel x with the DTOLx value.
- **OSCHx: Output Selection Clear for PWMH output of the channel x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will override the PWMH output of channel x with the DTOHx value.

33.7.21 Output Selection Set Update Register

Name: OSSUPD

Access Type: Write-only

Offset: 0x054

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSSUPL3	OSSUPL2	OSSUPL1	OSSUPL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSSUPH3	OSSUPH2	OSSUPH1	OSSUPH0

- **OSSUPLx: Output Selection Set for PWML output of the channel x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will override the PWML output of channel x with the OOVLx value at the beginning of the next channel x PWM period.
- **OSSUPHx: Output Selection Set for PWMH output of the channel x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will override the PWMH output of channel x with the OOVHx value at the beginning of the next channel x PWM period.

33.7.22 Output Selection Clear Update Register

Name: OSCUPD

Access Type: Write-only

Offset: 0x058

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSCUPL3	OSCUPL2	OSCUPL1	OSCUPL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCUPH3	OSCUPH2	OSCUPH1	OSCUPH0

- **OSCUPLx: Output Selection Clear for PWML output of the channel x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will override the PWML output of channel x with the DTOLx value at the beginning of the next channel x PWM period.
- **OSCUPHx: Output Selection Clear for PWMH output of the channel x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will override the PWMH output of channel x with the DTOHx value at the beginning of the next channel x PWM period.

33.7.23 Fault Mode Register

Name: FMR

Access Type: Read/Write

Offset: 0x05C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
FFIL7	FFIL6	FFIL5	FFIL4	FFIL3	FFIL2	FFIL1	FFIL0
15	14	13	12	11	10	9	8
FMOD7	FMOD6	FMOD5	FMOD4	FMOD3	FMOD2	FMOD1	FMOD0
7	6	5	4	3	2	1	0
FPOL7	FPOL6	FPOL5	FPOL4	FPOL3	FPOL2	FPOL1	FPOL0

This register can only be written if the WPSWS5 and WPHWS5 bits are cleared in ”Write Protect Status Register” on page 1031.

- **FFILy: Fault y Filtering**
  0: The fault input y is not filtered.
  1: The fault input y is filtered.
- **FMODy: Fault y Activation Mode**
  0: The fault y is active as long as the fault input x is at FPOLy.
  1: The fault y becomes active as soon as the fault input y is at FPOLy level. The fault y stays active until the fault input y is not at FPOLy level **AND** until it is cleared in ”Fault Clear Register” on page 1024.
- **FPOLy: Fault y Polarity**
  0: The fault y becomes active when the fault input y is set to 0.
  1: The fault y becomes active when the fault input y is set to 1.

CAUTION: To prevent an unexpected activation of the FSy bit in the ”Fault Status ReSister” on page 1023, the FMODy bit can be set to one only if the FPOLy bit has been previously configured to its final value.

33.7.24 Fault Status ReSister

Name: FMR

Access Type: Read/Write

Offset: 0x060

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
FS7	FS6	FS5	FS4	FS3	FS2	FS1	FS0
7	6	5	4	3	2	1	0
FIV7	FIV6	FIV5	FIV4	FIV3	FIV2	FIV1	FIV0

- **FSy: Fault y Status**
  0: The fault y is not currently active.
  1: The fault y is currently active.
- **FIVy: Fault Input y Value**
  0: The current sampled value of the fault input y is zero (after filtering if enabled).
  1: The current sampled value of the fault input y is one (after filtering if enabled).

33.7.25 Fault Clear Register

Name: FCR

Access Type: Write-only

Offset: 0x064

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
FCLR7	FCLR6	FCLR5	FCLR4	FCLR3	FCLR2	FCLR1	FCLR0

- **FCLRy: Fault y Clear**

  Writing a zero to this bit has no effect.

  If the FMODy bit is set to one and if the fault input y is not at the level defined by the FPOLy bit, then writing a one to this bit will clear the fault and the fault becomes inactive (FMODy and FPOLy bits are located in ”Fault Mode Register” on page 1022), else writing a one to this bit has no effect.

33.7.26 Fault Protection Value Register

Name: FPV

Access Type: Read/Write

Offset: 0x068

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FPVL3	FPVL2	FPVL1	FPVL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	FPVH3	FPVH2	FPVH1	FPVH0

This register can only be written if the WPSWS5 and WPHWS5 bits are cleared in ”Write Protect Status Register” on page 1031.

- **FPVLx: Fault Protection Value for PWML output on channel x**
  0: PWML output of channel x is forced to 0 when fault occurs.
  1: PWML output of channel x is forced to 1 when fault occurs.
- **FPVHx: Fault Protection Value for PWMH output on channel x**
  0: PWMH output of channel x is forced to 0 when fault occurs.
  1: PWMH output of channel x is forced to 1 when fault occurs.

33.7.27 Fault Protection Enable Register

Name: FPE

Access Type: Read/Write

Offset: 0x06C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
FPE3							

23	22	21	20	19	18	17	16
FPE2							

15	14	13	12	11	10	9	8
FPE1							

7	6	5	4	3	2	1	0
FPE0							

This register can only be written if the WPSWS5 and WPHWS5 bits are cleared in ”Write Protect Status Register” on page 1031.

Only the first 5 bits (number of fault input pins) of FPE0, FPE1, FPE2 and FPE3 are significant.

- **FPEx[y]: Fault Protection Enable with Fault y for channel x**
  0: Fault y is not used for the Fault Protection of the channel x.
  1: Fault y is used for the Fault Protection of the channel x.

CAUTION: To prevent an unexpected activation of the Fault Protection, the FPEx[y] bit can be written to one only if the FPOLy bit has been previously configured to its final value in ”Fault Mode Register” on page 1022.

33.7.28 Event Line x Register

Name: ELxMR

Access Type: Read/Write

Offset: 0x080 + [x * 0x04]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
CSEL7	CSEL6	CSEL5	CSEL4	CSEL3	CSEL2	CSEL1	CSEL0

- **CSELy: Comparison y Selection**
  0: A pulse is not generated on the event line x when the comparison y matches.
  1: A pulse is generated on the event line x when the comparison y matches.

33.7.29 Stepper Motor Mode Register

Name: SMMR

Access Type: Read/Write

Offset: 0x0B0

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	DOWN1	DOWN0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	GCEN1	GCEN0

- **GCENx: Gray Count Enable**
  0: Disable gray count on PWML[2*x], PWMH[2*x], PWML[2*x+1], PWMH[2*x+1].
  1: Enable gray count on PWML[2*x], PWMH[2*x], PWML[2*x+1], PWMH[2*x+1].
- **DOWNx: Down Count**
  0: Up counter.
  1: Down counter.

33.7.30 Write Protect Control Register

Name: WPCR

Access Type: Write-only

Offset: 0x0E4

Reset Value: -

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
WPRG5	WPRG4	WPRG3	WPRG2	WPRG1	WPRG0	WPCMD	

- **WPKEY: Write Protect Key**
  Should be written at value 0x50574D (“PWM” in ASCII). Writing any other value in this field aborts the write operation of the WPCMD field. Always reads as 0.
- **WPRGx: Write Protect Register Group x**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will allow to set the WPCMD command to the register group x.
- **WPCMD: Write Protect Command**

This command is performed only if the WPKEY value is correct.

0: Disable the Write Protect SW of the register groups of which the WPRGx bit is set to 1.
1: Enable the Write Protect SW of the register groups of which the WPRGx bit is set to 1.
2: Enable the Write Protect HW of the register groups of which the WPRGx bit is set to 1.
3: No effect.

Note: Only a hardware reset of the PWM controller can disable the Write Protect HW.

List of register groups:

- Register group 0:
  - ”Clock Register” on page 998
- Register group 1:
  - ”Disable Register” on page 1001
- Register group 2:
  - ”Sync Channels Mode Register” on page 1007
  - ”Channel Mode Register” on page 1037
  - ”Stepper Motor Mode Register” on page 1028
- Register group 3:

  - ”Channel Period Register” on page 1041
  - ”Channel Period Update Register” on page 1043
- Register group 4:
  - ”Channel Dead Time Register” on page 1046
  - ”Channel Dead Time Update Register” on page 1047
- Register group 5:
  - ”Fault Mode Register” on page 1022
  - ”Fault Protection Value Register” on page 1025
  -

33.7.31 Write Protect Status Register

Name: WPSR

Access Type: Read-only

Offset: 0x0E8

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
WPVSRC							
23	22	21	20	19	18	17	16
WPVSRC							
15	14	13	12	11	10	9	8
-	-	WPHWS5	WPHWS4	WPHWS3	WPHWS2	WPHWS1	WPHWS0
7	6	5	4	3	2	1	0
WPVS	-	WPSWS5	WPSWS4	WPSWS3	WPSWS2	WPSWS1	WPSWS0

- **WPVSRC: Write Protect Violation Source**
  When WPVS is active, this field indicates the write-protected register (through address offset divided by four) in which a write access has been attempted.
- **WPHWSx: Write Protect HW Status**
  0: The Write Protect HW x of the register group x is disabled.
  1: The Write Protect HW x of the register group x is enabled.
- **WPVS: Write Protect Violation Status**
  0: No Write Protect violation has occurred since the last read of the WPSR register.
  1: At least one Write Protect violation has occurred since the last read of the WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into WPVSRC field.
- **WPSWSx: Write Protect SW Status**
  0: The Write Protect SW x of the register group x is disabled.
  1: The Write Protect SW x of the register group x is enabled.

Note: Reading WPSR automatically clears WPVS and WPVSRC fields.

33.7.32 Version Register

Register Name: VERSION

Access Type: Read-only

Offset: 0x0FC

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	MFN		

15	14	13	12	11	10	9	8
-	-	-	-	VERSION			

7	6	5	4	3	2	1	0
VERSION							

- **MFN**
  Reserved. No functionality associated.
- **VERSION**
  Version number of the module. No functionality associated.

33.7.33 Comparison x Value Register

Name: CMPxV

Access Type: Read/Write

Offset: 0x130 + [x * 0x10]

Reset Value: 0x00000000

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

Only the first 20 bits (channel counter size) of CV field are significant.

- **CVM: Comparison x Value Mode**
  0: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is incrementing.
  1: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is decrementing.

Note: This bit is useless if the counter of the channel 0 is left aligned (CALG=0 in "Channel Mode Register" on page 1037)

- **CV: Comparison x Value**
  Defines the comparison x value to be compared with the counter of the channel 0.

33.7.34 Comparison x Value Update Register

Name: CMPxVUPD

Access Type: Write-only

Offset: 0x134 + [x * 0x10]

Reset Value: -

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

This register acts as a double buffer for the CV and CVM values. This prevents an unexpected comparison x match.

Only the first 20 bits (channel counter size) of CVUPD field are significant.

- **CVMUPD: Comparison x Value Mode Update**
  0: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is incrementing.
  1: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is decremented.

Note: This bit is useless if the counter of the channel 0 is left aligned (CALG=0 in "Channel Mode Register" on page 1037)

- **CVUPD: Comparison x Value Update**
  Defines the comparison x value to be compared with the counter of the channel 0.

CAUTION: to be taken into account, the write of the CMPxVUPD register must be followed by a write of the CMPxMUPD register.

33.7.35 Comparison x Mode Register

Name: CMPxM

Access Type: Read/Write

Offset: 0x138 + [x * 0x10]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
CUPRCNT				CUPR			

15	14	13	12	11	10	9	8
CPRCNT				CPR			

7	6	5	4	3	2	1	0
CTR				-	-	-	CEN

- **CUPRCNT: Comparison x Update Period Counter**
  Reports the value of the comparison x update period counter.

Note: The CUPRCNT field is read-only.

- **CUPR: Comparison x Update Period**
  Defines the time between each update of the comparison x mode and the comparison x value. This time is equal to CUPR+1 periods of the channel 0 counter.
- **CPRCNT: Comparison x Period Counter**
  Reports the value of the comparison x period counter.

Note: The CPRCNT field is read-only.

- **CPR: Comparison x Period**
  Defines the maximum value of the comparison x period counter (CPRCNT). The comparison x value is performed periodically once every CPR+1 periods of the channel 0 counter.
- **CTR: Comparison x Trigger**
  The comparison x is performed when the value of the comparison x period counter (CPRCNT) reaches the value defined by CTR.
- **CEN: Comparison x Enable**
  0: The comparison x is disabled and can not match.
  1: The comparison x is enabled and can match.

33.7.36 PWM Comparison x Mode Update Register

Name: CMPxMUPD

Access Type: Write-only

Offset: 0x13C + [x * 0x10]

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	CUPRUPD			

15	14	13	12	11	10	9	8
-	-	-	-	CPRUPD			

7	6	5	4	3	2	1	0
CTRUPD				-	-	-	CENUPD

This register acts as a double buffer for the CEN, CTR, CPR and CUPR values. This prevents an unexpected comparison x match.

- **CUPRUPD: Comparison x Update Period Update**
  Defines the time between each update of the comparison x mode and the comparison x value. This time is equal to CUPR+1 periods of the channel 0 counter.
- **CPRUPD: Comparison x Period Update**
  Defines the maximum value of the comparison x period counter (CPRCNT). The comparison x value is performed periodically once every CPR+1 periods of the channel 0 counter.
- **CTRUPD: Comparison x Trigger Update**
  The comparison x is performed when the value of the comparison x period counter (CPRCNT) reaches the value defined by CTR.
- **CENUPD: Comparison x Enable Update**
  0: The comparison x is disabled and can not match.
  1: The comparison x is enabled and can match.

33.7.37 Channel Mode Register

Name: CMR

Access Type: Read/Write

Offset: 0x200 + [ch_num * 0x20]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	DTLI	DTHI	DTE
15	14	13	12	11	10	9	8
-	-	-	-	-	CES	CPOL	CALG
7	6	5	4	3	2	1	0
-	-	-	-	CPRE			

This register can only be written if the WPSWS2 and WPHWS2 bits are cleared in ”Write Protect Status Register” on page 1031.

- **DTLI: Dead-Time PWMLx Output Inverted**
  0: The dead-time PWMLx output is not inverted.
  1: The dead-time PWMLx output is inverted.
- **DTHI: Dead-Time PWMHx Output Inverted**
  0: The dead-time PWMHx output is not inverted.
  1: The dead-time PWMHx output is inverted.
- **DTE: Dead-Time Generator Enable**
  0: The dead-time generator is disabled.
  1: The dead-time generator is enabled.
- **CES: Counter Event Selection**
  The CES bit defines when the channel counter event occurs when the period is center aligned (CHIDx in the ”Interrupt Status Register 1” on page 1006).

  CALG=0 (Left Alignment):

  0/1: The channel counter event occurs at the end of the PWM period.

  CALG=1 (Center Alignment):

  0: The channel counter event occurs at the end of the PWM period.
  1: The channel counter event occurs at the end of the PWM period and at half the PWM period.
- **CPOL: Channel Polarity**
  0: The OCx output waveform (output from the comparator) starts at a low level.
  1: The OCx output waveform (output from the comparator) starts at a high level.
- **CALG: Channel Alignment**
  0: The period is left aligned.
  1: The period is center aligned.

- **CPRE: Channel Pre-scaler**

Table 33-7. Channel Pre-scaler

CPRE	Channel Pre-scaler
0	CCK
1	CCK/2
2	CCK/4
3	CCK/8
4	CCK/16
5	CCK/32
6	CCK/64
7	CCK/128
8	CCK/256
9	CCK/512
10	CCK/1024
11	CLKA
12	CLKB
Other	Reserved

33.7.38 Channel Duty Cycle Register

Name: CDTY

Access Type: Read/Write

Offset: 0x204 + [ch_num * 0x20]

Reset Value: 0x00000000

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

Only the first 20 bits (channel counter size) are significant.

- **CDTY: Channel Duty-Cycle**
  Defines the waveform duty-cycle. This value must be defined between 0 and CPRD (CPRx).

33.7.39 Channel Duty Cycle Update Register

Name: CDTYUPD

Access Type: Write-only

Offset: 0x208 + [ch_num * 0x20]

Reset Value: -

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

This register acts as a double buffer for the CDTY value. This prevents an unexpected waveform when modifying the waveform duty-cycle.

Only the first 20 bits (channel counter size) are significant.

- **CDTYUPD: Channel Duty-Cycle Update**
  Defines the waveform duty-cycle. This value must be defined between 0 and CPRD (CPRx).

33.7.40 Channel Period Register

Name: CPRD

Access Type: Read/Write

Offset: 0x20C + [ch_num * 0x20]

Reset Value: 0x00000000

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

This register can only be written if the WPSWS3 and WPHWS3 bits are cleared in ”Write Protect Status Register” on page 1031.

Only the first 20 bits (channel counter size) are significant.

- **CPRD: Channel Period**

If the waveform is left-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM internal clock (CCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRD)}{CCK}$$

– By using the PWM internal clock (CCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{CCK} \quad or \quad \frac{(CRPD \times DIVB)}{CCK}$$

If the waveform is center-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM internal clock (CCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{CCK}$$

– By using the PWM internal clock (CCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{CCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{CCK}$$

33.7.41 Channel Period Update Register

Name: CPRDUPD

Access Type: Write-only

Offset: 0x210 + [ch_num * 0x20]

Reset Value: -

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

This register can only be written if the WPSWS3 and WPHWS3 bits are cleared in ”Write Protect Status Register” on page 1031.

This register acts as a double buffer for the CPRD value. This prevents an unexpected waveform when modifying the waveform period.

Only the first 20 bits (channel counter size) are significant.

- **CPRDUPD: Channel Period Update**

If the waveform is left-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM internal clock (CCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRDUPD)}{CCK}$$

– By using the PWM internal clock (CCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPDUPD \times DIVA)}{CCK} \quad or \quad \frac{(CRPDUPD \times DIVB)}{CCK}$$

If the waveform is center-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM internal clock (CCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRDUPD)}{CCK}$$

– By using the PWM internal clock (CCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRDUPD \times DIVA)}{CCK} \text{ or } \frac{(2 \times CPRDUPD \times DIVB)}{CCK}$$

33.7.42 Channel Counter Register

Name: CCNT

Access Type: Read-only

Offset: 0x214 + [ch_num * 0x20]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
CNT							

15	14	13	12	11	10	9	8
CNT							

7	6	5	4	3	2	1	0
CNT							

Only the first 20 bits (channel counter size) are significant.

- **CNT: Channel Counter Register**
  Gives the channel counter value. This register is reset when the channel counter reaches the CPRD value defined in the CPRDx register if the waveform is left aligned.

33.7.43 Channel Dead Time Register

Name: DT

Access Type: Read/Write

Offset: 0x218 + [ch_num * 0x20]

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DTL							

23	22	21	20	19	18	17	16
DTL							

15	14	13	12	11	10	9	8
DTH							

7	6	5	4	3	2	1	0
DTH							

This register can only be written if the WPSWS4 and WPHWS4 bits are cleared in ”Write Protect Status Register” on page 1031.

Only the first 16 bits (dead-time counter size) of DTH and DTL fields are significant.

- **DTL: Dead-Time Value for PWMLx Output**
  Defines the dead-time value for PWMLx output. This value must be defined between 0 and CDTY (CDTYx).
- **DTH: Dead-Time Value for PWMHx Output**
  Defines the dead-time value for PWMHx output. This value must be defined between 0 and CPRD-CDTY (CPRx and CDTYx).

33.7.44 Channel Dead Time Update Register

Name: DTUPD

Access Type: Write-only

Offset: 0x21C + [ch_num * 0x20]

Reset Value: -

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

This register can only be written if the WPSWS4 and WPHWS4 bits are cleared in ”Write Protect Status Register” on page 1031.

This register acts as a double buffer for the DTH and DTL values. This prevents an unexpected waveform when modifying the dead-time values.

Only the first 16 bits (dead-time counter size) of DTHUPD and DTLUPD fields are significant.

- **DTLUPD: Dead-Time Value Update for PWMLx Output**
  Defines the dead-time value for PWMLx output. This value must be defined between 0 and CDTY (CDTYx). This value is applied only at the beginning of the next channel x PWM period.
- **DTHUPD: Dead-Time Value Update for PWMHx Output**
  Defines the dead-time value for PWMHx output. This value must be defined between 0 and CPRD-CDTY (CPRx and CDTYx). This value is applied only at the beginning of the next channel x PWM period.

33.8 Module Configuration

The specific configuration for each PWM instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 33-8. PWM Clock Name

Module name	Clock Name	Description
PWM	CLK_PWM	Peripheral Bus clock from the PBA clock domain
	GCLK	The generic clock used for the PWM is GCLK4

Table 33-9. Register Reset Values

Register	Reset Value
VERSION	0x00000501

33.8.1 PWM fault sources

The following table define the mapping of the PWM fault sources. For details on PWM faults, see Fault Protection paragraph in the PWM chapter.

Table 33-10. PWM fault sources

fault input number	Description
0	PEVC channel output 8, this fault is only usable when the FMR.FMODy is set to 1.
1	PEVC channel output 9, this fault is only usable when the FMR.FMODy is set to 1.
2	EXT_FAULTS[0] input pin, See Peripheral Multiplexing on I/O line chapter.
3	EXT_FAULTS[1] input pin, See Peripheral Multiplexing on I/O line chapter.
4	clock failure detector of PM, See PM chapter for details.

34. Quadrature Decoder (QDEC)

Rev.: 1.0.0.0

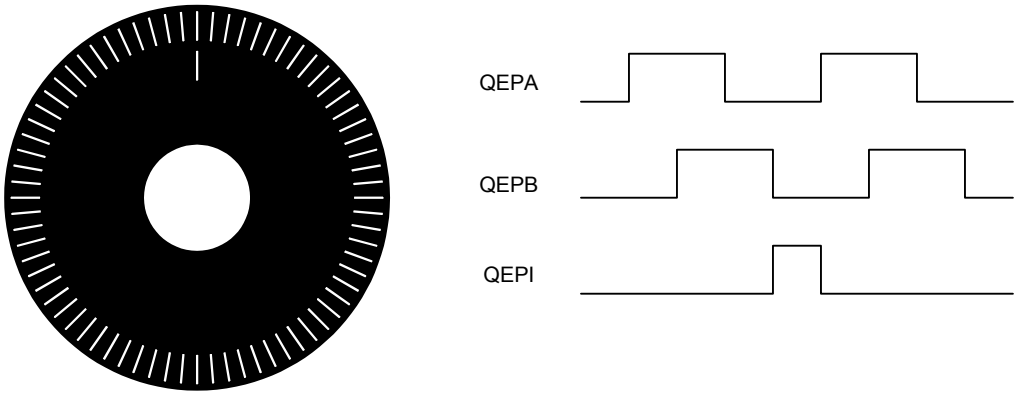
34.1 Features

- **Handles three input channels:**
  - **Two phase signals (QEPA, QEPB)**
  - **One index pulse (QEPI)**
- **Optional digital filter on inputs**
- **16-bit position counter and 16-bit revolution counter**
- **32-bit timer/counter mode**
- **Software trigger or peripheral event trigger**
- **Compare function with peripheral event generation**
- **Capture function on peripheral event**

34.2 Overview

The QDEC is used in rotating motion systems for position and speed detection. It decodes quadrature signals, normally two 90 degrees out-of-phases pulses, into count and direction informations. The quadrature signals are usually generated by a wheel with periodic transparent gaps (a.k.a. lines) and an optical system composed of one light source and 2 sensors.

Figure 34-1. Quadrature Signals Description

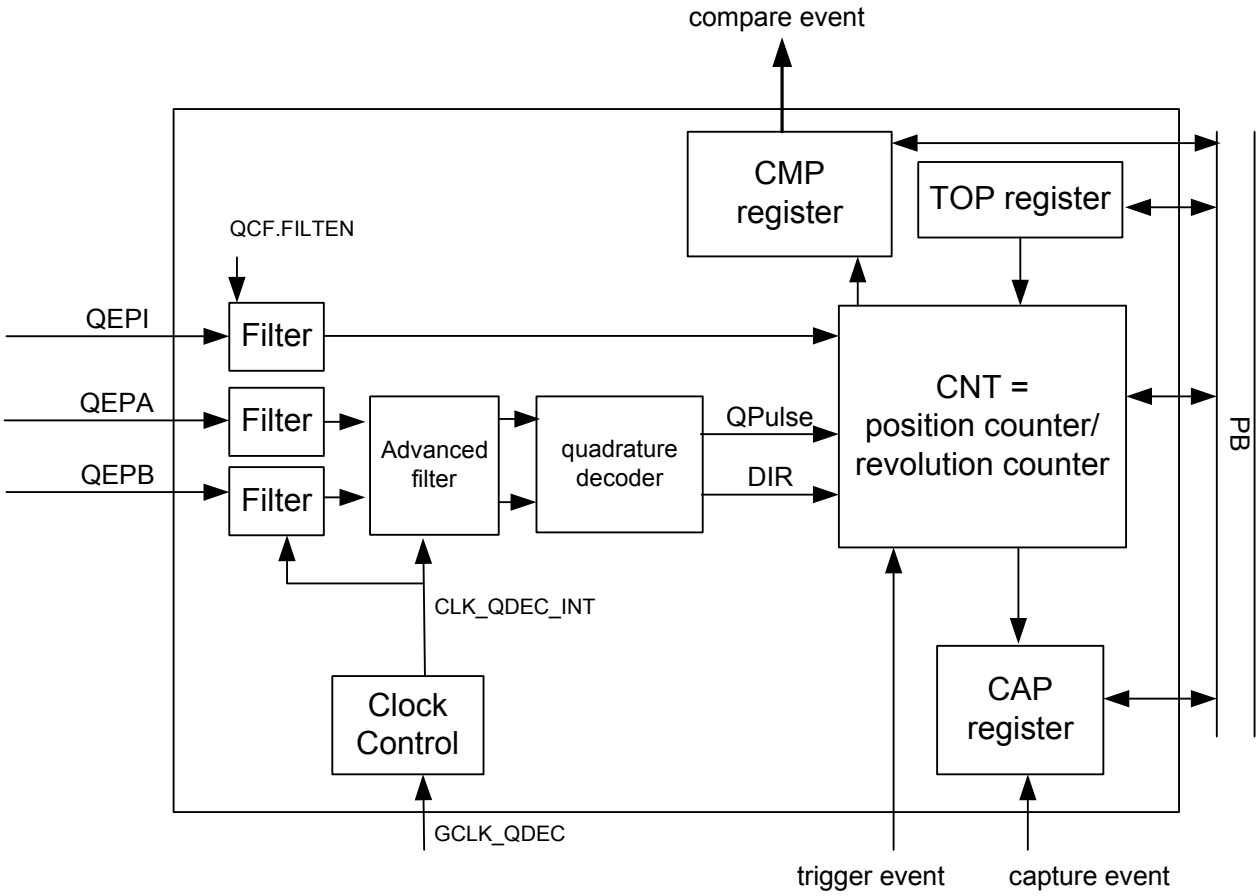


The phase signals are labelled QEPA and QEPB. If QEPA leads QEPB, the rotation direction is defined as positive. If QEPB leads QEPA, the rotation direction is defined as negative. Each time a line passes the sensor, a counter register (CNT.PC) is incremented or decremented depending of the rotation direction. A third input signal, QEPI, can be used to reset CNT.PC.

QDEC can count the total number of revolutions. A top value of CNT.PC is written to the Position Counter Top Value in the Top Value register (TOP.PCTOP). When CNT.PC counts up to this value, it wraps around to zero and the Revolution Counter field (CNT.RC) is incremented. CNT.RC is decremented when CNT.PC counts down and crosses the zero value. CNT.PC will then be reset to the value in TOP.PCTOP.

34.3 Block Diagram

Figure 34-2. QDEC Block Diagram



34.4 I/O Lines Description

Table 34-1. I/O Lines Description

Pin Name	Pin Description	Type
QEPA	Quadrature phase signal A	Input
QEPB	Quadrature phase signal B	Input
QEPI	Quadrature index signal	Input

34.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

34.5.1 I/O Lines

The QDEC pins are multiplexed with other peripherals. The user must first program the I/O Controller to give control of the pins to the QDEC.

34.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the QDEC, the QDEC will stop functioning and resume operation after the system wakes up from sleep mode.

34.5.3 Clocks

The QDEC has two clocks connected: One Peripheral Bus clock (CLK_QDEC) and one generic clock (GCLK_QDEC). These clocks are generated by the Power Manager. CLK_QDEC is enabled at reset, and can be disabled in the Power Manager.

GCLK_QDEC is used for filtering in QDEC mode and is the timer clock in Timer Mode. It is a generic clock generated by the Power Manager. The programmer must configure the Power Manager to enable GCLK_QDEC. The GCLK_QDEC frequency must less than half the CLK_QDEC clock frequency.

34.5.4 Interrupts

The QDEC interrupt request line is connected to the interrupt controller. Using the QDEC interrupt requires the interrupt controller to be programmed first.

34.5.5 Peripheral Events

The QDEC peripheral events are connected via the Peripheral Event System. Refer to the Peripheral Event System chapter for details.

34.5.6 Debug Operation

When an external debugger forces the CPU into debug mode, the QDEC continues normal operation; the timer is not frozen, but peripheral events are masked.

In OCD mode, the timer is not frozen and the events are masked. Reading the CAP register does not clear the reminding of last capture event for the OVR interrupt.

If the QDEC is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

The QDEC peripheral events are masked during debug operation, unless the Run In Debug bit in the Development Control Register is written to one and the bit corresponding to the QDEC is written to one in the Peripheral Debug Register (PDBG). Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual, and the OCD Module Configuration section, for details.

34.6 Functional Description

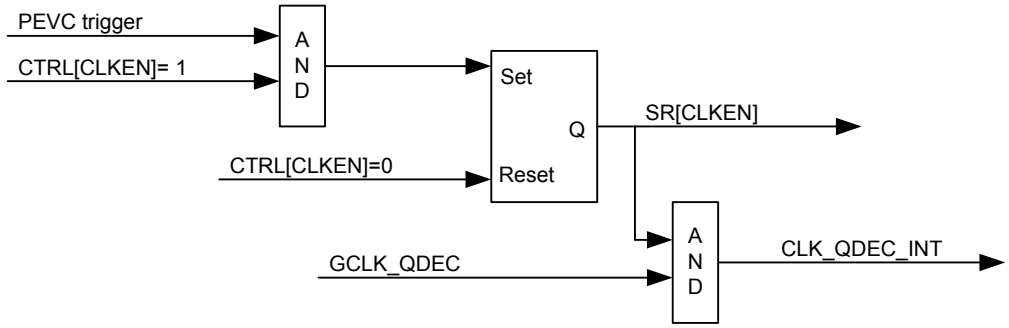
34.6.1 Basic Operation

34.6.1.1 Enabling the QDEC

The QDEC is enabled by writing a one to the Clock Enable bit in the Control Register (CTRL.CLKEN). This will also enable the internal CLK_QDEC_INT. This clock is generated from GCLK_QDEC.

CLK_QDEC_INT is used in the filter blocks and for clocking the counter in Timer Mode. A software trigger or peripheral event trigger is needed for CLK_QDEC_INT to start. The Clock Enable bit in the Status Register (SR.CLKEN) indicates if the clock is running.

Figure 34-3. Clock Control



34.6.1.2 Trigger

A trigger resets the QDEC counter and starts CLK_QDEC_INT. Two triggers are possible:

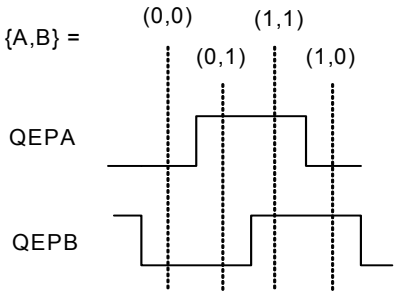
- A software trigger, by writing a one to the Software Trigger bit in CTRL (CTRL.SWTRG).
- Trigger peripheral event from the PEVC: If enabled by writing a one to the Event Trigger Enable bit in the Configuration Register (CF.EVTRGE).

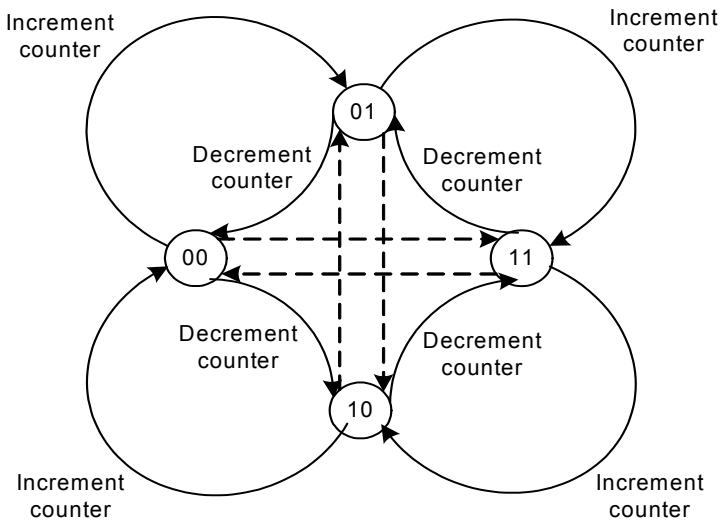
The QDEC counter is reset when the peripheral trigger event occurs.

34.6.1.3 Quadrature decoder logic and digital filter

The quadrature decoder logic converts the 2-phase signals QEPA and QEPB in a count pulse signal (QPulse) for each transition and a DIR signal to indicate the rotation direction.

Figure 34-4. Quadrature Description





The QEPI signal may be used to detect a reference position once per revolution.

The 3 inputs (QEPA/QEPB/QEPI) can be inverted by writing to appropriate bits in CF.

34.6.1.4 Position counter

The 16-bit position counter is incremented or decremented on every count pulse, generated by the quadrature decoder module. The counting direction is displayed in SR.CNTDIR.

If the position counter reaches the TOP.PCTOP value when counting up or the 0 value when counting down, the counter wraps around. The Position Counter Roll Over (PCRO) interrupt is generated.

Usually, the TOP.PCTOP value should always be set to the total number of quadrature states minus one, which is (quadrature_encoder_lines_count *4) -1.

34.6.1.5 Index detection

The optional index signal QEPI may be used to correct the position counter if quadrature states have been missed due to noise. If the Index Enable bit (CF.IDXE) is written to one, detection of the QEPI signal will reset the position counter to 0 on a selected phase of quadrature signals. This selected phase is configured via the Detection Phase field (CF.IDXPHS).

The Index Error (IDXERR) interrupt is triggered if the index detection occurs and the position counter value is not 0.

34.6.1.6 Revolution counter

The revolution counter is incremented/decremented each time the position counter rolls over. It rolls over when it reaches the TOP.RCTOP value.

34.6.1.7 Waveforms

Figure 34-5. Quadrature Clock and Direction Decoding (TOP.PCTOP = 4)

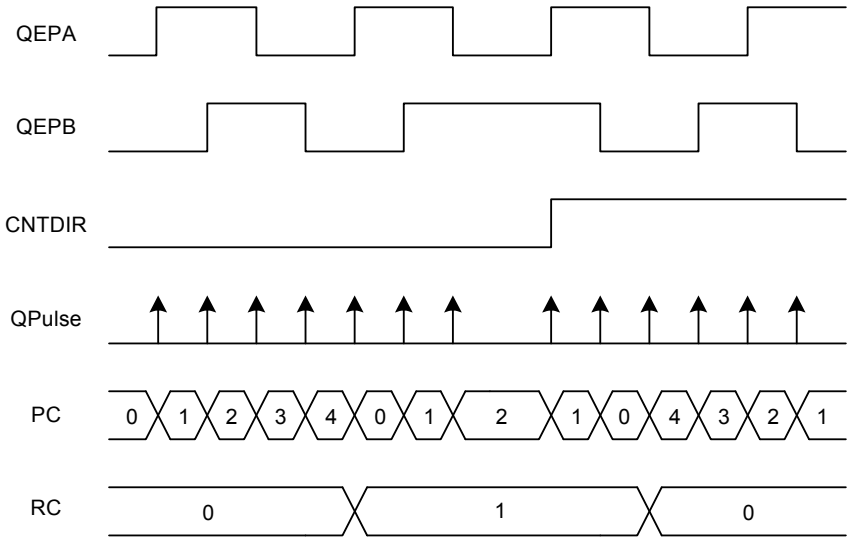
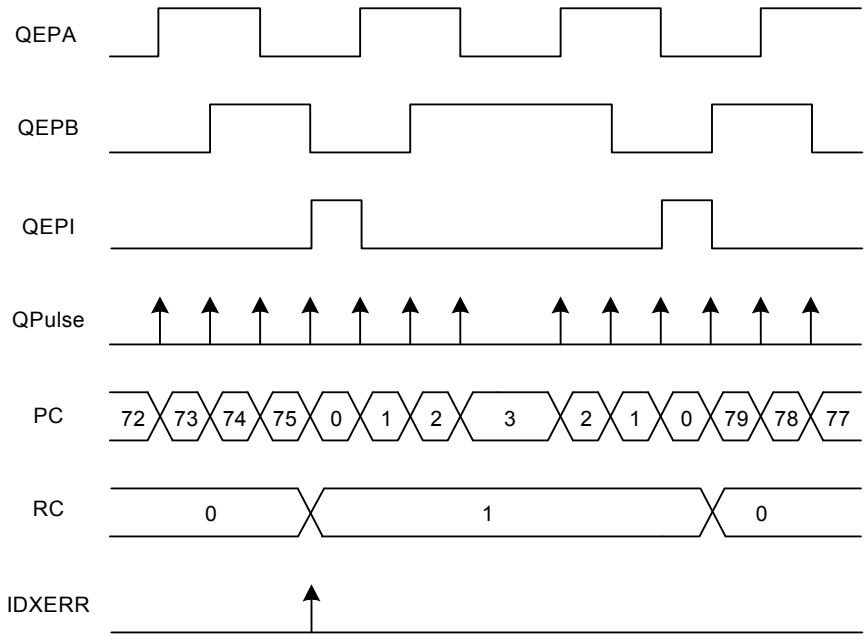


Figure 34-6. PC reset by QEPI signal (TOP.PCTOP = 79, CF.IDXPHS =”00”, CF.IDXE = “1”)



34.6.1.8 Quadrature frequency

The CLK_QDEC clock frequency must be at least two times the QEPA and QEPB frequency as these signals are synchronized to the CLK_QDEC clock. To get the maximum available frequency on QEPA/QEPB signals, the filter on inputs should be bypassed.

For a 33 MHz peripheral bus clock the maximum QEPA frequency is 16.5 MHz. For a wheel with 8192 lines the maximum rotational speed supported by QDEC is 16.5MHz / 8192 = 2014 rps = 120 849 rpm.

34.6.1.9 Disabling the QDEC

The QDEC is disabled by writing a zero to CTRL.CLKEN.

34.6.2 Advanced Operation

34.6.2.1 Compare register

The Compare register (CMP) is used to generate an interrupt and a peripheral event when the CNT register reaches the value defined in CMP.

If RC compare is enabled (CF.RCCE is one), a compare match occurs when RC is equal to RCCMP. A peripheral event is generated and the CMP interrupt line is set if enabled.

If the PC compare is enabled (CF.PCCE is one), a compare match occurs when the PC is equal to PCCMP. A peripheral event is generated and the CMP interrupt line is set if enabled.

If both RC compare and PC compare are enabled, a compare match occurs when CNT is equal to CMP. A peripheral event is generated and the CMP interrupt line is set if enabled.

The compare peripheral event should be mapped through the PEVC to another peripheral.

34.6.2.2 Capture register

The capture function saves the QDEC counter value in the Capture register (CAP) when a capture event has occurred. The capture function is enabled if the QDEC counter is running.

The CAP register will not be updated with a new value if the previous value has not been read. If a capture event occurs and the previous value has not been read, the SR.OVR bit is set.

34.6.2.3 Glitch filter

The QDEC inputs (QEPA/QEPB/QEPI) are passed through a glitch filter that is enabled by writing a one to the CF.FILTEN bit. The input sent to the QDEC counter will toggle if the input is stable for three CLK_QDEC_INT periods.

34.6.2.4 Timer/Counter mode

QDEC can be used as a 32-bit/counter with compare/capture capabilities. This timer includes an up/down (UPD) mode where the timer counts up or down according to a toggle direction event from the PEVC.

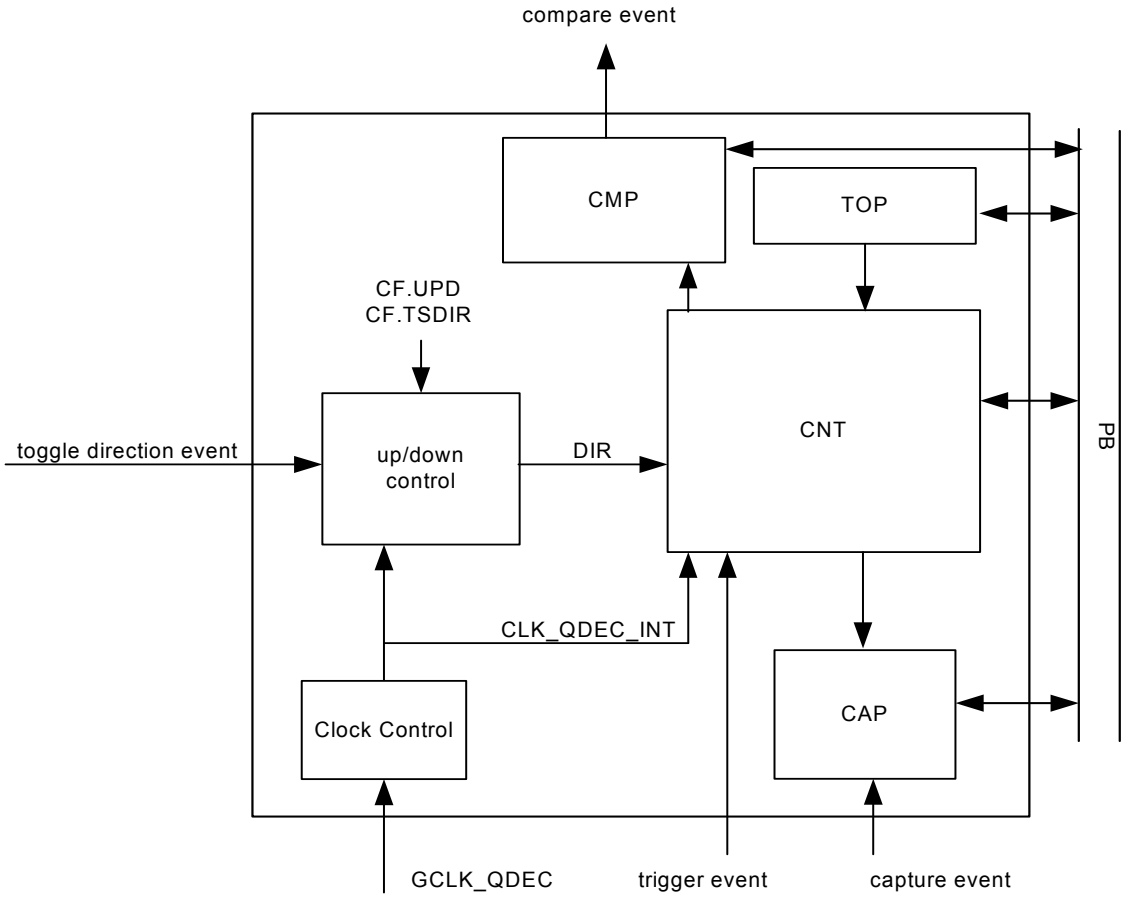
The timer/counter is available by writing a zero to the CF.QDEC bit. Timer/Counter mode uses the same resources as QDEC mode:

- The CNT QDEC counter
- The TOP register to reload the CNT value
- The CMP register to generate a compare peripheral event/interrupt
- The CAP register to save the CNT value in case of a capture peripheral event occurs
- The clock selection
- The trigger mechanism

It does not use the input filters and the index pulse control.

The timer/counter includes an up/down mode that is enabled by writing a one to the CF.UPD bit.

Figure 34-7. Timer Block Diagram



The user can set the timer count direction by writing to the CF.TSDIR bit. The counting direction is taken into account when a trigger occurs. The counting direction is shown in SR.CNTDIR.

The user has to set the initial direction of counting by writing to CF.TSDIR. When the timer is triggered, i.e. each time a toggle direction event occurs, the counter changes counting direction. If the counter reaches 0, it will be stuck at 0 as long as the counting direction is down.

34.6.3 Interrupts

The QDEC has one interrupt request line connected to the interrupt controller. The sources of this interrupt are:

- The QEPI interrupt to detect the index signal.
- The CMP interrupt to detect a compare match.
- The CAP interrupt to detect that the QDEC counter value has been saved in the CAP register due to the capture event.
- The OVR interrupt to detect that a capture event was received without the CAP register having been read since the last capture event. In OCD mode, reading the CAP register does not clear the memory of the last capture event.
- The PCRO interrupt to detect a roll-over of the position counter. In Timer/Counter mode, the roll over occurs when (PC = 0xFFFF and RC != RCTOP), when PC = TOP.PCTOP or when RC = RCTOP.

- The RCRO interrupt to detect a roll-over of the revolution counter.
- The IDXERR interrupt to detect that the index signal (QEPI) is detected and the position counter does not have the expected value (TOP.PCTOP if the counter counts up, 1 if the counter counts down).
- The DIRINV interrupt occurs when the count direction changes.
- The QDERR interrupt occurs when a bad transition in the quadrature signals is detected (for example, from “00” to “11”). This could be caused by erroneous programming of the GCLK_QDEC frequency.
- The TRIGGER interrupt occurs when a trigger event from PEVC is detected. It could be used by software to detect a reset of the counters.
-

Each interrupt source can be enabled by writing a one to the corresponding bit in the Interrupt Enable Register (IER) and disabled by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The enable status can be read from the Interrupt Mask Register (IMR). The status of the interrupt sources, even if the interrupt is masked, can be read in SR. When an interrupt has occurred, it is reset by writing a one to the corresponding bit in the Status Clear Register (SCR).

34.6.4 Peripheral Events

The QDEC can receive three peripheral events from the Peripheral Event Controller (PEVC):

- The trigger peripheral event starts CLK_QDEC_INT and enables the counter.
- The capture peripheral event captures CNT in the Capture register (CAP).
- The toggle_dir peripheral event toggles the count direction when the QDEC works in Timer mode with UPD mode active.

The QDEC can send one event to the PEVC:

- The compare peripheral event when the CNT register reaches the Compare register (CMP) value.

The PEVC must be programmed to enable QDEC peripheral events.

34.7 User Interface

Table 34-2. QDEC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CTRL	Read/Write	0x00000000
0x04	Configuration Register	CF	Read/Write	0x00000000
0x08	Counter Register	CNT	Read/Write	0x00000000
0x0C	Top Value Register	TOP	Read/Write	0x00000000
0x10	Compare Register	CMP	Read/Write	0x00000000
0x14	Capture Register	CAP	Read-only	0x00000000
0x18	Status Register	SR	Read-only	0x00000000
0x1C	Status Clear Register	SCR	Write-only	0x00000000
0x20	Interrupt Mask Register	IMR	Read-only	0x00000000
0x24	Interrupt Enable Register	IER	Write-only	0x00000000
0x28	Interrupt Disable Register	IDR	Write-only	0x00000000
0x2C	Parameter Register	PARAMETER	Read-only	- (1)
0x30	Version Register	VERSION	Read-only	- (1)

1. The reset values for this register is device specific. Please refer to the Module Configuration section at the end of this chapter.

34.7.1 Control Register

Name: CTRL

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	SWTRG	CLKEN

- **SWTRG: Software Trigger**
  Writing a one to this bit generates a software trigger if CTRL.CLKEN is one.
  This bit always reads as 0.
- **CLKEN: QDEC Module and Clock Enable**
  Writing a zero to this bit disables the QDEC and CLK_QDEC_INT clock.
  Writing a one to this bit enables the QDEC and CLK_QDEC_INT clock.

34.7.2 Configuration Register

Name: CF

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	UPD	TSDIR

15	14	13	12	11	10	9	8
-	-	FILTEN	IDXPHS		IDXINV	PHSINVB	PHSINVA

7	6	5	4	3	2	1	0
-	-	-	EVTRGE	RCCE	PCCE	IDXE	QDEC

- **UPD: Up/Down Timer Mode**
  0: Up/Down functionality in Timer Mode is disabled
  1: Up/Down functionality in Timer Mode is ensabled
- **TSDIR: Timer Set Direction**
  0: The counters count up in Timer Mode
  1: The counters count down in Timer Mode
  The count direction is updated when a trigger (software or hardware) occurs
- **FILTEN: Input Digital Filter Enable**
  0: The input digital filter is disabled
  1: The input digital filter is enabled
- **IDXPHS: QEPI Detection Phase**
  0: QEPI detection enabled when QEPA signal equals 0 and QEPB signal equals 0
  1: QEPI detection enabled when QEPA signal equals 0 and QEPB signal equals 1
  2: QEPI detection enabled when QEPA signal equals 1 and QEPB signal equals 0
  3: QEPI detection enabled when QEPA signal equals 1 and QEPB signal equals 1
- **IDXINV: QEPI Phase**
  0: QEPI will not be inverted
  1: QEPI will be inverted
- **PHSINVB: QEPB Phase**
  0: QEPB will not be inverted
  1: QEPB will be inverted
- **PHSINVA: QEPA Phase**
  0: QEPA will not be inverted
  1: QEPA will be inverted
- **EVTRGE: Event Trigger Enable**
  0: The event trigger function is disabled
  1: The event trigger function is enabled

- **RCCE: Revolution Counter Compare Enable**
  0: The revolution counter compare is disabled
  1: The revolution counter compare is enabled
- **PCCE: Position Counter Compare Enable**
  0: The position counter compare is disabled
  1: The position counter compare is enabled
- **IDXE: Index Enable**
  0: The index signal detection is disabled
  1: The index signal detection is enabled
- **QDEC: QDEC Mode**
  0: QDEC is in Timer Mode
  1: QDEC is in Quadrature Decoder Mode

34.7.3 Counter Register

Name: CNT

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RC[15:8]							

23	22	21	20	19	18	17	16
RC[7:0]							

15	14	13	12	11	10	9	8
PC[15:8]							

7	6	5	4	3	2	1	0
PC[7:0]							

- **RC: Revolution Counter**
  The current value of the revolution counter
- **PC: Position Counter**
  The current value of the position counter

34.7.4 Top Register

Name: TOP

Access Type: Read/Write

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RCTOP[15:8]							

23	22	21	20	19	18	17	16
RCTOP[7:0]							

15	14	13	12	11	10	9	8
PCTOP[15:8]							

7	6	5	4	3	2	1	0
PCTOP[7:0]							

- **RCTOP: Revolution Counter Top Value**
  The top value of the revolution counter
- **PCTOP: Position Counter Top Value**
  The top value of the position counter

34.7.5 Compare Register

Name: CMP

Access Type: Read/Write

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RCCMP[15:8]							

23	22	21	20	19	18	17	16
RCCMP[7:0]							

15	14	13	12	11	10	9	8
PCCMP[15:8]							

7	6	5	4	3	2	1	0
PCCMP[7:0]							

- **RCCMP: Revolution Counter Compare Value**
  The Revolution Counter value that generates a compare event
- **PCCMP: Position Counter Compare Value**
  The Position Counter value that generates a compare event

34.7.6 Capture Register

Name: CAP

Access Type: Read-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RCCAP[15:8]							

23	22	21	20	19	18	17	16
RCCAP[7:0]							

15	14	13	12	11	10	9	8
PCCAP[15:8]							

7	6	5	4	3	2	1	0
PCCAP[7:0]							

- **RCCAP: Revolution Capture**
  The last capture value of the revolution counter
- **PCCAP: Position Capture**
  The last capture value of the position counter

34.7.7 Status Register

Name: SR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
-	-	-	-	-	-	CLKEN	CNTDIR

15	14	13	12	11	10	9	8
-	-	-	-	-	-	TRIGGER	QDERR

7	6	5	4	3	2	1	0
OVR	DIRINV	IDXERR	RCRO	PCRO	CAP	CMP	QEPI

- **CLKEN: QDEC Counter Clock Status**
  This bit is cleared when the QDEC and CLK_QDEC_INT has been disabled
  This bit is set when the QDEC and CLK_QDEC_INT has been enabled
- **CNTDIR: Counter Direction**
  This bit is cleared when the counter counts up
  This bit is set when the counter counts down
- **TRIGGER: Trigger Event Occurrence**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when the trigger event has occurred
- **QDERR: Illegal Quadrature Signals Transition**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when an illegal transition of quadrature signals has occurred
- **OVR: Overrun Capture**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when the overrun capture event has occurred
- **DIRINV: Count Direction Inversion**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when the count direction has changed
- **IDXERR: Index Error**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when an index error has occurred
- **RCRO: Revolution Counter Roll Over**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when the revolution counter has rolled over
- **PCRO: Position Counter Roll Over**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when the position counter has rolled over

- **CAP: Counter Capture**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when a capture event has occurred
- **CMP: Counter Compare**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when compare match occurred
- **QEPI: Index Signal Detection**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when an index detection has occurred

34.7.8 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TRIGGER	QDERR

7	6	5	4	3	2	1	0
OVR	DIRINV	IDXERR	RCRO	PCRO	CAP	CMP	QEPI

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

34.7.9 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TRIGGER	QDERR

7	6	5	4	3	2	1	0
OVR	DIRINV	IDXERR	RCRO	PCRO	CAP	CMP	QEPI

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

34.7.10 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x28

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TRIGGER	QDERR

7	6	5	4	3	2	1	0
OVR	DIRINV	IDXERR	RCRO	PCRO	CAP	CMP	QEPI

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

34.7.11 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x2C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TRIGGER	QDERR
7	6	5	4	3	2	1	0
OVR	DIRINV	IDXERR	RCRO	PCRO	CAP	CMP	QEPI

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

34.7.12 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x30

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
RCSIZE				PCSIZE			

- **RCSIZE:**
  Number of bits -1 in CNT.RC registers
- **PCSIZE:**
  Number of bits -1 in CNT.PC registers

34.7.13 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x34

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	VARIANT			

15	14	13	12	11	10	9	8
–	–	–	–	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

34.8 Module Configuration

The specific configuration for each QDEC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 34-3. QDEC Clock Name

Module Name	Clock Name	Description
QDEC0	CLK_QDEC0	Peripheral Bus clock from the PBA clock domain
	GCLK_QDEC0	The generic clock used for the QDEC0 is GCLK5
QDEC1	CLK_QDEC1	Peripheral Bus clock from the PBA clock domain
	GCLK_QDEC1	The generic clock used for the QDEC1 is GCLK6

Table 34-4. Register Reset Values

Register	Reset Value
VERSION	0x00000100
PARAMETER	0x000000FF

35. Analog Comparator Interface (ACIFA)

Rev: 1.0.0.0

35.1 Features

- **Control one set of two analog comparators**
- **High speed option versus low power option**
  - **shortest propagation delay/highest current consumption**
  - **longest propagation delay/lowest current consumption**
- **Selectable input hysteresis:**
  - **0 mV, 20 mV, 50 mV**
- **Input selection between external input pin and internal inputs**
- **Window function**
- **Interrupt on:**
  - **Rising edge, Falling edge, toggle**
  - **Signal above/below window, signal inside/outside window**
  - **startup time**
- **Two Analog comparators interface events available on pin through PEVC**

35.2 Overview

The Analog Comparator Interface (ACIFA) is able to control two Analog Comparators (AC) with identical behavior. An Analog Comparator compares two voltages and gives a compare output depending on this comparison.

The ACIFA can be configured in normal mode (see Figure 35-1 on page 1076) or in window mode (see Figure 35-2 on page 1076).

The AC's Inputs are programmable between internal inputs (DAC, voltage reference, ...) and external input pins.

According to the comparison result, each comparator can trigger a separate interrupt in normal mode. In window function, an additional interrupt can be triggered, depending if the voltage to be compared is inside or outside the window.

The ACIFA is able to generate two output events that can be used through PEVC to trigger a hardware process.

35.3 Block Diagram

Figure 35-1. Analog Comparator Interface Overview in normal mode

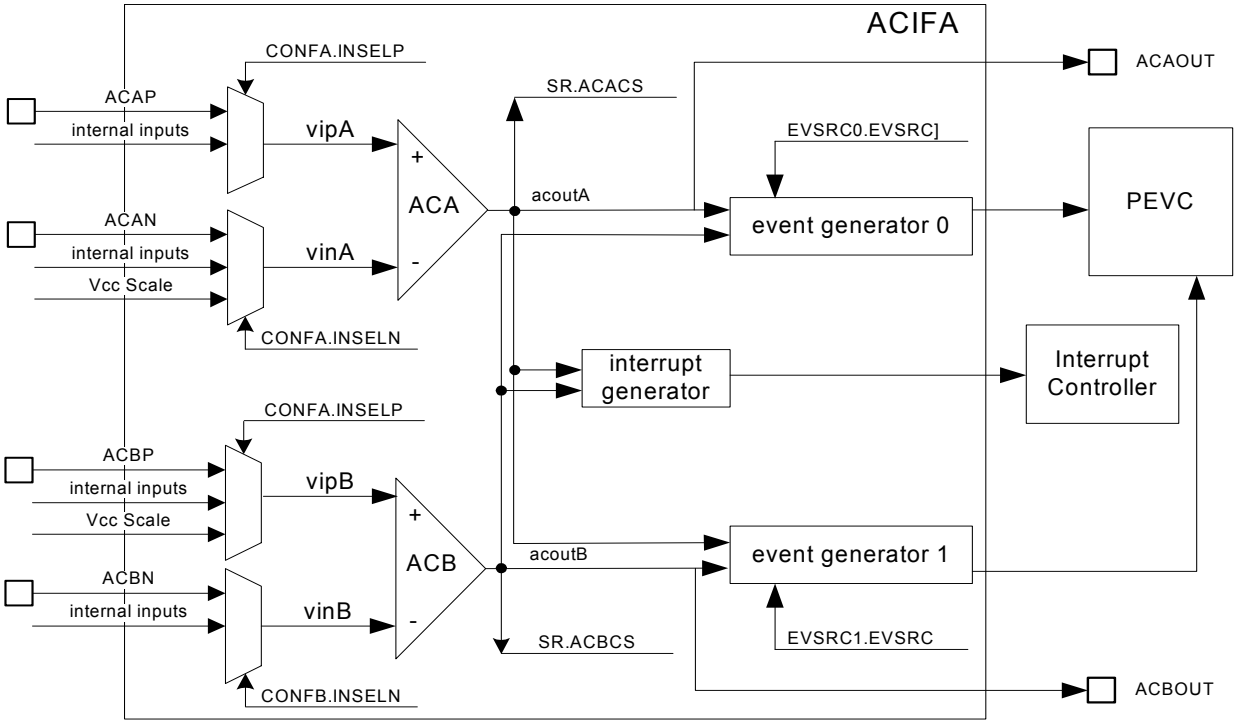
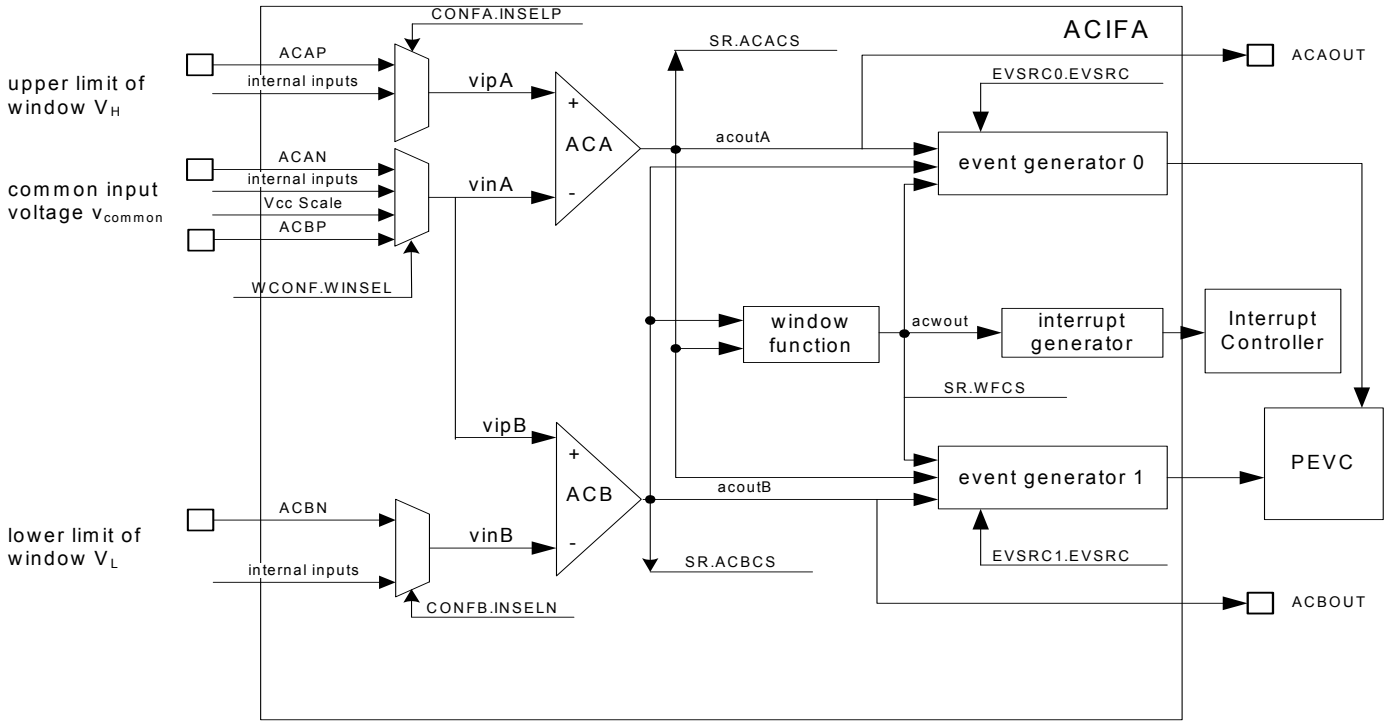


Figure 35-2. Analog Comparator Interface in window mode



35.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

35.4.1 Power Management

When the ACIFA is enabled it will remain clocked as long as its selected clock source is running. It can also wake the CPU from the currently active sleep mode. Refer to the Power Manager chapter for details on the different sleep modes.

35.4.2 Clocks

The clock for the ACIFA bus interface (CLK_ACIFA) is generated by the Power Manager. this clock is turned on by default, and can be enabled and disabled in the Power Manager.

35.4.3 Interrupts

The ACIFA interrupt request lines are connected to the interrupt controller. Using the ACIFA interrupts requires the Interrupt Controller to be programmed first.

35.4.4 Peripheral Events

The ACIFA peripheral events are connected via the Peripheral Event System. Refer to the Peripheral Event System chapter for details.

35.4.5 Debug Operation

The ACIFA is frozen during debug operation, unless the Run In Debug bit in the Development Control Register is set and the bit corresponding to the ACIFA is set in the Peripheral Debug Register (PDBG). Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual, and the OCD Module Configuration section, for details.

35.5 Functional Description

35.5.1 Normal mode

In normal mode, both analog comparators are independent.

35.5.1.1 ACIFA Output

An analog comparator generates one output acout[i] (with i = a or b) according to the input voltages vip_i (AC positive input) and vin_i (AC negative input):

- acout[i] = 1 if $vip_i > vin_i$
- acout[i] = 0 if $vip_i < vin_i$
- acout[i] = 0 if the AC output is not available (ie. The AC Ready bit in the Status Register (SR.ACRDYi) is still zero)

The ACIFA generates two independent events according to the configuration of the Event Source Selection field in the Event Configuration register (EVSRC0.EVSRC and EVSRC1.EVSRC):

- as soon as vipA > vinA or
- as soon as vipA < vinA or
- as soon as vipB > vinB or

- as soon as vipB < vinB or
- on toggle of the ACA output (acoutA) or
- on toggle of the ACB output (acoutB)

35.5.1.2 ACIFA Interrupt

Each AC has one source of interrupt. The configuration of the source of the interruption is done by writing in the Interrupt Settings field in the Configuration Register (CONF0.IS and CONF1.IS). The interrupt can be triggered:

- as soon as vip > vin
- as soon as vip < vin
- on toggle of the AC output (acout[i])

35.5.2 Window Mode

In window mode, the two ACs are grouped. The negative input of ACA and the positive input of ACB are the same and are defined in the Window Common Input Selection field in the Window Configuration register (WCONF.WINSEL). The positive input of ACA and the negative input of ACB are still configured by CONF0.INSELP and CONF1.INSELN.

35.5.2.1 ACIFA Output

Like in normal mode, the ACs generate the acout[i] outputs according to the input voltages vip_i (AC positive input) and vin_i (AC negative input):

- acout[i] = 1 if $vip_i > vin_i$
- acout[i] = 0 if $vip_i < vin_i$
- acout[i] = 0 if the AC output is not available (ie. SR.ACRDYi is still 0)

The ACIFA generates a window function signal (acwout) according to the common input voltage to be compared:

- acwout = 1 if the common input voltage is inside the window, $vin_B < v_{common} < vip_A$
- acwout = 0 if the common input voltage is outside the window, $v_{common} < vin_1$ or $v_{common} > vip_0$
- acwout = 0 if the window mode output is not available (ie. The Window Function Ready bit in the Status Register (SR.WFRDY) is still 0)

The ACIFA generates two independent events (like in normal mode) according to the configuration of EVSRC0 and EVSRC1:

- as soon as vipA > vinA or
- as soon as vipA < vinA or
- as soon as vipB > vinB or
- as soon as vipB < vinB or
- as soon as vinB < vcommon < VipA or
- as soon as vcommon < vinB or vcommon > vipA or
- on toggle of the ACA output (acoutA) or
- on toggle of the ACB output (acoutB) or
- on toggle of the window compare output (acwout)

35.5.2.2 *ACIFA Interrupt*

Like in normal mode, each AC has one source of interrupt. The configuration of the source of interruption is set in CONF0.IS and CONF1.IS. The interrupt can be triggered:

- as soon as vip > vin
- as soon as vip < vin
- on toggle of the AC output (acout[i])

An additional source of interrupt can be generated in window mode. Its configuration is set in the Window Interrupt Settings field in the Window Configuration register (WCONF.WIS). The source of interrupt can be triggered:

- as soon as the common input voltage is inside the window
- as soon as the common input voltage is outside the window
- on toggle of the window compare output (acwout)

35.5.3 Input Channels

Each Analog Comparator has one positive and one negative input. Each input may be chosen among one external input pin in addition to some internal signals. The user writes the input selection:

- in normal mode by writing in the Positive Input Selection field (CONFi.INSELP) and in the Negative Input Selection field in the Configuration register (CONFi.INSELN)
- in window mode by writing in the CONF0.INSELP, WCONF.WINSEL and CONF1.INSELN fields. In this case the WCONF.WINSEL field overrides the CONF0.INSELN and CONF1.INSELP fields.

35.5.4 Internal Inputs

Three internal inputs are available for the Analog Comparator.

35.5.5 High-speed vs Low Power Modes

It is possible to enable High-speed mode by writing a one to the Speed Selection bit in the CONF register (CONFi.SS) to get the shortest possible propagation delay. This mode consumes more power than the default low power mode (when CONFi.SS is written to zero) that has a longer propagation delay.

35.5.6 Input Hysteresis

The user can select between no, low, and high hysteresis, by writing in the Hysteresis Selection field in the CONF register (CONFi.HS). Adding hysteresis can avoid constant toggling of the compare output if the input signals are very close to each other.

35.5.7 Startup Time

After enabling an Analog Comparator, the comparison is available after a start-up time defined in the Start Up Time field in the Start Up Time register (SUT.SUT). During this time the AC output is not available. The status bit SR.ACRDYi gives the information that the ACi has its output available or not. In window mode the window mode output is available (SR.WFRDY is one) if both comparator outputs are available (SR.ACARDY and SR.ACBRDY are both one).

When the start-up time is finished, the comparison can be disabled/enabled, by writing a one to the corresponding AC Comparison Enable bit in the Enable register (EN.ACCPEN), without waiting anymore.

35.5.8 Starting Signal Compare

In order to start a voltage comparison, the Analog Comparator must be configured with the preferred properties and the inputs to be used. After enabling the Analog Comparator, it should wait for the startup time. When the startup time is over, the result of the comparison is available. It can be read at all times in the SR register.

35.6 User Interface

Table 35-1. ACIFA Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	ACA Configuration Register	CONFA	Read/Write	0x80000000
0x04	ACB Configuration Register	CONFB	Read/Write	0x80000000
0x08	AC Window Function Configuration Register	WCONF	Read/Write	0x00000000
0x0C	AC Event 0 source Configuration	EVSRC0	Read/Write	0x00000000
0x10	AC Event 1 source Configuration	EVSRC1	Read/Write	0x00000000
0x14	ACA Scale factor selection	SCFA	Read/Write	0x00000000
0x18	ACB Scale factor selection	SCFB	Read/Write	0x00000000
0x1C	AC Enable Register	EN	Write-only	0x00000000
0x20	AC Disable Register	DIS	Write-only	0x00000000
0x24	AC Startup Timer Register	SUT	Read/Write	0x00000000
0x28	AC Interrupt Enable Register	IER	Write-Only	0x00000000
0x2C	AC Interrupt Disable Register	IDR	Write-Only	0x00000000
0x30	AC Interrupt Mask Register	IMR	Read-Only	0x00000000
0x34	AC Event Enable Register	EVE	Write-Only	0x00000000
0x38	AC Event Disable Register	EVD	Write-Only	0x00000000
0x3C	AC Event Mask Register	EVM	Read-Only	0x00000000
0x40	AC Status Register	SR	Read-Only	0x00000000
0x44	AC Status Clear Register	SCR	Write-only	0x00000000
0x48	Version Register	VERSION	Read-Only	-(1)

Notes: 1. The reset values for this register is device specific. Please refer to the Module Configuration section at the end of this chapter.

35.6.1 ACA Configuration Register

Name: CONFA

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x80000000

31	30	29	28	27	26	25	24
OEN	-	-	SS	-	-	HS	

23	22	21	20	19	18	17	16
-	-	-	-	INSELN			

15	14	13	12	11	10	9	8
-	-	-	-	INSELP			

7	6	5	4	3	2	1	0
-	-	-	-	-	-	IS	

- **OEN: Output Enable**
  "0": Enable the redirection of the output of AC on pad
  "1": The output redirection is disabled
- **SS: Speed select**
  0: The low power mode is selected (longest propagation delay, lowest current consumption)
  1: The high-speed mode is selected (shortest propagation delay, highest current consumption)
- **HS: Hysteresis select**
  "00" or "11": no hysteresis
  "01": small hysteresis, 20 mV
  "10": high hysteresis, 50 mv
- **INSELN: Negative input select**
  i: select the i^{th} input of the mux
- **INSELP: Positive input select**
  i: select the i^{th} input of the mux
- **IS: Interrupt settings**
  "00": The comparator interrupt is set as soon as vip > vin
  "01": The comparator interrupt is set as soon as vip < vin
  "10": The comparator interrupt is set on toggle of analog comparator output

35.6.2 ACB Configuration Register

Name: CONFB

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x80000000

31	30	29	28	27	26	25	24
OEN	-	-	SS	-	-	HS	

23	22	21	20	19	18	17	16
-	-	-	-	INSELN			

15	14	13	12	11	10	9	8
-	-	-	-	INSELP			

7	6	5	4	3	2	1	0
-	-	-	-	-	-	IS	

- **OEN: Output Enable**
  "0": Enable the redirection of the output of AC on pad
  "1": The output redirection is disabled
- **SS: Speed select**
  0: The low power mode is selected (longest propagation delay, lowest current consumption)
  1: The high-speed mode is selected (shortest propagation delay, highest current consumption)
- **HS: Hysteresis select**
  "00" or "11": no hysteresis
  "01": small hysteresis, 20 mV
  "10": high hysteresis, 50 mv
- **INSELN: Negative input select**
  i: select the i^{th} input of the mux
- **INSELP: Positive input select**
  i: select the i^{th} input of the mux
- **IS: Interrupt settings**
  "00": The comparator interrupt is set as soon as vip > vin
  "01": The comparator interrupt is set as soon as vip < vin
  "10": The comparator interrupt is set on toggle of analog comparator output

35.6.3 AC Window Function Configuration Register

Name: WCONF

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	WINSEL			

7	6	5	4	3	2	1	0
-	-	-	-	-	-	WIS	

- **WINSEL: Window function common input select**
  i: select the i^{th} input of the mux
- **WIS: Window mode Interrupt settings**
  “00”: The window interrupt is set as soon as the input voltage is inside the window
  “01”: The window interrupt is set as soon as the input voltage is outside the window
  “10”: The window interrupt is set on toggle of window compare output

35.6.4 AC Event 0/1 Configuration Register

Name: EVSRC0-EVSRC1

Access Type: Read/Write

Offset: 0x0C-0X10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	EVSRC			

- **EVSRC: Event source selection**
  - “0000”: The event is set on ACA.acout rising edge
  - “0001”: The event is set on ACA.acout falling edge
  - “0010”: The event is set on ACA.acout rising or falling edge
  - “0011”: The event is set on ACB.acout rising edge
  - “0100”: The event is set on ACB.acout falling edge
  - “0101”: The event is set on ACB.acout rising or falling edge
  - “0110”: The event is set on acwout rising edge
  - “0111”: The event is set on acwout falling edge
  - “1000”: The event is set on acwout rising or falling edge
  - “1001”,”1010”,”1011”,”1100”,”1101”,”1110”,”1111”: no effect

35.6.5 ACA/B Scale Factor Selection Register

Name: SCFA-SCFB

Access Type: Read/Write

Offset: 0x14-0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	EN	SCF					

- **SCF: Scale Factor selection for Supply divider**
  VCC Scale = (64 -SCF) * VDDANA / 65
- **EN : Supply divider enable**
  0: The supply divider is disabled
  1: The supply divider is enabled

35.6.6 AC Enable Register

Name: EN

Access Type: Write-Only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	ACBCPEN	ACACPEN	WFEN	ACBEN	ACAEN

- **ACBCPEN: ACB Comparison enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will enable the comparison of the Analog Comparator B
- **ACACPEN: ACA Comparison enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will enable the comparison of the Analog Comparator A
- **WFEN: Window function enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will enable the window function. Enabling the window function automatically enable both comparators if they are not already enabled, and also the two comparison (SR.ACAEN, SR.ACBEN, SR.ACACPEN,and SR.ACBCPEN are set)
- **ACBEN: ACB enable**
  Writing a zero to this bit will has no effect
  Writing a one to this bit will enable the Analog Comparator B
- **ACAEN: ACA enable**
  Writing a zero to this bit will has no effect
  Writing a one to this bit will enable the Analog Comparator A

35.6.7 AC Disable Register

Name: DIS

Access Type: Write-Only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	ACBCPDIS	ACACPDIS	WFDIS	ACBDIS	ACADIS

- **ACBCPEN: ACB Comparison enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will disable the comparison of the Analog Comparator B
- **ACACPEN: ACA Comparison enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will disable the comparison of the Analog Comparator A
- **WFEN: Window function enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will disable the window function. Disabling the window function automatically disable both comparison (ACACPEN and ACBCPEN are cleared)
- **ACBEN: ACB enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will disable the Analog Comparator B
- **ACAEN: ACA enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will disable the Analog Comparator A

35.6.8 AC Startup Time Register

Name: SUT

Access Type: Read/Write

Offset: 0x24

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	SUT	

7	6	5	4	3	2	1	0
SUT							

- **SUT: Startup Time**

  Analog comparator startup time = 1/freq(ACIFA) x SUT

  Each time, an AC is enabled, the AC comparison will be enabled after the startup time due to the startup time of the AC

35.6.9 AC Interrupt Enable Register

Name: IER

Access Type: Write-Only

Offset: 0x28

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	SUTBINT	SUTAINT	WFINT	ACBINT	ACAINT

- **SUTBINT: ACB startup time interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will enable the ACB startup interrupt
- **SUTAINT: ACA startup time interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will enable the ACA startup interrupt
- **WFINT: Window function interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will enable the window function interrupt defined in the WCONF.WIS field
- **ACBINT: ACB interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will enable the ACB interrupt defined in the CONFB.IS field
- **ACAINT: ACA interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will enable the ACA interrupt defined in the CONFA.IS field

35.6.10 AC Interrupt Disable Register

Name: IDR

Access Type: Write-Only

Offset: 0x2C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	SUTBINT	SUTAINT	WFINT	ACBINT	ACAINT

- **SUTBINT: ACB startup time interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will disable the ACB startup interrupt
- **SUTAINT: ACA startup time interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will disable the ACA startup interrupt
- **WFINT: Window function interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will disable the window function interrupt defined in the WCONF.WIS field
- **ACBINT: ACB interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will disable the ACB interrupt defined in the CONFB.IS field
- **ACAINT: ACA interrupt enable**
  writing a zero to this bit has no effect
  Writing a one to this bit will disable the ACA interrupt defined in the CONFA.IS field

35.6.11 AC Interrupt Mask Register

Name: IMR

Access Type: Read-Only

Offset: 0x30

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	SUTBINT	SUTAINT	WFINT	ACBNT	ACAINT

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

35.6.12 AC Event Enable Register

Name: EVE

Access Type: Write-Only

Offset: 0x34

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ACEV1	ACEV0

- **ACEV1: Event 1 enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will enable the event zero defined in the EVSRC1.EVSCR field
- **ACEV0: Event 0 enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will enable the event one defined in the EVSRC0.EVSCR field

35.6.13 AC Event Disable Register

Name: EVD

Access Type: Write-Only

Offset: 0x38

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ACEV1	ACEV0

- **ACEV1: Event 1 enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will disable the event one defined in the EVSRC1.EVSCR field
- **ACEV0: Event 0 enable**
  Writing a zero to this bit has no effect
  Writing a one to this bit will disable the event one defined in the EVSRC0.EVSCR field

35.6.14 AC Event Mask Register

Name: EVM

Access Type: Read-Only

Offset: 0x3C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ACEV1	ACEV0

0: The corresponding peripheral event is disabled
1: The corresponding peripheral event is enabled
Theses bits are cleared when the corresponding bit in EVD is written to zero
These bits are cleared when the corresponding bit in EVE is written to one

35.6.15 AC Status Register

Name: SR

Access Type: Read-Only

Offset: 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	WFRDY	ACBRDY	ACARDY
23	22	21	20	19	18	17	16
-	-	-	-	-	WFCS	ACBCS	ACACS
15	14	13	12	11	10	9	8
-	-	-	ACBCPEN	ACACPEN	WFEN	ACBEN	ACAEN
7	6	5	4	3	2	1	0
-	-	-	SUTBINT	SUTAINT	WFINT	ACBNT	ACAINT

- **WFRDY: Window function ready**
  this bit is cleared when the window function output (wout) is not ready (one or both ACA or ACB comparison outputs is not ready)
  This bit is set when the window function output (wout) is ready (both ACA and ACB comparison outputs are ready)
- **ACBRDY: ACB ready**
  This bit is cleared when the ACB output (acoutB) is not ready
  This bit is set when the ACB output (acoutB) is ready (ACB is enabled and its SUT is over)
- **ACARDY: ACA ready**
  This bit is cleared when the ACA output (acoutA) is not ready
  This bit is set when the ACA output (acoutA) is ready (ACA is enabled and its SUT is over)
- **WFCS: Window function current status**
  This bit is cleared when the common input voltage is currently outside the window
  This bit is set when the common input voltage is currently inside the window
- **ACBCS: ACB current status of comparison**
  This bit is cleared when v_{ipB} is currently lower than v_{inB}
  This bit is set when v_{ipB} is currently greater than v_{inB}
- **ACACS: ACA current status of comparison**
  This bit is cleared when v_{ipA} is currently lower than v_{inA}
  This bit is set when v_{ipA} is currently greater than v_{inA}
- **ACBCPEN: ACB Comparison enable**
  This bit is cleared when the ACB comparison is disabled
  This bit is set when the ACB comparison is enabled
- **ACACPEN: ACA Comparison enable**
  This bit is cleared when the ACA comparison is disabled
  This bit is set when the ACA comparison is enabled

- **WFEN: Window function enable**
  - This bit is cleared when the window function is disabled
  - This bit is set when the window function is enabled
- **ACBEN: ACB enable**
  - This bit is cleared when the ACB is disabled
  - This bit is set when the ACB is enabled
- **ACAEN: ACA enable**
  - This bit is cleared when the ACA is disabled
  - This bit is set when the ACA is enabled
- **SUTBINT: ACB startup time interrupt status**
  - This bit is cleared when the ACB interrupt is not pending
  - This bit is set when the ACB interrupt is pending (the ACB has reached its startup time (SUT), the ACB comparison is valid)
- **SUTAINT: ACA startup time interrupt status**
  - This bit is cleared when the ACA interrupt is not pending
  - This bit is set when the ACA interrupt is pending (the ACA has reached its startup time (SUT), the ACA comparison is valid)
- **WFINT: Window function interrupt status**
  - This bit is cleared when the interrupt is not pending
  - This bit is set when the interrupt is pending
- **ACBINT: ACB Interrupt Status**
  - This bit is cleared when the interrupt is not pending
  - This bit is set when the interrupt is pending
- **ACAINT: ACA Interrupt Status**
  - This bit is cleared when the interrupt is not pending
  - This bit is set when the interrupt is pending

35.6.16 AC Status Clear Register

Name: SCR

Access Type: Write-Only

Offset: 0x44

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	SUT1INT	SUT0INT	WFINT	ACBNT	ACAINT

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

35.6.17 Version Register

Name: VERSION

Access Type: Read-Only

Offset: 0x48

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

35.7 Module configuration

The specific configuration for each ACIFA instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks according to the table in the System Bus Clock Connections section.

Table 35-2. Module clock name

Module name	Clock name	Description
ACIFA0	CLK_ACIFA0	Peripheral Bus clock from the PBA clock domain
ACIFA1	CLK_ACIFA1	Peripheral Bus clock from the PBA clock domain

Table 35-3. Register Reset Values

Register	Reset Value
VERSION	0x00000100

The following table gives which ACIFA module controls the analog comparators.

Table 35-4. Analog comparators controlled by ACIFA

Module name	Analog comparator name
ACIFA0	AC0A and AC0B
ACIFA1	AC1A and AC1B

The inputs of the AC are configured through the CONFA and CONFB registers. The configuration allows to select pin or internal voltage from the DACs.

The following table defines the valid settings for the CONFA and CONFB registers for each ACIFA instance. This setting defines the mapping of the AC input voltage.

Table 35-5. v_{ip} of AC0A selection

CONFA[INSELP]	Name	Connection
0	AC0AP0	See Peripheral Multiplexing on I/O line chapter
1	AC0AP1	
2	DAC0_int	Internal output of the DAC0

Table 35-6. v_{in} of AC0A selection

CONFA[INSELN]	Name	Connection
0	AC0AN0	See Peripheral Multiplexing on I/O line chapter
1	AC0AN1	
2	AC0BP0	
3	AC0BP1	
4	V_{VDDANA} scaled	V_{VDDANA} scaled
5	DAC1_int	Internal output of the DAC1

Table 35-7. v_{ip} of AC0B selection

CONFB[INSELP]	Name	Connection
0	AC0AN0	See Peripheral Multiplexing on I/O line chapter
1	AC0AN1	
2	AC0BP0	
3	AC0BP1	
4	DAC0_int	Internal output of the DAC0
5	V_{VDDANA} scaled	V_{VDDANA} scaled

Table 35-8. v_{in} of AC0B selection

CONFB[INSELN]	Description	Connection
0	AC0BN0	See Peripheral Multiplexing on I/O line chapter
1	AC0BN1	
2	DAC1_int	Internal output of the DAC1

Table 35-9. v_{ip} of AC1A selection

CONFA[INSELP]	Description	Connection
0	AC1AP0	See Peripheral Multiplexing on I/O line chapter
1	AC1AP1	
2	DAC0_int	Internal output of the DAC0

Table 35-10. v_{in} of AC1A selection

CONFA[INSELN]	Description	Connection
0	AC1AN0	See Peripheral Multiplexing on I/O line chapter
1	AC1AN1	
2	AC1BP0	
3	AC1BP1	
4	V_{VDDANA} scaled	V_{VDDANA} scaled
5	DAC1_int	Internal output of the DAC1

Table 35-11. v_{ip} of AC1B selection

CONFB[INSELP]	Description	Connection
0	AC1AN0	See Peripheral Multiplexing on I/O line chapter
1	AC1AN1	
2	AC1BP0	
3	AC1BP1	
4	DAC0_int	Internal output of the DAC0
5	V_{VDDANA} scaled	V_{VDDANA} scaled

Table 35-12. v_{in} of AC1B selection

CONFB[INSELN]	Description	Connection
0	AC1BN0	See Peripheral Multiplexing on I/O line chapter
1	AC1BN1	
2	DAC1_int	Internal output of the DAC1

In window mode, the window common input is configured through WCONF register.

Table 35-13. common input voltage of AC0 selection

WCONF[WINSEL]	Description	Connection
0	AC0AN0	See Peripheral Multiplexing on I/O line chapter
1	AC0AN1	
2	AC0BP0	
3	AC0BP1	

Table 35-14. common input voltage of AC1 selection

WCONF[WINSEL]	Description	Connection
0	AC1AN0	See Peripheral Multiplexing on I/O line chapter
1	AC1AN1	
2	AC1BP0	
3	AC1BP1	

36. ADC Interface (ADCIFA)

Rev. 1.1.0.2

36.1 Features

- 8/10/12-bit ADC core with built-in dual sample and hold (S/H)
- 16 channels
- Up to 1.5 mega-samples per second conversion rate for 12 bits resolution
  - Conversion time near to 5.3µs (12 bits resolution at 1.5 Msps)
- Up to 2 mega-samples per second conversion for lower resolution
  - Conversion time near to 3.5µs (10 bits resolution at 2 Msps)
  - Conversion time near to 3µs (8 bits resolution at 2 Msps)
- Multiple reference sources
  - 1V internal voltage reference
  - 0.6 * VDDANA internal
  - Two external reference voltage
- Direct measures or sampled with sample-and-hold
- Sample-and-hold (S/H) acquisition time window has separate prescale control (gain: 1, 2, 4, 8, 16, 32, 64).
- Sequencer can be operated as two independent 8-state sequencers operating on its own S/H (dual sequencer mode) or as one large 16-state sequencer (single sequencer mode)
- 16 result registers
- Source selection for the start-of-conversion (SOC)
  - Software
  - Embedded timer
  - Peripheral Event Controller
  - Continuous
- Two sequencer modes:
  - Run the whole sequence on a start-of-conversion
  - Run a single conversion on a start-of-conversion
- Flexible interrupt control allows interrupt request on every end-of-sequence or on every single conversion.
- Windowing mechanism, with selectable channel
- Free running mode
- 2 PDCA channels (one per sequencer)
- Power reduction modes
- Programmable ADC timings

36.2 Overview

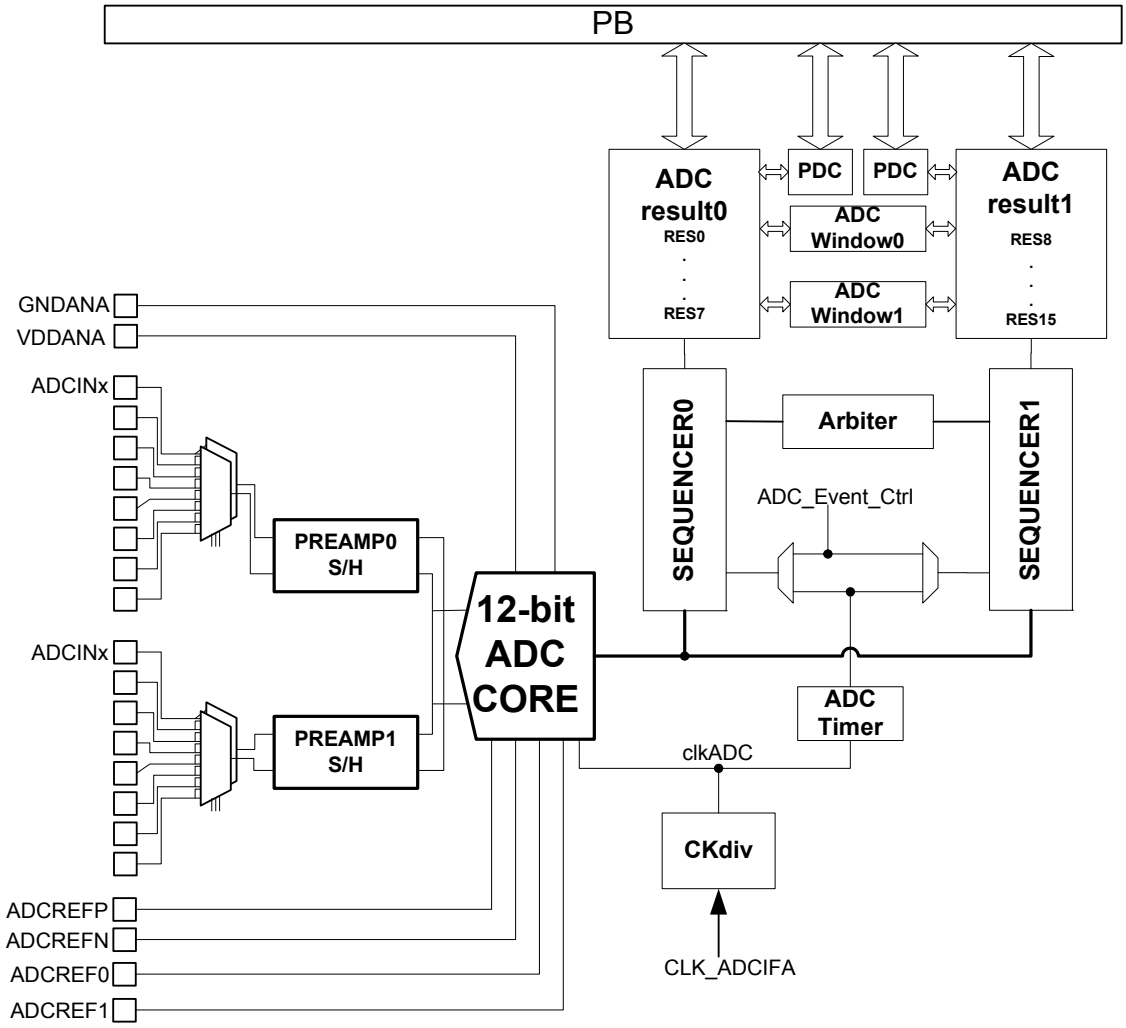
The Analog-to-Digital Converter (ADC) is fully differential and based on a 12-bit pipelined topology using switched capacitors circuitry. Two sample and hold (S/H) running simultaneously with 1, 2, 4, 8, 16, 32, 64 gain factors are feeding a single ADC analog block so that the system acts as if there were two conversion running in parallel. It can be configured as a 8-bit 10-bit or 12-bit ADC and is capable of converting 1.5 million samples per second thanks to its pipeline topology. 10-bit and 8-bit conversion resolution can be achieved at higher conversion rates. Note that results are always signed in 2's complement.

Note: The pipelined topology implies a latency between the sampling event and the update of the result register of: *(Resolution(SRES)/2 + 3 - SHD) · T(CkADC)*

The ADC has an internal defined conversion range of ±1.0V. An additional internal reference mode allows conversion range of ±0.6*VDDANA. In addition, the ADC may operate with external references for different conversion ranges.

36.3 Block Diagram

Figure 36-1. ADCIFA Block Diagram



36.4 I/O Lines Description

Table 36-1. I/O Lines Description

Name	Description
ADCINx	ADC analog input
ADCREFP	CFG.EXREF= 0: Normal operation, this pin is used to decouple ADC internal reference. ADCREFP should be connected to a 100nF external decoupling capacitor. CFG.EXREF= 1: Forcing reference using ADCREFP/ADCREFN differential pin pair voltage Please refer to the Section 36.6.10 for more information.
ADCREFN	CFG.EXREF= 0: Normal operation, this pin is used to decouple ADC internal reference. ADCREFN should be connected to a 100nF external decoupling capacitor. CFG.EXREF= 1: Forcing reference using ADCREFP/ADCREFN differential pin pair voltage Please refer to the Section 36.6.10 for more information.
ADCREF0	External reference input (with respect to analog ground) bypassed when CFG.RS is enabled
ADCREF1	External reference input (with respect to analog ground) bypassed when CFG.RS is enabled
VDDANA	Analog power supply
GNDANA	Analog ground

36.5 Product Dependencies

36.5.1 I/O Lines

The pins used for interfacing the ADCIFA may be multiplexed with the I/O Controller lines. The programmer must first program the I/O Controller to assign the desired ADCIFA pins to their peripheral function. If I/O lines of the ADCIFA are not used by the application, they can be used for other purposes by the I/O Controller.

Not all ADCIFA inputs may be enabled. If an application requires only four channels, then only four ADCIFA lines need to be assigned to ADCIFA inputs.

36.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the ADCIFA, the ADCIFA will stop functioning and resume operation after the system wakes up from sleep mode. Before entering a sleep mode where the clock to the ADCIFA is stopped, make sure the Analog-to-Digital Converter cell is put in an inactive state. Refer to Section 36.6.3 for more information.

36.5.3 Clocks

The clock for the ADCIFA bus interface (CLK_ADICFA) is generated by the Power Manager. This clock is turned on by default, and can be enabled and disabled in the Power Manager. It is recommended to disable the ADCIFA before disabling the clock, to avoid freezing the ADCIFA in an undefined state.

36.5.4 Interrupts

The ADCIFA interrupt line is connected to one of the internal sources of the Interrupt Controller (INTC). Using the ADCIFA requires the INTC to be configured first.

36.5.5 Event System

The event controller provides the ADCIFA two trigger sources.

36.5.6 Debug Operation

When an external debugger forces the CPU into debug mode:

- the ADCIFA continues normal operation if the bit related to ADCIFA in PDBG register is ‘0’. PDCA access continues normal operation and may interfere with debug operation.
- the ADCIFA is frozen if the bit related to ADCIFA in PDBG register is ‘1’. When the ADCIFA is frozen, ADCIFA PB registers can still be accessed. Then, reading registers may modify status bits (OVRx, LOVRx) like in normal operation. PDCA access are pending.

36.6 Functional Description

36.6.1 ADC Resolution

The ADC supports 8-bit, 10-bit or 12 bits resolutions. Precision can be set differently for each sequencer by setting the SRES bits in the SEQCFGx register. By default, after a reset, the resolution is set to 12 bits. To get full resolution, the user should first calibrate the ADC as detailed in Section 36.6.16.

36.6.2 ADC Conversion Modes

36.6.2.1 Differential / single ended

The ADC is fully differential. To perform single ended measures, the user can perform pseudo unipolar conversions by connecting ground onto the negative input. User can connect it to an external ground through pads or internal ground depending on if there's one connected onto the negative input multiplexer. Since conversion results are always 12 bits in 2's complement representation, the sign bit will not change, and then the resulting resolution is 11 bits max.

36.6.2.2 S/H versus DIRECT conversions

By default S/H are enabled, to change that setting, set the Sample and Hold disable bit (SHD) located in the CFG register. Maximum accuracy is achieved when disabling S/H but setting this bit forbids dual sequencer mode, Sequencer 1 is then switched off. Furthermore, in this mode S/H are switched off to lower power consumption.

Table 36-2. S/H versus DIRECT Conversions

Mode	Characteristics	
S/H	Pros	Gain setting (1, 2, 4, 8, 16, 32, 64) Dual sequencer mode
	Cons	Reduced accuracy Dynamic limitation (fixed with over-sampling) 1 ADC clock period spent to propagate into S/H
DIRECT	Pros	No dynamic limitation due to S/H Full accuracy Saves 1 ADC clock period compared to the features list timings
	Cons	No gain Single sequencer mode only

36.6.3 Power Reduction Modes

Configuration bits acting on the power consumption of the digital and analog blocks are ADC enable **(**ADCEN) and Sleep Mode Selection (SLEEP) bits located in the CFG register:

Table 36-3. Power Reduction Mode over the ADCEN Setting

ADCEN	Behavior
0	Digital controller dynamic activity is stopped (gated clocks) All analog is powered off (reference sources, ADC, sample & hold)
1	Digital controller enabled Analog references are switched on The ADC block is powered on depending on the SLEEP bit

Table 36-4. Power Reduction Mode

SLEEP	Behavior
0	Analog ADC block always powered on
1	Analog ADC block powered off after each conversion

Depending on the Start Of Conversion Behavior (SOCB) bit in the Sequencer Configuration (SEQCFGx) register, the HOT start-up sequence will be performed before each conversion or before each new conversion sequence. The ADC analog block is powered off when not used, it needs 24 ADC clock cycles to wake-up. If start of conversion frequency is lower than 1/25.f(CkADC) then no conversion will be lost.

36.6.4 ADC Sequencer Operating Modes

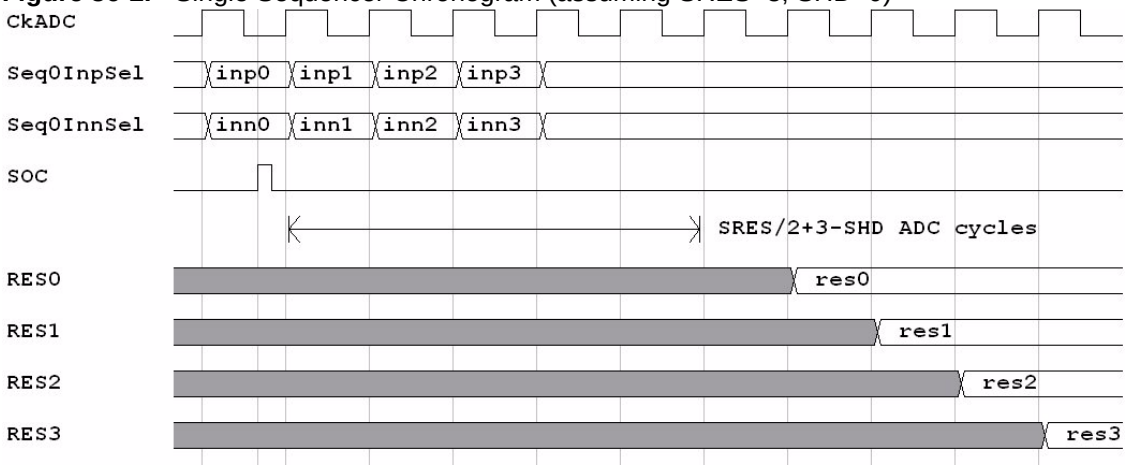
36.6.4.1 General

The ADC sequencer consists in two independent 8-state sequencers (SEQ0 and SEQ1) that can also be cascaded together to form one 16-state sequencer (SEQ). The word “state” represents the number of auto-conversions that can be performed with the sequencer. In both cases, the ADC has the ability to auto-sequence a series of conversions. This means that each time a sequencer receives a start-of-conversion request, it can perform multiple conversions automatically. For every sequencer conversion in dual-sequencer mode, any one of the available sequencer 16 input channels can be selected through the analog MUX. In the same way, in single-sequencer mode, any of the SEQ0 input channels can be selected. After conversion, the digital value of the selected channel is stored in the appropriate result register (RESn). It is also possible to sample the same channel multiple times, allowing the user to perform “over-sampling”, which gives increased resolution over traditional single-sampled conversion results.

36.6.4.2 Single-sequencer mode (cascaded mode)

By setting the Single Sequencer Mode (SSMQ) bit in the CFG register, the two sequencers are cascaded allowing a maximum of 16 successive measures among the SEQ0 16 analog inputs. Figure 36-2 shows a sequence of 4 differential measures, initiated by the Start Of Conversion (SOC) request. The sequence of analog inputs to be measured is determined by the values of (INPSEL0x, INNSEL0x) and (INPSEL1x, INNSEL1x) couples of registers. Each analog input is selected by the analog multiplexer then sampled one by one every ADC clock cycle. In addition, the conversion lasts (SRES / 2 + 3 - SHD) ADC clock cycles due to the ADC pipelined topology.

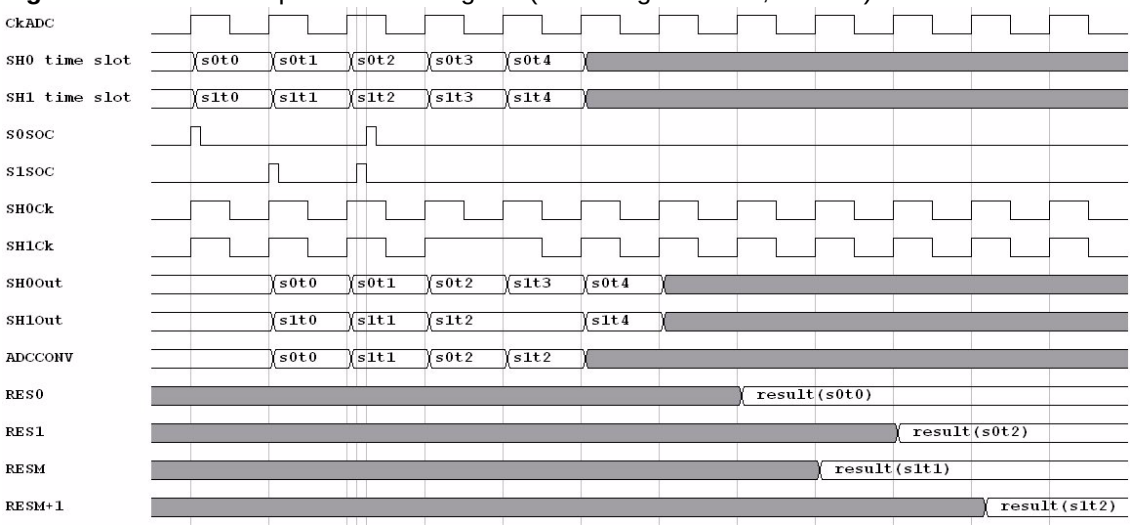
Figure 36-2. Single Sequencer Chronogram (assuming SRES=8, SHD=0)



36.6.4.3 *Dual-sequencer mode (simultaneous sampling)*

The ADC has the ability to sample two pairs of ADCINx inputs simultaneously (see Figure 36-3), provided that one pair is from the inputs available on the sequencer 0 and the other is from the inputs available on the sequencer 1 (see Figure 36-1). To put the ADC into simultaneous sampling mode, the SSMQ bit needs to be set in the CFG register.

Figure 36-3. Dual Sequencer Chronogram (assuming SRES=8, SHD=0)



In this chronogram, ADCCONV signal represents the value being sampled by the ADC

36.6.4.4 *Sequencer behavior on a Start Of Conversion*

Thanks to the SOCB bit in the SEQCFGx register, 2 different sequencer behaviors are possible:

Table 36-5. SOCB Behavior

SOCB	Comment
0	All sequence conversions are performed on a SOC event.
1	A single conversion belonging to the sequence is performed on a SOC event.

36.6.4.5 *Sequencer start/stop mode*

Thanks to the Software Acknowledge bit (SA) in the SEQCFGx register, the behavior of sequencer x at the end of a sequence can be configured.

Table 36-6. Sequencer Start/Stop Mode

SA	Comment
0	The sequencer waits for software acknowledge. Acknowledge is done by writing a 1 in the SEOSx bit of the SCR register.
1	The sequencer will restart automatically a new sequence on a new SOC. Results will be overwritten if not processed.

The Sequencer x Overrun Error bit (OVRx) in SR register indicates that an overrun error occurred in the sequencer x. This means that the RES0 register has not been read while a new sequence is starting. Events such as end-of-sequence or end-of-conversion can be caught by interrupt servicing or polling routines thanks to the SEOSx and SEOCx bits in the SR register.

36.6.4.6 *Sequencer free running-mode*

Only SEQ0 has the free-running mode capability. In free-running mode the ADC continuously converts analog values configured in the sequencer. In this mode, the sequence restarts automatically after each end of sequence without waiting for the last conversion to finish. This mode is configured by setting the Free Running Mode (FRM) bit in the CFG register. The conversion sequence will start on the first SOC defined by the Trigger Selection (TRGSEL) field in the SEQCFG0 register. In this mode only SEQ0 is running once triggered.

When converting at full speed the sequencer always wait for the last conversion to be finished to rise the sequencer end of sequence status bit (EOS). Figure 36-4 shows a 3 conversions sequence running. When the third channel is sampled the sequencer has to wait for the pipeline to be flushed. This takes SRES/2+3-SHD clock cycles. To avoid this you can run that sequence in free running mode. Please refer to Figure 36-5. The sequencer will run the sequence without waiting for the pipeline to be flushed but the user will have to read the converted value before it is overwritten by a new conversion.

Figure 36-4. Not Using FRM and Converting at Full Speed

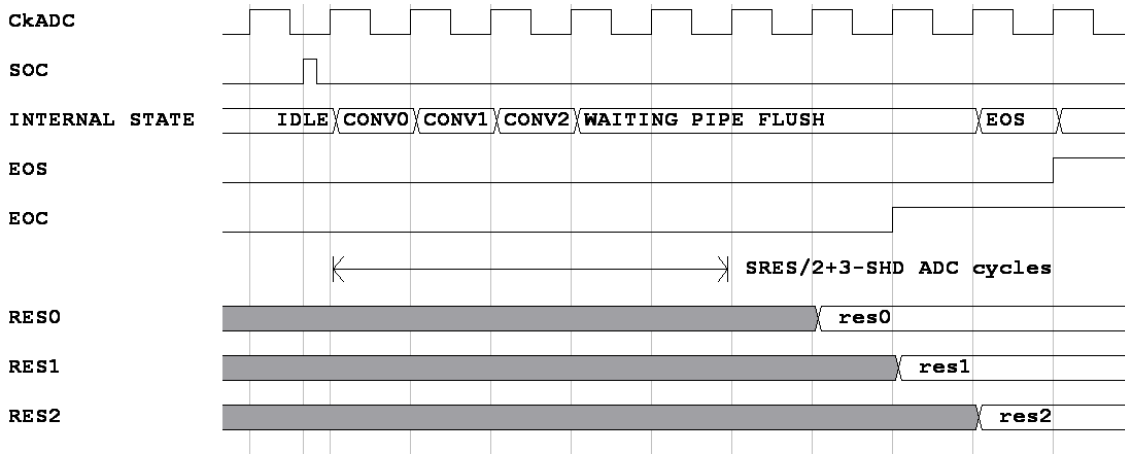
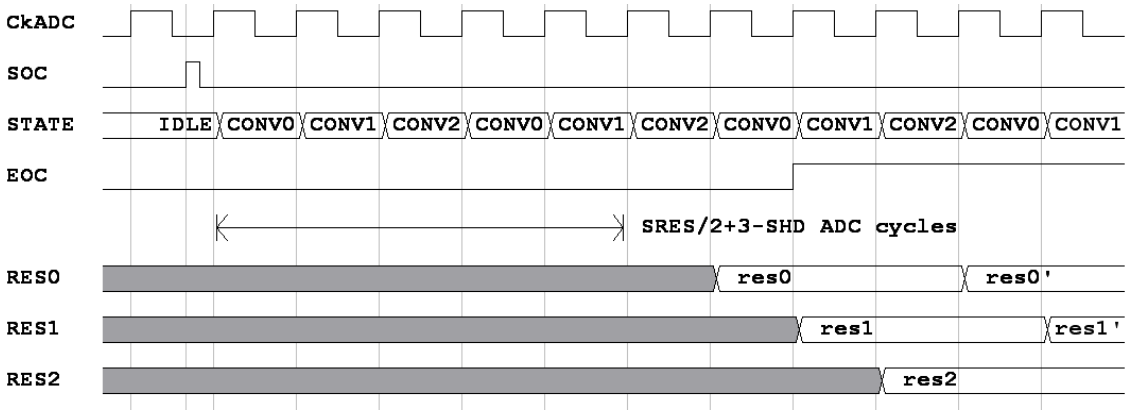


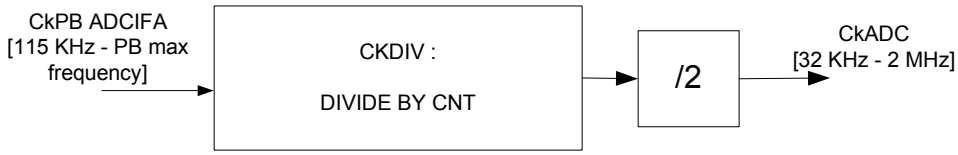
Figure 36-5. Using FRM and Converting at Full Speed



36.6.5 ADC Clock Configuration (CKDIV)

The clock frequency range for the ADC is [1.5 MHz - 32 KHz]. Since the ADC interface uses the system clock up to the PB maximum frequency, a clock downscale must be done if a higher frequency system clock is used. This scaling may also be done in order to slow down the ADC conversions or increase the S/H time, without affecting the system clock. The downscale is done by writing the maximum counter value in the Counter Value (CNT) field of the CKDIV register, with a possible division factor from 1 to 512 giving the following transfer function: T(CkADC)= T(CkPB) · ((CNT + 1) · 2). The divider is enabled as soon as the ADC is enabled by setting the ADCEN bit in the CFG register.

Figure 36-6. Clock Generator Block Diagram



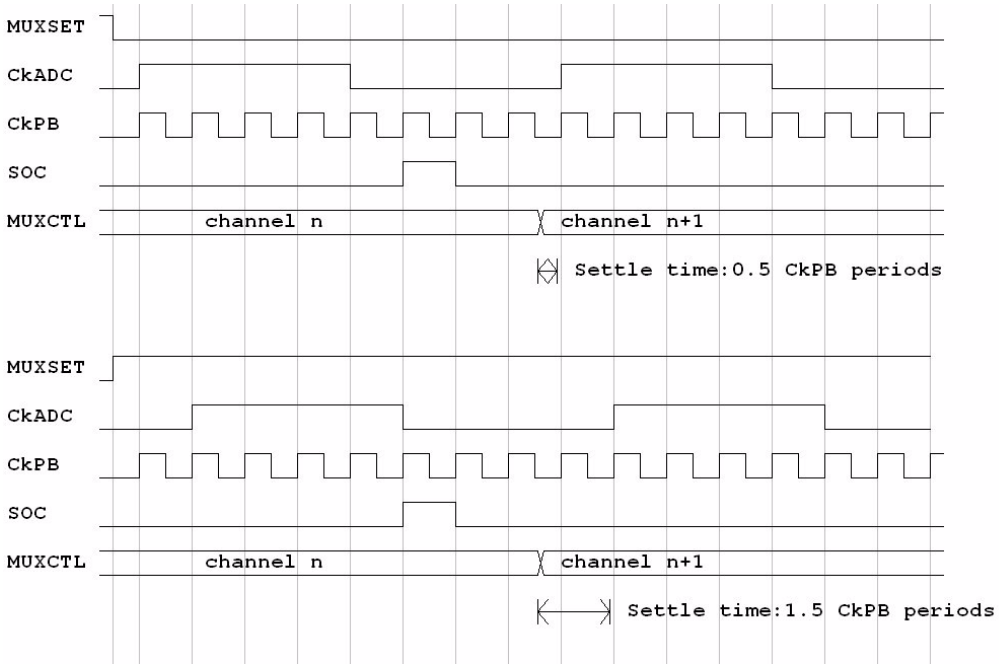
In addition when CKDIV is written, the internal counter is reset to avoid rollover phenomena: DO NOT WRITE CKDIV WHEN PERFORMING CONVERSIONS.

F(CkPB) should at least be 4 times greater than F(CkADC) to make the ADC controller work properly.

36.6.6 ADC Multiplexers Settle Time

By default, channel multiplexers settle time is set to half a PB clock period. If operating with a high PB clock frequency, then MUX settle time can be increased to achieve a 1.5 PB clock periods settle time by writing a one in the MUXSET field in the CFG register. For more information, please refer to the ADC electrical characteristics.

Figure 36-7. Multiplexers Settle Time Depending on the CFG.MUXSET Configuration Bit

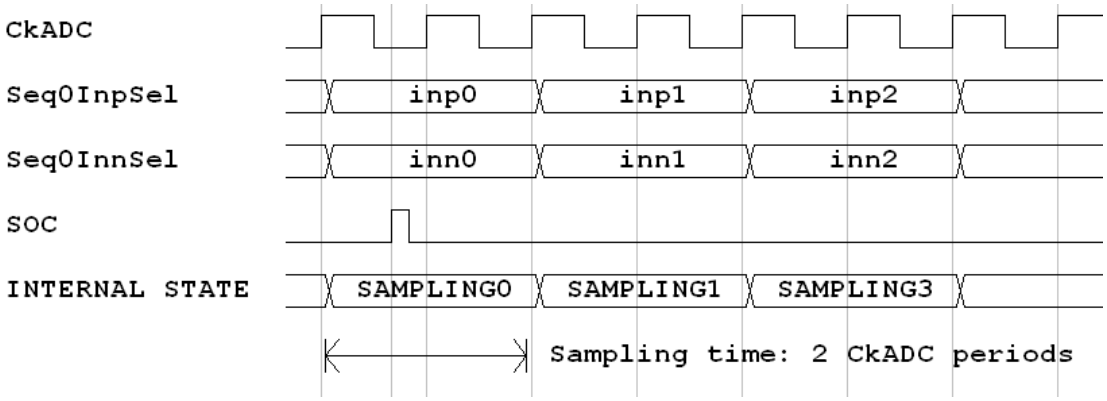


The chronogram above shows that for the same start of conversion (SOC) event, CkADC rises one PB clock period later. The ADC and S/H are sampling when CkADC is high, so setting the CFG.MUXSET bit will delay the sampling phase by one PB clock period.

36.6.7 Oversampling Mode

To improve conversion accuracy, it is recommended to perform oversampling. This is particularly useful for high impedance sources. This mode can be used whether the ADC is used in direct mode or not. Please note that it behaves as if a sequence of 2 consecutive conversions had been programmed with the first conversion ignored. The consequence is a conversion rate divided by 2. Also note that this mode cannot be used in conjonction with the Dynamic mode. If dynamic and oversampling modes are both enabled, dynamic mode will be applied.

Figure 36-8. Oversampling



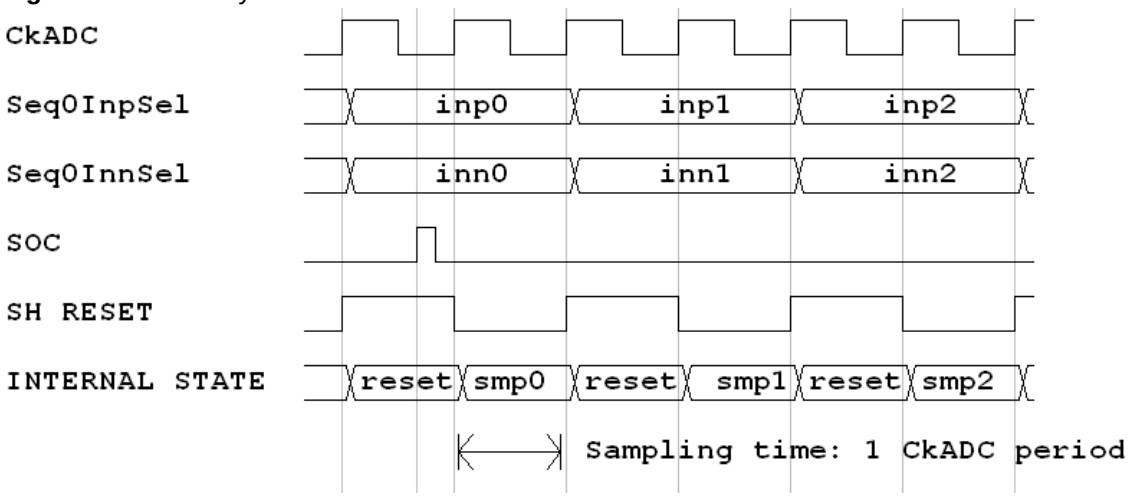
36.6.8 Sample & Hold (S/H) with Gain

The ADC preamplifiers are made of two cascaded switched-capacitors amplifiers stages. They are used to sample analog voltages and provide it to the ADC block when it has a time slot to make the conversion. It also amplifies the input voltage.

36.6.8.1 *Dynamic mode*

Dynamic mode aims at improving conversion accuracy when performing channel sweeping or measures on high frequency input signals. It is then recommended using the SHDYN (sample and hold dynamic mode) bit control in the SEQCFGx register. Doing this causes the insertion of a supplementary sampling cycle of one CkADC clock period used to reset the sample and hold. As a consequence, conversion rate is divided by two. Please note that it is useless performing oversampling when using that mode since the S/H are reseted before actually sampling.

Figure 36-9. SH Dynamic Mode



36.6.8.2 *Gain factor*

S/H allows the amplification of very small signals or buffering of very high impedance signal sources. The gain factor may be configured from 1x to 64x by writing to the Sequencer Conversion n Sample and Hold Gain (GCNVn) field of the SHGx register. The gain can be changed from sample to sample by writing the SHGx registers.

Table 36-7. Gain Factor

GCNVn			Gain
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	Reserved

36.6.9 Power-up and Startup Time

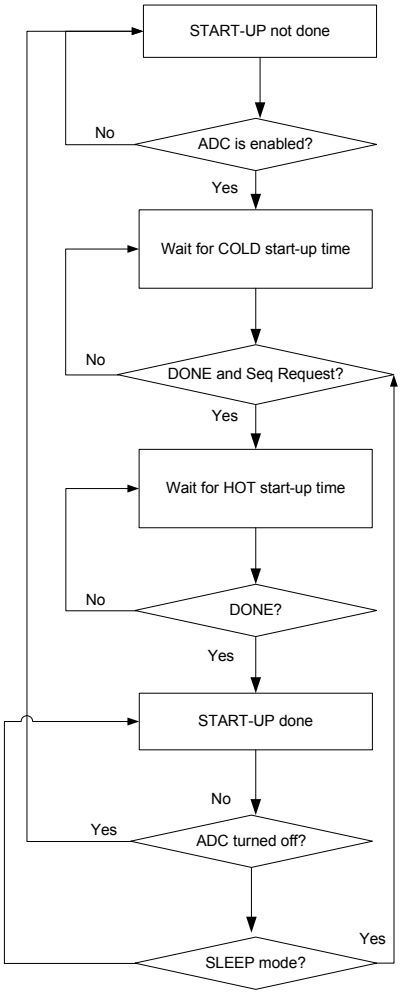
To convert correct values, both references and ADC have to be powered-up correctly, otherwise wrong values will be converted until the end of the start-up time.

- Cold start-up: References needs 1 ms max to establish.
- Hot start-up: Once references are up, 24 CkADC clock periods are needed.

When in sleep mode, the HOT start-up sequence is performed each time a conversion or a sequence is triggered thanks to the SOCB bit in the SEQCFGx register. Indeed, the ADC analog block is powered off while not used.

The end of the power-up sequence can be read from the Start-Up Time Done (SUTD) bit of the SR register. This bit is set by hardware at the end of the start-up sequence and cleared by software by writing a '1' in the SUTD bit of the SCR register. It is also cleared by hardware when the ADCIFA is turned off then on by clearing and setting the ADCEN bit of the CFG register.

Figure 36-10. Power-Up Sequence



36.6.10 Analog Reference

The following sources are available as analog reference (AREF) in the ADC. They are selected through the Reference Source (RS) field in the CFG register:

- 1V internal voltage reference
- 0.6*VDDANA internal voltage reference
- Two external reference voltage (ADCREF0 or ADCREF1 over chip analog ground)

When using an internal reference, it is recommended inserting a decoupling capacitor between ADCREFP and ADCREFN externally (mandatory to get the full 12-bits precision). This means that two pins will be dedicated to reference decoupling. If the pins are needed for other purposes, the decoupling may be skipped giving a conversion accuracy of 10 bits.

It is also possible to force a differential reference by setting the CFG.EXREF bit. This will bypass the CFG.RS selection setting and make the ADC use the differential ADCREFP/ADCREFN pin pair voltage as reference.

36.6.11 Conversion Range

The conversion amplitude range is given by the ADC acquisition mode and the reference source:

Table 36-8. Conversion Range vs. Reference

Reference	Conversion range
Internal reference 1	±1V
Internal reference 2	±0.6 * VDDANA
External reference 1	± min(3.5 V, VDDANA - 0.7)
External reference 2	± min(3.5 V, VDDANA - 0.7)

36.6.12 Conversion Results

If the Half Word Left Adjust (HWLA) bit in the SEQCFGx register is set, then the result will be left adjusted on the 16 lower bits of the RESn register. Otherwise, results will be right-adjusted.

ADC transfer function:

$$RESn = \frac{GAIN \times (V(ADCIN(p) - V(ADCIN(n))))}{(V(VREFP) - V(VREFN))} \times 2^{SRES + HWLA \times (16 - SRES)}$$

All conversion results are signed in two's complement representation. Extra bits depending on resolution and left adjust settings are padded with the sign bit. It means that if you read RESn registers as a 32 bits register, the result will be correct.

36.6.13 Start Of Conversion (SOC)

ADC sequencers conversions can be triggered for each sequencer with the following sources:

Table 36-9. Trigger of Start Of Conversion

Source Sequencer	Internal timer (SOCx)	Internal Timer	Event controller	Event controller re-synchronized	
SEQ0	*	*	*	*	
SEQ1	*	*	*	*	

The sources must be configured through the Trigger Selection (TRGSEL) field of the SEQCFGxregister. Selecting the event controller source allows any event controller source to generate SOC.

The ADC can serve a maximum of one SOC per ADC clock cycle. Extra SOC will be ignored and the Missed Start-Of-Conversion (MSOCx) bit in the SR register will be set. If the SOC frequency provided by the event controller exceeds the ADC capability, the event controller will generate an underrun status.

36.6.14 Internal Timer

The ADCIFA embeds an internal timer used as a trigger source for SEQ0, SEQ1 and TSSEQ which can be configured by setting the ITMC fields of the ITIMER register.

*Internal Timer Trigger Period= (ITMC+1)*T(CkADC)*

The 17 bits counter allows SOC period up to 174ms when CkADC clock frequency is set to 1.5 MHz.

Once set as a SOC source, the internal timer as to be started by writing a '1' in the Internal Timer Start (TSTART) bit of the CR register. It can be stopped in the same way by writing a '1' in the TSTOP bit of the CR register. The current status of the internal timer can be read from the Running timer status (RUN) field of the SR register: 0 means stopped, 1 means running. In addition when the internal timer is running, if ITIMER register is written to change its frequency, the internal counter is cleared to avoid rollover phenomena.

Note: It is possible to generate an internal timer event each CkADC time slot by writing 0x0 to the ITIMER register ITMC field and by selecting the internal timer as a SOC source.

36.6.15 Peripheral DMA

There are two Peripheral DMA Controller (PDC) channels corresponding to the maximum number of sequencers that can be run at the same time. The Sequencer x Last Converted Value (LCVx) register contains the last converted value of the sequencer x according to the conversion result format. The LCV register is updated each time the sequencer ends a conversion.
If the last converted value has not been read when a new one is available, the previous data is overwritten. This overrun status is signalled by the Sequencer x Last Converted Value Overrun (LOVRx) bit in the SR register indicating that at least one overrun error occurred concerning sequencer x.
The OVRx and LOVRx bits of the SR register are cleared by writing a ‘1’ respectively in the OVRx and LOVRx fields of the SCR register.

Note: PDC transfers are 16 bits wide.

36.6.16 Calibration

Accuracy of the conversion is based on calibration of switched capacitors and operational amplifiers offset cancellation. Gain correction is done by writing a calibration word into the ADCCAL and SHCAL registers since it is temperature and operating voltage independent.

36.6.16.1 ADC gain error calibration

The ADC is gain-calibrated during production, but to take advantage of this the calibration value must be read from the factory page in flash and written to the Gain Calibration (GCAL) field of the ADCCAL register.

36.6.16.2 *ADC offset error calibration*

Offset cancellation has to be performed by the user due to temperature and operating voltage conditions dependence. The offset can be obtained by converting a null differential value. That offset has to be negated and written into the Offset Calibration (OCAL) field of the ADCCAL register. Then, for each conversion result, the controller will return the converted value added with the signed OCAL value. For instance, if the offset value obtained is 0x3, then the value 0xFD must be written to OCAL. Please note that OCAL is a 6 bits register, if the MSB is high then the value will be considered negative. A saturation mechanism avoids flipping phenomena. OCAL stores a signed number of LSB assuming the calibration has been performed in 12 bits resolution. If converting at a lower resolution, correction will only take into account the appropriate most significant bits.

36.6.16.3 *Sample and hold gain error calibration*

S/H are gain-calibrated during production, but to take advantage of this the calibration value must be read from the factory page in flash and written to the Sample and Hold Gain Calibration (GAIN0 and GAIN1) fields of the SHCAL register.

36.6.17 Window Monitor

There are 2 window monitors that allow to compare two of the result registers to some predefined threshold values. The Window Mode (WM) field in WCFGy register (see Table 36-10) allows the user to configure operating mode in order to generate interrupts. The High Threshold (HT) and Low Threshold (LT) fields in WCFGy register give the threshold voltage values of the comparators. The result register to monitor is selected by the Source (SRC) field in WCFGy register.

Table 36-10. Window Modes

WM			Modes
0	0	0	No window mode (default)
0	0	1	Mode 1: active when result < HT
0	1	0	Mode 2: active when result > LT
0	1	1	Mode 3: active when LT < result < HT
1	0	0	Mode 4: active when result >= LT or result >= HT
1	0	1	reserved
1	1	0	reserved
1	1	1	reserved

Note: Comparisons are performed regardless with the HWLA setting (half word left adjust).

36.6.18 Arbitration

In dual sequencer mode, SEQ0 has priority over SEQ 1. Due to the ADC pipeline topology, the arbiter is implemented in order to allocate optimal time slots to each sequencer in order to pipe requests. When all analog voltages have been taken into account in the ADC pipeline, an other sequencer can drive the analog blocs without waiting for the end of the whole conversion process. The ADC result will be sampled by another process when getting the wanted precision.

36.6.19 Interrupts

Table 36-11. ADCIFA Interrupt Group

Line	Line Description	Related Status
0	Sequencer 0	Sequencer 0 end of sequence Sequencer 0 end of conversion Sequencer 0 overrun Sequencer 0 (last converted value) overrun Sequencer 0 missed start-of-conversion
1	Sequencer 1	Sequencer 1 end of sequence Sequencer 1 end of conversion Sequencer 1 overrun Sequencer 1 (last converted value) overrun Sequencer 1 missed start-of-conversion
2	Start-up done	Start-up done
3	Window	Window 0 Window 1

36.7 User Interface

Table 36-12. ADCIFA Register Memory Map

Offset	Register	Name	Access	Reset State
0x0000	CR register	CR	Write-only	0x00000000
0x0004	CFG register	CFG	Read/Write	0x00000000
0x0008	SR register	SR	Read-only	0x00000000
0x000C	SCR register	SCR	Write-only	0x00000000
0x0010	SSR register	SSR	Write-only	0x00000000
0x0014	SEQCFG0 register	SEQCFG0	Read/Write	0x00000000
0x0018	SEQCFG1 register	SEQCFG1	Read/Write	0x00000000
0x001C	SHG0 register	SHG0	Read/Write	0x00000000
0x0020	SHG1 register	SHG1	Read/Write	0x00000000
0x0024	INPSEL00 register	INPSEL00	Read/Write	0x00000000
0x0028	INPSEL01 register	INPSEL01	Read/Write	0x00000000
0x002C	INPSEL10 register	INPSEL10	Read/Write	0x00000000
0x0030	INPSEL11 register	INPSEL11	Read/Write	0x00000000
0x0034	INNSEL00 register	INNSEL00	Read/Write	0x00000000
0x0038	INNSEL01 register	INNSEL01	Read/Write	0x00000000
0x003C	INNSEL10 register	INNSEL10	Read/Write	0x00000000
0x0040	INNSEL11 register	INNSEL11	Read/Write	0x00000000
0x0044	CKDIV register	CKDIV	Read/Write	0x00000000
0x0048	ITIMER register	ITIMER	Read/Write	0x00000000
0x0058	WCFG0 register	WCFG0	Read/Write	0x00000000
0x005C	WCFG1 register	WCFG1	Read/Write	0x00000000
0x0060	LCV0 register	LCV0	Read-only	0x00000000
0x0064	LCV1 register	LCV1	Read-only	0x00000000
0x0068	ADCCAL register	ADCCAL	Read/Write	0x00000000
0x006C	SHCAL register	SHCAL	Read/Write	0x00000000
0x0070	IER register	IER	Write-only	0x00000000
0x0074	IDR register	IDR	Write-only	0x00000000
0x0078	IMR register	IMR	Read-only	0x00000000
0x007C	VERSION register	VERSION	Read-only	-(1)
0x0080	PARAMETER register	PARAMETER	Read-only	-(1)
0x0084	RES register	RES	Read-only	-

Note: 1. The reset values are device specific. Please refer to the Modue Configuration section at the end of this chapter.

36.7.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	TSTART	TSTOP	SOC1	SOC0

- **TSTART: Internal Timer Start Bit**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts the internal timer.
  This bit always reads as zero.
  The internal timer status can be read in the RUNT field of the SR register.

- **TSTOP: Internal Timer Stop Bit**
  Writing a zero to this bit has no effect.
  Writing a one to this bit stops the internal timer.
  This bit always reads as zero.
  The internal timer status can be read in the RUNT field of the SR register.

- **SOC1: Sequencer 1 Start of Conversion**
  Writing a zero to this bit has no effect.
  Writing a one to this bit makes the sequencer 1 to start a conversion.
  This bit always reads as zero.

- **SOC0: Sequencer 0 Start Of Conversion**
  Writing a zero to this bit has no effect.
  Writing a one to this bit makes the sequencer 0 to start a conversion.
  This bit always reads as zero.

36.7.2 Configuration Register

Name: CFG

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	SUT					

15	14	13	12	11	10	9	8
-	-	-	-	-	MUXSET	EXREF	-

7	6	5	4	3	2	1	0
SHD	RS		FRM	SSMQ	SLEEP	-	ADCEN

- **SUT: Start-up Time**
  Number of ADC clock cycles to wait for: (STARTUP + 1) * 32.

- **MUXSET: Mux Settle Time**
  1: The multiplexers settle time is set to 0.5 PB clock periods.
  0: The multiplexers settle time is set to 1.5 PB clock periods.

- **EXREF: External Reference**
  1: The external forcing of references is enabled, ADC references are the ADCREFN and ADCREFP pads.
  0: The external forcing of references is disabled, ADC reference is given by the RS field.

- **SHD: Sample-and-Hold Disabled**
  1: The Sample and Hold is disabled.
  0: The Sample and Hold is enabled.
  note: when set to one, sequencer 1 is turned off, as a consequence the ADC pipeline latency is decreased by one ADC clock period.

- **RS: Reference Source**
  0: Internal 1V reference.
  1: Internal 0.6 *VDDANA reference.
  2: External reference ADCREF0 over chip analog ground.
  3: External reference ADCREF1 over chip analog ground.

- **FRM: Free Running Mode**
  1: The free running mode is enabled, sequencer 0 performs conversions continuously.
  0: The free running modeis disabled.
  note: once in this mode, sequencer 1 requests cannot be serviced.

- **SSMQ: Single Sequencer Mode**
  1: The single sequencer mode is enabled, sequencers 0 and 1 are merged, increasing the number of conversions per sequence.
  0: The single sequencer mode is disabled, SEQ0 and SEQ1 are in simultaneous mode.

- **SLEEP: Sleep Mode Selection**
  1: The power saving mode is enabled. The analog ADC block is powered off after each conversion.
  0: The power saving mode is disabled.
  note: when enabled, start-up time is required before each new conversion.

- **ADCEN: ADC Enable**
  1: The ADC controller is enabled, the analog ADC block is powered-on according to SLEEP mode.
  0: The ADC controller is disabled, the analog ADC block is powered-off.

36.7.3 Status Register

Name: SR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	STATE1			
23	22	21	20	19	18	17	16
-	-	-		STATE0			
15	14	13	12	11	10	9	8
RUNT	SUTD	MSOC1	MSOC0	WM1	WM0	-	-
7	6	5	4	3	2	1	0
LOVR1	OVR1	SEOC1	SEOS1	LOVR0	OVR0	SEOC0	SEOS0

- **STATE1: Sequencer 1 State Register**
  This field is set to the current conversion identifier.

- **STATE0: Sequencer 0 State Register**
  This field is set to the current conversion identifier.

- **RUNT: Running Timer Status**
  This bit is set when the internal timer is started.
  This bit is cleared when the internal timer is stopped.

- **SUTD: Start-up Time Done**
  This bit is set when a start-up done event occurs.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **MSOC1: Sequencer 1 Missed Start-Of-Conversion**
  This bit is set when a start-of-conversion is missed.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **MSOC0: Sequencer 0 Missed Start-Of-Conversion**
  This bit is set when a start-of-conversion is missed.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **WM1: Window Monitor 1**
  This bit is set when the watched result value goes to the defined window.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **WM0: Window Monitor 0**
  This bit is set when the watched result value goes to the defined window.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **LOVR1: Sequencer 1 Last Converted Value Overrun**
  This bit is set when an overrun error occurs on the LCV register.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **OVR1: Sequencer 1 Overrun Error**
  This bit is set when an overrun error occurs.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **SEOC1: Sequencer 1 End Of Conversion**
  This bit is set when an end of conversion occurs.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **SEOS1: Sequencer 1 End Of Sequence**
  This bit is set when an end of sequence occurs.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **LOVR0: Sequencer 0 Last Converted Value Overrun**
  This bit is set when an overrun error occurs on the LCV register.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **OVR0: Sequencer 0 Overrun Error**
  This bit is set when an overrun error occurs.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **SEOC0: Sequencer 0 End Of Conversion**
  This bit is set when an end of conversion occurs.
  This bit is cleared when the corresponding bit in SCR is written to one.

- **SEOS0: Sequencer 0 End Of Sequence**
  This bit is set when an end of sequence occurs.
  This bit is cleared when the corresponding bit in SCR is written to one.

36.7.4 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-		-	-	-	-

15	14	13	12	11	10	9	8
-	SUTD	MSOC1	MSOC0	WM1	WM0	-	-

7	6	5	4	3	2	1	0
LOVR1	OVR1	SEOC1	SEOS1	LOVR0	OVR0	SEOC0	SEOS0

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register clears the corresponding bit in SR.

36.7.5 Status Set Register

Name: SSR

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-		-	-	-	-
15	14	13	12	11	10	9	8
-	SUTD	MSOC1	MSOC0	WM1	WM0	-	-
7	6	5	4	3	2	1	0
LOVR1	OVR1	SEOC1	SEOS1	LOVR0	OVR0	SEOC0	SEOS0

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register sets the corresponding bit in SR.

36.7.6 Sequencer n Configuration Register

Name: SEQCFGn

Access Type: Read/Write

Offset: 0x14 + n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-		CNVNB			

15	14	13	12	11	10	9	8
-	-	SRES		-	-	TRGSEL	

7	6	5	4	3	2	1	0
-	-	-	SHDYN	OVSX2	SOCB	HWLA	SA

- **CNVNB: Number of Conversions in a Sequence**
  The number of conversions to perform in the sequence is (CNVNB+1).

- **SRES: Resolution**
  0: 12 bits.
  1: 10 bits.
  2: 8 bits.
  3: Reserved.

- **TRGSEL: Trigger Selection**
  0: Software.
  1: Internal ADC timer.
  2: Event controller source.
  3: Continuous.

- **SHDYN: Sample and Hold Dynamic Mode**
  1: The SH dynamic mode,is enabled, a conversion takes two ADC clock cycles, SH is reseted on the first cycle.
  0: The SH dynamic mode is disabled, a conversion takes a single ADC clock cycle.

- **OVSX2: Oversampling X2**
  1: The oversampling mode is enabled, a conversion takes two ADC clock cycles.
  0: The oversampling mode is disabled, a conversion takes a single ADC clock cycle.

- **SOCB: Start of Conversion Behavior**
  1: The SOCB mode is enabled, a single conversion is performed on a SOC event.
  0: The SOCB mode is disabled, a complete sequence is performed on a SOC event.

- **HWLA: Half Word Left Adjust**
  1: The HWLA mode is enabled.
  0: The HWLA mode is disabled.

- **SA: Software Acknowledge**
  1: The SA mode is enabled.
  0: The SA mode is disabled.

36.7.7 Sequencer n Sample and Hold Gain for Each Conversion

Name: SHGn

Access Type: Read/Write

Offset: 0x1C+ n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	GCNV7			-	GCNV6		

23	22	21	20	19	18	17	16
-	GCNV5			-	GCNV4		

15	14	13	12	11	10	9	8
-	GCNV3			-	GCNV2		

7	6	5	4	3	2	1	0
-	GCNV1			-	GCNV0		

- **GCNV7: Sequencer n Conversion 7 Sample and Hold Gain**
- **GCNV6: Sequencer n Conversion 6 Sample and Hold Gain**
- **GCNV5: Sequencer n Conversion 5 Sample and Hold Gain**
- **GCNV4: Sequencer n Conversion 4 Sample and Hold Gain**
- **GCNV3: Sequencer n Conversion 3 Sample and Hold Gain**
- **GCNV2: Sequencer n Conversion 2 Sample and Hold Gain**
- **GCNV1: Sequencer n Conversion 1 Sample and Hold Gain**
- **GCNV0: Sequencer n Conversion 0 Sample and Hold Gain**

36.7.8 Sequencer n INP Conversions 0 to 3 Selection

Name: INPSEL0n

Access Type: Read/Write

Offset: 0x24+ n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	CNV3			
23	22	21	20	19	18	17	16
-	-	-	-	CNV2			
15	14	13	12	11	10	9	8
-	-	-	-	CNV1			
7	6	5	4	3	2	1	0
-	-	-	-	CNV0			

- **CNV3: Sequencer n INP Identifier of the 4th Conversion to Perform**
- **CNV2: Sequencer n INP Identifier of the 3rd Conversion to Perform**
- **CNV1: Sequencer n INP Identifier of the 2nd Conversion to Perform**
- **CNV0: Sequencer n INP Identifier of the 1st Conversion to Perform**

36.7.9 Sequencer n INP Conversions 4 to 7 Selection

Name: INPSEL1n

Access Type: Read/Write

Offset: 0x2C+ n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	CNV7			
23	22	21	20	19	18	17	16
-	-	-	-	CNV6			
15	14	13	12	11	10	9	8
-	-	-	-	CNV5			
7	6	5	4	3	2	1	0
-	-	-	-	CNV4			

- **CNV7: Sequencer 0 INP Identifier of the 8th Conversion to Perform**
- **CNV6: Sequencer 0 INP Identifier of the 7th Conversion to Perform**
- **CNV5: Sequencer 0 INP Identifier of the 6th Conversion to Perform**
- **CNV4: Sequencer 0 INP Identifier of the 5th Conversion to Perform**

36.7.10 Sequencer n INN Conversions 0 to 3 Selection

Name: INNSEL0n

Access Type: Read/Write

Offset: 0x34+ n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	CNV3			
23	22	21	20	19	18	17	16
-	-	-	-	CNV2			
15	14	13	12	11	10	9	8
-	-	-	-	CNV1			
7	6	5	4	3	2	1	0
-	-	-	-	CNV0			

- **CNV3: Sequencer n INN Identifier of the 4th Conversion to Perform**
- **CNV2: Sequencer n INN Identifier of the 3rd Conversion to Perform**
- **CNV1: Sequencer n INN Identifier of the 2nd Conversion to Perform**
- **CNV0: Sequencer n INN Identifier of the 1st Conversion to Perform**

36.7.11 Sequencer n INN Conversions 4 to 7 Selection

Name: INNSEL1n

Access Type: Read/Write

Offset: 0x3C+ n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	CNV7			

23	22	21	20	19	18	17	16
-	-	-	-	CNV6			

15	14	13	12	11	10	9	8
-	-	-	-	CNV5			

7	6	5	4	3	2	1	0
-	-	-	-	CNV4			

- **CNV7: Sequencer n INN Identifier of the 8th Conversion to Perform**
- **CNV6: Sequencer n INN Identifier of the 7th Conversion to Perform**
- **CNV5: Sequencer n INN Identifier of the 6th Conversion to Perform**
- **CNV4: Sequencer n INN Identifier of the 5th conversion to Perform**

36.7.12 Clock Divider Register

Name: CKDIV

Access Type: Read/Write

Offset: 0x44

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	CNT[8]

7	6	5	4	3	2	1	0
CNT[7:0]							

- **CNT: Max Counter Value**
  Number of ADC clock cycles to count: (CNT + 1) * 2.

36.7.13 Internal Timer Register

Name: ITIMER

Access Type: Read/Write

Offset: 0x48

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	ITMC[16]

15	14	13	12	11	10	9	8
ITMC[15:8]							

7	6	5	4	3	2	1	0
ITMC[7:0]							

- **ITMC: Internal Timer Max Counter**

  Number of ADC clock cycles to wait for is (ITMC + 1).

  note: This allows SOC period up to 167 ms when CkADC clock is running at 1.5 MHz.

36.7.14 Window Monitor n Configuration Register

Name: WCFGn

Access Type: Read/Write

Offset: 0x58+ n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	SRC				WM		

23	22	21	20	19	18	17	16
HT[11:4]							

15	14	13	12	11	10	9	8
HT[3:0]				LT[11:8]			

7	6	5	4	3	2	1	0
LT[7:0]							

- **SRC:**
  Index of the result register to monitor (0 to 16).

- **WM: Window Mode**
  0: No window mode.
  1: Mode 1: RES(SRC) < HT.
  2: Mode 2: RES(SRC) > LT.
  3: Mode 3: LT< RES(SRC) < HT.
  4: Mode 4: (LT >= RES(SRC)) || (RES(SRC) >= HT).
  5: Reserved.
  6: Reserved.
  7: Reserved.

- **HT: High Threshold**

- **LT: Low Threshold**

36.7.15 Sequencer n Last Converted Value

Name: LCVn

Access Type: Read-only

Offset: 0x60+ n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
LCV[31:24]							

23	22	21	20	19	18	17	16
LCV[23:16]							

15	14	13	12	11	10	9	8
LCV[15:8]							

7	6	5	4	3	2	1	0
LCV[7:0]							

- **LCV: Last Converted Value**
  This field is set by hardware to the last sequencer converted value. Depending on precision, the higher bits are padded with the sign bit.

36.7.16 ADC calibration register

Name: ADCCAL

Access Type: Read/Write

Offset: 0x68

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-		OCAL					

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	GCAL[14:8]						

7	6	5	4	3	2	1	0
GCAL[7:0]							

- **OCAL: Offset Calibration**
- **GCAL: Gain Calibration**

36.7.17 SH Calibration Register

Name: SHCAL

Access Type: Read/Write

Offset: 0x6C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-		-	-	-	-	GAIN1[9:8]	

23	22	21	20	19	18	17	16
GAIN1[7:0]							

15	14	13	12	11	10	9	8
-	-					GAIN0[9:8]	

7	6	5	4	3	2	1	0
GAIN0[7:0]							

- **GAIN1: Sample and Hold 1 Gain Calibration**
- **GAIN0: Sample and Hold 0 Gain Calibration**

36.7.18 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x70

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-		-	-	-	-

15	14	13	12	11	10	9	8
-	SUTD	MSOC1	MSOC0	WM1	WM0	-	-

7	6	5	4	3	2	1	0
LOVR1	OVR1	SEOC1	SEOS1	LOVR0	OVR0	SEOC0	SEOS0

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

36.7.19 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x74

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-		-	-	-	-
15	14	13	12	11	10	9	8
-	SUTD	MSOC1	MSOC0	WM1	WM0	-	-
7	6	5	4	3	2	1	0
LOVR1	OVR1	SEOC1	SEOS1	LOVR0	OVR0	SEOC0	SEOS0

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

36.7.20 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x78

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-		-	-	-	-
15	14	13	12	11	10	9	8
-	SUTD	MSOC1	MSOC0	WM1	WM0	-	-
7	6	5	4	3	2	1	0
LOVR1	OVR1	SEOC1	SEOS1	LOVR0	OVR0	SEOC0	SEOS0

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

36.7.21 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x7C

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.

- **VERSION: Version Number**
  Version number of the module. No functionality associated.

36.7.22 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x80

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
N							

7	6	5	4	3	2	1	0
M							

- **N: Number of Channels**
- **M: Number of States per Sequencer**

36.7.23 Result Register

Name: RESn

Access Type: Read-only

Offset: 0x84+ n * 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RES[31:24]							

23	22	21	20	19	18	17	16
RES[23:16]							

15	14	13	12	11	10	9	8
RES[15:8]							

7	6	5	4	3	2	1	0
RES[7:0]							

- **RES: Result register**
  Contains value of conversion n.

36.8 Module configuration

The specific configuration for each ADC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks according to the table in the System Bus Clock Connections section.

Table 36-13. Module configuration

Feature	ADCIFA
NBCONV	8

Table 36-14. Module clock name

Module name	Clock name	Description
ADCIFA	CLK_ADCIFA	Peripheral Bus clock from the PBC clock domain

Table 36-15. Register Reset Values

Register	Reset Value
VERSION	0x00000110
PARAMETER	0X00000808

The differential inputs of the ADC are configured through the following registers of the ADCIFA:

- INPSEL00, INPSEL01, INNSEL00 and INNSEL01 for the sequencer 0
- INPSEL10, INPSEL11, INNSEL10 and INNSEL11 for the sequencer 1

The configuration allows to select pin or internal voltage.

The ADC voltage reference can be selected as external reference through the RS register of the ADCIFA.

For detail, see the ADCIFA chapter.

The following table defines the valid settings for the CONV field of the INPSELxy and INNSELxy registers in the ADCIFA. This setting defines the mapping of the ADC input voltage.

Table 36-16. INP0/1 selection

INPSEL00[CONVi], INPSEL10[CONVi], INPSEL01[CONVi], INPSEL11[CONVi]	Name	Connection
0	ADCIN0	See Peripheral Multiplexing on I/O line chapter
1	ADCIN1	
2	ADCIN2	
3	ADCIN3	
4	ADCIN4	
5	ADCIN5	
6	ADCIN6	
7	ADCIN7	
8	DAC0_int	Internal output of the DAC0
10	GNDANA	Analog Ground

Table 36-17. INN0/1 selection

INNSEL00[CONVi], INNSEL10[CONVi], INNSEL01[CONVi], INNSEL11[CONVi]	Name	Connection
0	ADCIN8	See Peripheral Multiplexing on I/O line chapter
1	ADCIN9	
2	ADCIN10	
3	ADCIN11	
4	ADCIN12	
5	ADCIN13	
6	ADCIN14	
7	ADCIN15	
8	DAC1_int	Internal output of the DAC1
9	GNDANA	Analog Ground

Table 36-18. External Reference selection

RS	Name	Connection
0	Internal 1V reference	
1	Internal 0.6*VDDANA reference	
2	ADCREF0	See Peripheral Multiplexing on I/O line chapter
3	ADCREF1	

37. DACIFB Interface (DACIFB)

Rev.: 1.1.0.1

37.1 Features

- **12-bit resolution**
- **Flexible conversion range**
- **1 continuous time or 2 Sample/Hold (S/H) outputs**
- **Multiple trigger sources for each channel**
- **Built-in offset and gain calibration**
- **Can be used as input to analog comparator or ADC (as an internal wire and without S/H stage)**
- **Two PDCA channels**
- **Low-power mode**

37.2 Overview

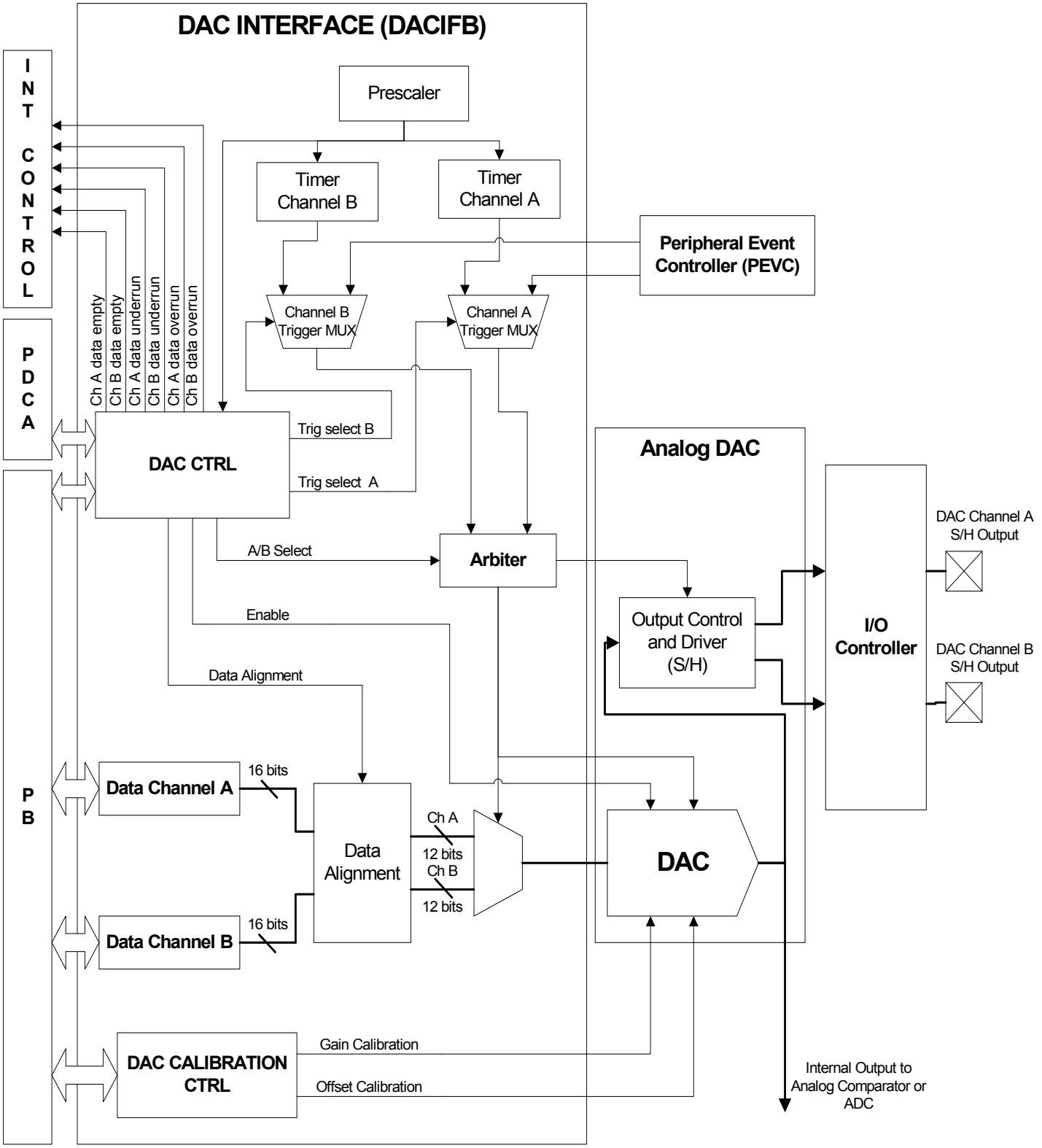
The DAC interface (DACIFB) controls a DAC that converts digital values to analog voltages with 12-bit resolution. The output from the DAC can either be continuous to one pin, or fed to two different pins using a sample and hold circuitry. Options like low power mode and gain and offset calibration are available.

The output signal swing is defined by the reference voltage AREF. The DAC operates in unipolar mode, i.e. output voltage shifts within the 0V to AREF range. The following sources are available as AREF reference voltage:

- VDDANA
- An external reference applied to the DACREF input pin.

37.3 Block Diagram

Figure 37-1. DACIFB Block Diagram



37.4 I/O Lines Description

Table 37-1. I/O Lines Description

Pin Name	Pin Description	Type	Active Level
DACA	DAC channel A analog output	Output	N/A
DACB	DAC channel B analog output	Output	N/A
DACREF	DAC voltage reference	Input	N/A

37.5 Product Dependencies

37.5.1 I/O Lines

The pins used for interfacing the DAC may be multiplexed with GPIO lines. The programmer must first program the GPIO controller to assign the desired DAC pins to their peripheral function. If I/O lines of the DAC are not used by the application, they can be used for other purposes by the GPIO controller.

37.5.2 Power Management

If the CPU enters a sleep mode that disables CLK_DACIFB used by the DACIFB, the DACIFB will stop functioning and will resume operation after the system wakes up from sleep mode.

37.5.3 Clocks

The DACIFB is clocked through the Power Manager (PM), therefore the programmer must first configure the PM to enable the CLK_DACIFB clock.

This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the DACIFB before disabling the clock, to avoid freezing the DACIFB in an undefined state.

37.5.4 Interrupts

The DACIFB interrupt lines are connected to the internal sources of the interrupt controller. Using the DACIFB interrupts requires the interrupt controller to be programmed first.

37.5.5 Peripheral Events

The DACIFB peripheral events are connected via the Peripheral Event Controller. Refer to the Peripheral Event Controller chapter for details.

37.5.6 Debug Operation

The DACIFB is disabled during debug operation, unless the Run In Debug bit in the Development Control Register is set and the bit corresponding to the DACIFB is set in the Peripheral Debug Register (PDBG). Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual, and the OCD Module Configuration section, for details.

The DACIFB is debug-mode aware. When the CPU is in debug mode, all the incoming triggers are blocked, therefore all DMA based conversions are halted upon debug mode activation. The auto-refresh functionality is kept active so that the last converted value remains visible on both channels outputs.

Apart from the auto-trig mode, it is possible to perform “one shot” conversions triggered by write accesses to the data register.

37.6 Functional Description

37.6.1 Basic Operation

37.6.1.1 Output channels

The output from the DAC can either be continuous to one pin (DAC channel A only), or fed to two different pins using a sample and hold circuitry (S/H). With S/H these two outputs can act independently and create two different analog signals, different in both voltage and frequency.

The two S/H outputs have individual data and conversion control registers.

The DAC output may be used as internal input signal to other peripherals, such as the Analog Comparator or the ADC. Only the DAC internal output can be used as internal input, the S/H outputs can not be used as such. Note that in this internal routing mode, both S/H modules must be deactivated in order to avoid disturbances on the internal channel.

37.6.1.2 Timing constraints

Some timing constraints must be observed in order to make sure the S/H circuitry operates correctly. These are relative to the frequency of the peripheral clock of the DACIFB, as this will affect the charging/discharging periods of the S/H circuitry. Not meeting these constraints will result in reducing the accuracy of the DAC conversions.

The DAC sampling time is the time interval between two conversions. Without S/H operating, this figure should be equal or greater than the DAC minimum sampling period (corresponding to the DAC maximum sampling frequency). With S/H operating, this figure should be equal or greater than 1.5 times the DAC minimum sampling period.

When the sampling frequency is too low, the S/H circuitry may let the output voltage drop significantly between two consecutive conversions. If enabled, the auto-refresh mode will automatically repeat continuously the conversion of the last channel data. This allows maintaining the voltage level on the S/H outputs, whenever the time elapsed between two data to be converted is too long. The DAC refresh time is the time interval between two channel data updates, it can not be smaller than the minimum sampling period.

The analog DAC startup time is non-null, therefore a waiting period of a few CLK_DACIFB clock cycles must be observed before considering conversion of the first data.

Note that if S/H is enabled, the actual sampling period will be 1.5 times greater than the programmed sampling period.

37.6.1.3 Starting a conversion

Conversions are either performed upon writes to the data registers or triggered by an incoming event (auto-trig mode). Both application software and the Peripheral DMA controller may write to the data registers.

Using the Peripheral DMA Controller to write data to the DACIFB, together with an event input to trigger conversions, gives the most accurate timing for conversions.

The Peripheral DMA Controller data transfer rate depends on the sampling frequency imposed by the event line. The DACIFB sends a request to the Peripheral DMA Controller, and once the request is granted (Peripheral DMA Controller acknowledge) the conversion is performed upon reception of a trigger event.

37.6.1.4 Data Registers

Data to be converted is taken from two registers, one for each channel: Data Register 0 (DR0) for channel A and Data Register 1 (DR1) for channel B.

Alternatively both samples to be converted can be written to DR0 in a single write cycle, in this configuration the values for channel B and A are written to the upper and the lower half words of DR0, respectively. This operation is possible only if the DAC Dual Data in Data Register A bit of the Configuration register (CFR.DDA) is enabled.

While the field reserved for the data to be converted is 16 bits wide, only the 12 lower bits are considered for conversion. In order to match the expected data alignment, rounded right and left shifts are programmable within a separate register for each data channel.

37.6.2 Advanced Operation

37.6.2.1 Prescaler

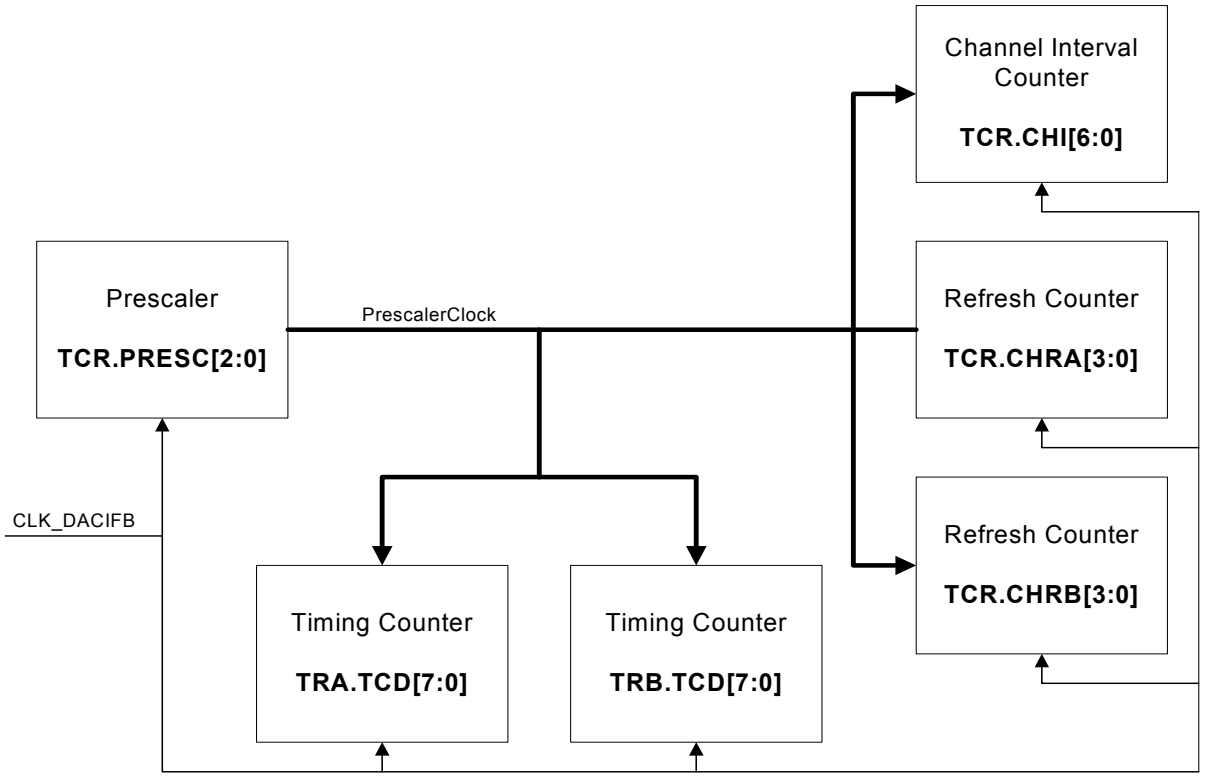
A programmable prescaler generates a divided clock signal from the system clock. This signal is then fed to the following DACIFB programmable counters, as shown on Figure 37-2:

- The channel interval counter, which sets the minimum time interval between two samples, or in other words, the maximum sampling frequency (see CHI bitfield in the TCR register).
- The S/H refresh counter which, in refresh mode, counts a defined number of prescaled clock ticks (corresponding to the refresh time) before repeating the conversion of the last received data (see CHRA and CHRB bitfields in the TCR register).
- Trigger event timer counters for both channels which, in timer trigged mode, count a defined number of prescaled clock ticks before triggering a conversion (see the TRA and TRB registers).

The limitations described in the “Timing constraints” paragraph must be taken into consideration when configuring these programmable counters.

In addition to these constraints, when both channels are in use with auto-refresh mode enabled, the refresh rate should not be significantly higher than the sampling rate on the other channel as this might cause unexpected behavior on the latter channel.

Figure 37-2. DAC Timing Counters



37.6.2.2 *Low Power mode*

In order to reduce the power consumption during DAC conversions, the DAC Low Power mode may be enabled. In low power mode, the DAC is turned off between each conversion.

Conversion time will be longer if new conversions are started in this mode: a fourfold increase of the DAC's output settling time should be expected.

37.6.2.3 *Calibration*

To achieve optimal accuracy, it is possible to calibrate both gain (GOC.GCR) and offset error (GOC.OCR) in the DAC.

Gain and Offset are not calibrated automatically and this must be done by software. To perform this operation, the DAC internal output must be beforehand routed to the ADC using the CR.IE bit.

The test values converted by the DAC are sampled by the ADC which returns a measured value of the DAC output. Calculating the difference between a series of DACIFB channel input values and their respective effective levels after conversion makes it possible to deduce both DAC gain and offset biases.

To get the best calibration result, it is recommended to use the same AREF voltage, output channel selection, sampling time, and refresh interval when calibrating as in normal DAC operation.

Including errors, the DAC output value can be expressed as:

$$V_{DACxX} = gain \times (DATA_{CHx} / 0xFFF) + offset$$

In an ideal DAC, gain is 1 and offset 0.

37.6.3 Interrupts

An interrupt request will be generated if the corresponding bit in the Interrupt Mask Register (IMR) is set. Bits in IMR are set by writing a one to the corresponding bit in the Interrupt Enable Register (IER), and cleared by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The interrupt request remains active until the corresponding bit in SR is cleared by writing a one to the corresponding bit in the Status Clear Register (SCR).

After processing a channel input data, if the data input buffer is empty, the DACIFB signals a data empty interrupt to the interrupt controller.

An underrun interrupt will be generated if two consecutive trigger events are issued without any new channel data being fed to the DACIFB in the meantime.

An overrun interrupt will be generated if an additional channel data is sent to the DACIFB while the input buffer is already full.

37.6.4 Peripheral Events

Channel conversions can be triggered by an independent event source. A simple arbiter prioritizes trigger event requests if the two channels are activated at the same time.

Trigger events for both channels are taken either from the PEVC input or from the DACIFB internal trigger timers. These two timers are set up separately and both use PrescalerClock as their reference clock (see the TRA and TRB registers).

37.7 User Interface

Table 37-2. DACIFB Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Read/Write	0x00000000
0x04	Configuration Register	CFR	Read/Write	0x00000000
0x08	Event Input Control Register	ECR	Read/Write	0x00000000
0x0C	Timing Control Register	TCR	Read/Write	0x00000000
0x10	Interrupt Enable Register	IER	Write-only	-
0x14	Interrupt Disable Register	IDR	Write-only	-
0x18	Interrupt Mask Register	IMR	Read-only	0x00000000
0x1C	Status Register	SR	Read-only	0x00000000
0x20	Status Clear Register	SCR	Write-only	-
0x24	Data Register Control Channel A	DRCA	Read/Write	0x00000001
0x28	Data Register Control Channel B	DRCB	Read/Write	0x00000001
0x2C	Data Register 0	DR0	Read/Write	0x00000000
0x30	Data Register 1	DR1	Read/Write	0x00000000
0x34	Gain and Offset Calibration Register	GOC	Read/Write	0x00000000
0x38	Timer Register Channel A	TRA	Read/Write	0x00000000
0x3C	Timer Register Channel B	TRB	Read/Write	0x00000000
0x40	Version Register	VERSION	Read-only	- (1)

1. The reset value for this register is device specific. Please refer to the Module Configuration section at the end of this chapter.

37.7.1 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	ARBE	ARAE	TRBE	TRAE
23	22	21	20	19	18	17	16
-	-	-	-	-	-	BOE	AOE
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	EN

- **ARBE: DAC Auto Refresh Channel B Enable**
  0: auto-refresh on channel B is disabled.
  1: auto-refresh on channel B is enabled.
- **ARAE: DAC Auto Refresh Channel A Enable**
  0: auto-refresh on channel A is disabled.
  1: auto-refresh on channel A is enabled.
- **TRBE: DAC Timer Register Channel B Enable**
  0: the timer generating a clocked trigger on channel B is disabled.
  1: the timer generating a clocked trigger on channel B is enabled.
- **TRAE: DAC Timer Register Channel A Enable**
  0: the timer generating a clocked trigger on channel A is disabled.
  1: the timer generating a clocked trigger on channel A is enabled.
- **BOE: DAC Channel B Output Enable**
  0: channel B analog output is disabled.
  1: channel B analog output is enabled.
- **AOE: DAC Channel A Output Enable**
  0: channel A analog output is disabled.
  1: channel A analog output is enabled.
- **EN: DAC Enable**
  0: DAC is disabled.
  1: DAC is enabled.

37.7.2 Configuration Register

Name: CFR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	CHC	

23	22	21	20	19	18	17	16
-	-	-	-	-	-	ABE	AAE

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	REF

7	6	5	4	3	2	1	0
-	-	-	-	-	DSE	DDA	LP

- **CHC: DAC Channel Configuration**
  These bits control whether the DAC should operate with sample and hold on outputs or not.

CHC	Description
00	S/H modules on both channels are deactivated. Internal routing only.
01	Single channel operation: conversions on channel A only (S/H module activated).
10	Single channel operation: conversions on channel B only (S/H module activated).
11	Dual channel operation: conversions on both channels with both S/H modules activated.

- **ABE: DAC Auto Triggered Mode Enable Channel B**
  0: The conversion is triggered by the data register write access.
  1: the incoming event (from the event line selected in the ECR Register) triggers the conversion.
- **AAE: DAC Auto Triggered Mode Enable Channel A**
  0: The conversion is triggered by the data register write access.
  1: the incoming event (from the event line selected in the ECR Register) triggers the conversion.
- **REF: DAC Reference Selection**
  This bit controls the voltage reference selection and therefore the output voltage range of the DAC.

REF	Voltage Reference Selection
0	External Reference (VREF+ pin)
1	VDDANA

- **DSE: DAC Data Setup Extra Clock Cycle**
  0: No extra clock latency.
  1: Add an extra clock cycle latency between data written and start of conversion. This may be useful when the DAC clock is running fast. Adding an extra clock cycle latency might help meeting the data setup time constraint.
- **DDA: DAC Dual Data in Data Register A**
  0:No dual data in DR0.
  1:Dual data in DR0. This allows writing two 16-bit wide data words in a single write operation to the DR0 register. In this case the 16 upper bits are assigned to the channel B data word while the lower 16 bits remain assigned to the channel A data word.
- **LP: DAC Low Power Reduction Mode**
  0: DAC low power mode disabled.
  1: DAC low power mode enabled.

37.7.3 Event Input Control Register

Name: ECR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	ESLB

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	ESLA

- **ESLB: DAC Event Input Selection Channel B**
  0: the channel B trigger timer is used
  1: the peripheral event controller input is used
- **ESLA: DAC Event Input Selection Channel A**
  0: the channel A trigger timer is used
  1: the peripheral event controller input is used

37.7.4 Timing Control Register

Name: TCR

Access Type: Read/Write

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	PRESC		

23	22	21	20	19	18	17	16
-	CHI						

15	14	13	12	11	10	9	8
-	-	-	-	CHRB			

7	6	5	4	3	2	1	0
-	-	-	-	CHRA			

- **PRESC: DAC Timer Prescaler**
  The prescaling ratio between PrescalerClock and CLK_DACIFB.

PRESC	Description
000	clk_dacifb / 1
001	clk_dacifb / 2
010	clk_dacifb / 4
011	clk_dacifb / 8
100	clk_dacifb / 16
101	clk_dacifb / 32
110	clk_dacifb / 64
111	clk_dacifb / 128

- **CHI: DAC Channel Interval Control**
  Number of PrescalerClock ticks counted as the minimum time gap between two consecutive conversions . This should not be lower than the minimum sampling period.

- **CHRn : DAC Channel Refresh Timing Control Channel n**
  The time interval between each channel output refresh. This interval avoids losing accuracy of the converted value between two consecutive conversions when the sampling rate is low.

CHRn	Description
0000	PrescalerClock / 2
0001	PrescalerClock / 4
0010	PrescalerClock / 8
0011	PrescalerClock / 16
0100	PrescalerClock / 32
0101	PrescalerClock / 64
0110	PrescalerClock / 128
0111	PrescalerClock / 256
1000	PrescalerClock / 512
1001	PrescalerClock / 1024
others	Reserved

37.7.5 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x10

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	DEB	DEA

15	14	13	12	11	10	9	8
-	-	-	-	-	-	UA	UA

7	6	5	4	3	2	1	0
-	-	-	-	-	-	OB	OA

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

37.7.6 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x14

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	DEB	DEA

15	14	13	12	11	10	9	8
-	-	-	-	-	-	UB	UA

7	6	5	4	3	2	1	0
-	-	-	-	-	-	OB	OA

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

37.7.7 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	DEB	DEA

15	14	13	12	11	10	9	8
-	-	-	-	-	-	UB	UA

7	6	5	4	3	2	1	0
-	-	-	-	-	-	OB	OA

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

37.7.8 Status Register

Name: SR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	DEB	DEA

15	14	13	12	11	10	9	8
-	-	-	-	-	-	UB	UA

7	6	5	4	3	2	1	0
-	-	-	-	-	-	OB	OA

- **DEB: DAC Data Register Empty Channel B Flag**
  0: Data register not empty, writing to the data register may cause losing a conversion value.
  1: Data register for channel B is empty, meaning that a new channel input data may be written.
- **DEA: DAC Data Register Empty Channel A Flag**
  0: Data register not empty, writing to the data register may cause losing a conversion value.
  1: Data register for channel A is empty, meaning that a new channel input data may be written.
- **UB: DAC Underrun Interrupt Channel B Flag**
  0: No underrun in channel B has occurred.
  1: Underrun has occurred, at least two consecutive trigger events were received without any new incoming data on channel B in the meantime.
- **UA: DAC Underrun Interrupt Channel A Flag**
  0: No underrun in channel A has occurred.
  1: Underrun has occurred, at least two consecutive trigger events were received without any new incoming data on channel A in the meantime.
- **OB: DAC Overrun Interrupt Channel B Flag**
  0: No overrun in channel B has occurred.
  1: Overrun has occurred, a new input data was received on channel B while the input buffer was already full.
- **OA: DAC Overrun Interrupt Channel A Flag**
  0: No overrun in channel A has occurred.
  1: Overrun has occurred, a new input data was received on channel A while the input buffer was already full.

37.7.9 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	DEB	DEA

15	14	13	12	11	10	9	8
-	-	-	-	-	-	UB	UA

7	6	5	4	3	2	1	0
-	-	-	-	-	-	OB	OA

Writing a bit to one will clear the corresponding bit in ISR.
Writing a bit to zero has no effect.

37.7.10 Data Register Control Channel A

Name: DRCA

Access Type: Read/Write

Offset: 0x24

Reset Value: 0x00000001

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	DSD	DSV		

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DRN

- **DSD: DAC Data Shift Direction**
  0: DAC input value to be converted is right aligned
  1: DAC input value to be converted is left aligned.
- **DSV: DAC Data Shift Value**
  The number of left or right shifts to be performed on the 16-bits data word before being fed to the DAC.
  Up to 4 left shifts and 4 right shifts are possible. Set bit 3 to obtain a left shift, leave it de-asserted to perform a right shift.
- **DRN: DAC Data Rounding Enable**
  0: No rounding
  1: Rounding with right shifting is enabled. This adds "1" to the LSB before the last right shift. This feature is enabled by default.

37.7.11 Data Register Control Channel B

Name: DRCB

Access Type: Read/Write

Offset: 0x28

Reset Value: 0x00000001

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	DSD	DSV		

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DRN

- **DSD: DAC Data Shift Direction**
  0: DAC input value to be converted is right aligned
  1: DAC input value to be converted is left aligned.
- **DSV: DAC Data Shift Value**
  The number of left or right shifts to be performed on the 16-bits data word before being fed to the DAC.
  Up to 4 left shifts and 4 right shifts are possible. Set bit 3 to obtain a left shift, leave it de-asserted to perform a right shift.
- **DRN: DAC Data Rounding Enable**
  0: No rounding
  1: rounding with right shifting is enabled. This adds "1" to the LSB before the last right shift. This feature is enabled by default.

37.7.12 Data Register Channel 0

Name: DR0

Access Type: Read/Write

Offset: 0x2C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DCB							

23	22	21	20	19	18	17	16
DCB							

15	14	13	12	11	10	9	8
DCA							

7	6	5	4	3	2	1	0
DCA							

- **DCB: DAC Data Channel B**
  The right-aligned 12-bit value to be converted on channel B, when the DDA bit within the CFR register is activated. This allows conversions on both channels in a single register write cycle. When DDA is deactivated this field is ignored, and the data to be converted on channel B should then be written to the DR1 register.
- **DCA: DAC Data Channel A**
  The right-aligned 12-bit value to be converted on channel A.

37.7.13 Data Register Channel 1

Name: DR1

Access Type: Read/Write

Offset: 0x30

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
DCB							

7	6	5	4	3	2	1	0
DCB							

- **DCB: DAC Data Channel B**
  The right-aligned 12-bit value to be converted on channel B.

37.7.14 Gain and Offset Calibration Register

Name: GOC

Access Type: Read/Write

Offset: 0x34

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
GCR							

23	22	21	20	19	18	17	16
GCR							

15	14	13	12	11	10	9	8
OCR							

7	6	5	4	3	2	1	0
OCR							

- **GCR: DAC Gain Calibration Value**
  These bits are used to compensate the gain error in the DAC.
  Note that the size of the GCR field can change depending of implementation. See the Module Configuration section.
- **OCR: DAC Offset Calibration Value**
  These bits are used to compensate the offset error in the DAC.
  Note that the size of the OCR field can change depending of implementation. See the Module Configuration section.

37.7.15 Timer Register Channel A

Name: TRA

Access Type: Read/Write

Offset: 0x38

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
TRL	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
TCD							

- **TRL: DAC Timer Reload**
  Write this bit to one to re-initialize the counter.
- **TCD: DAC Timer Count Down Value**
  These bits are used to program the timing counter countdown value.
  The timer counts from this value down to zero and then issues a trigger signal before reloading the count register.

37.7.16 Timer Register Channel B

Name: TRB

Access Type: Read/Write

Offset: 0x3C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
TRL	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
TCD							

- **TRL: DAC Timer Reload**
  Write this bit to one to re-initialize the counter.
- **TCD: DAC Timer Count Down Value**
  These bits are used to program the timing counter countdown value.
  The timer counts from this value down to zero and then issues a trigger signal before reloading the count register.

37.7.17 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x40

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	VARIANT			

15	14	13	12	11	10	9	8
-	-	-	-	VERSION			

7	6	5	4	3	2	1	0
VERSION							

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

37.8 Module Configuration

The specific configuration for each DACIFB instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 37-3. DACIFB Configuration

Feature	DACIFB0	DACIFB1
GOC.GCR size	8-bit	8-bit
GOC.OCR size	9-bit	9-bit

Table 37-4. DACIFB Clock Name

Module Name	Clock Name	Description
DACIFB0	CLK_DACIFB0	Peripheral Bus clock from the PBA clock domain
DACIFB1	CLK_DACIFB1	Peripheral Bus clock from the PBA clock domain

Table 37-5. Register Reset Values

Register	Reset Value
VERSION	0x00000110

38. aWire UART (AW)

Rev: 2.3.0.0

38.1 Features

- **Asynchronous receiver or transmitter when the aWire system is not used for debugging.**
- **One- or two-pin operation supported.**

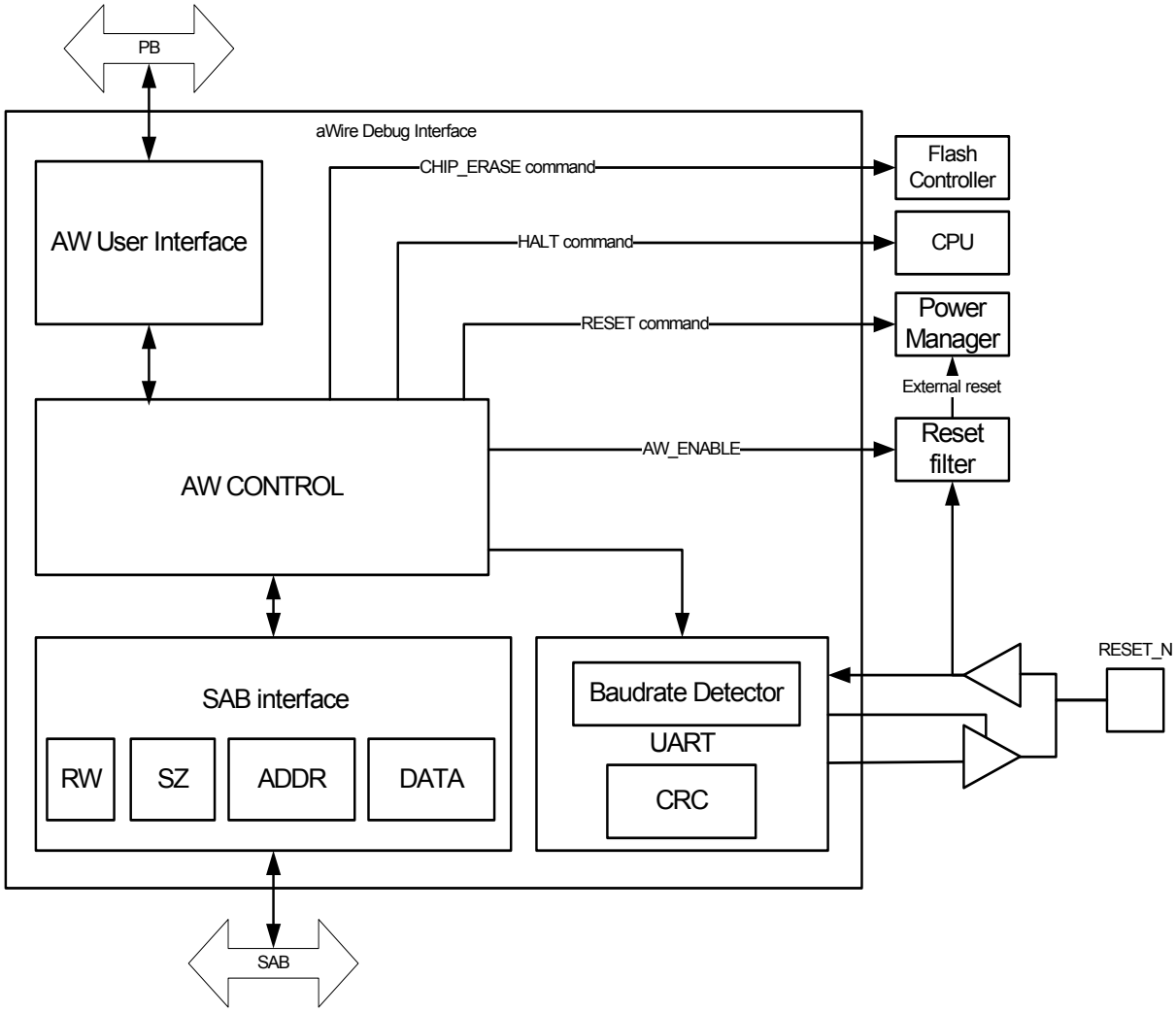
38.2 Overview

If the AW is not used for debugging, the aWire UART can be used by the user to send or receive data with one start bit, eight data bits, no parity bits, and one stop bit. This can be controlled through the aWire UART user interface.

This chapter only describes the aWire UART user interface. For a description of the aWire Debug Interface, please see the Programming and Debugging chapter.

38.3 Block Diagram

Figure 38-1. aWire Debug Interface Block Diagram



38.4 I/O Lines Description

Table 38-1. I/O Lines Description

Name	Description	Type
DATA	aWire data multiplexed with the RESET_N pin.	Input/Output

38.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

38.5.1 I/O Lines

The pin used by AW is multiplexed with the RESET_N pin. The reset functionality is the default function of this pin. To enable the aWire functionality on the RESET_N pin the user must enable the aWire UART user interface.

38.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the aWire UART user interface, the aWire UART user interface will stop functioning and resume operation after the system wakes up from sleep mode.

38.5.3 Clocks

The aWire UART uses the internal 120 MHz RC oscillator (RC120M) as clock source for its operation. When using the aWire UART user interface RC120M must enabled using the Clock Request Register (see Section 38.6.1).

The clock for the aWire UART user interface (CLK_AW) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the aWire UART user interface before disabling the clock, to avoid freezing the aWire UART user interface in an undefined state.

38.5.4 Interrupts

The aWire UART user interface interrupt request line is connected to the interrupt controller. Using the aWire UART user interface interrupt requires the interrupt controller to be programmed first.

38.5.5 Debug Operation

If the AW is used for debugging the aWire UART user interface will not be usable.

When an external debugger forces the CPU into debug mode, the aWire UART user interface continues normal operation. If the aWire UART user interface is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

38.6 Functional Description

The aWire UART user interface can be used as a spare Asynchronous Receiver or Transmitter when AW is not used for debugging.

38.6.1 How to Initialize The Module

To initialize the aWire UART user interface the user must first enable the clock by writing a one to the Clock Enable bit in the Clock Request Register (CLKR.CLKEN) and wait for the Clock Enable bit in the Status Register (SR.CENABLED) to be set. After doing this either receive, transmit or receive with resync must be selected by writing the corresponding value into the Mode field of the Control (CTRL.MODE) Register. Due to the RC120M being asynchronous with the system clock values must be allowed to propagate in the system. During this time the aWire master will set the Busy bit in the Status Register (SR.BUSY).

After the SR.BUSY bit is cleared the Baud Rate field in the Baud Rate Register (BRR.BR) can be written with the wanted baudrate (f_{br}) according to the following formula (f_{aw} is the RC120M clock frequency):

$$f_{br} = \frac{8 f_{aw}}{BR}$$

After this operation the user must wait until the SR.BUSY is cleared. The interface is now ready to be used.

38.6.2 Basic Asynchronous Receiver Operation

The aWire UART user interface must be initialized according to the sequence above, but the CTRL.MODE field must be written to one (Receive mode).

When a data byte arrives the aWire UART user interface will indicate this by setting the Data Ready Interrupt bit in the Status Register (SR.DREADYINT). The user must read the Data in the Receive Holding Register (RHR.RXDATA) and clear the Interrupt bit by writing a one to the Data Ready Interrupt Clear bit in the Status Clear Register (SCR.DREADYINT). The interface is now ready to receive another byte.

38.6.3 Basic Asynchronous Transmitter Operation

The aWire UART user interface must be initialized according to the sequence above, but the CTRL.MODE field must be written to two (Transmit mode).

To transmit a data byte the user must write the data to the Transmit Holding Register (THE.TXDATA). Before the next byte can be written the SR.BUSY must be cleared.

38.6.4 Basic Asynchronous Receiver with Resynchronization

By writing three into CTRL.MODE the aWire UART user interface will assume that the first byte it receives is a sync byte (0x55) and set BRR.BR according to this. All subsequent transfers will assume this baudrate, unless BRR.BR is rewritten by the user.

To make the aWire UART user interface accept a new sync resynchronization the aWire UART user interface must be disabled by writing zero to CTRL.MODE and then reenable the interface.

38.6.5 Overrun

In Receive mode an overrun can occur if the user has not read the previous received data from the RHR.RXDATA when the newest data should be placed there. Such a condition is flagged by setting the Overrun bit in the Status Register (SR.OVERRUN). If SR.OVERRUN is set the newest data received is placed in RHR.RXDATA and the data that was there before is overwritten.

38.6.6 Interrupts

To make the CPU able to do other things while waiting for the aWire UART user interface to finish its operations the aWire UART user interface supports generating interrupts. All status bits in the Status Register can be used as interrupt sources, except the SR.BUSY and SR.CENABLED bits.

To enable an interrupt the user must write a one to the corresponding bit in the Interrupt Enable Register (IER). Upon the next zero to one transition of this SR bit the aWire UART user interface will flag this interrupt to the CPU. To clear the interrupt the user must write a one to the corresponding bit in the Status Clear Register (SCR).

Interrupts can be disabled by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The interrupt Mask Register (IMR) can be read to check if an interrupt is enabled or disabled.

38.6.7 Using the Peripheral DMA Controller

To relieve the CPU of data transfers the aWire UART user interface support using the Peripheral DMA controller.

To transmit using the Peripheral DMA Controller do the following:

1. Setup the aWire UART user interface in transmit mode.
2. Setup the Peripheral DMA Controller with buffer address and length, use byte as transfer size.
3. Enable the Peripheral DMA Controller.
4. Wait until the Peripheral DMA Controller is done.

To receive using the Peripheral DMA Controller do the following:

1. Setup the aWire UART user interface in receive mode
2. Setup the Peripheral DMA Controller with buffer address and length, use byte as transfer size.
3. Enable the Peripheral DMA Controller.
4. Wait until the Peripheral DMA Controller is ready.

38.7 User Interface

Table 38-2. aWire UART user interface Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CTRL	Read/Write	0x00000000
0x04	Status Register	SR	Read-only	0x00000000
0x08	Status Clear Register	SCR	Write-only	-
0x0C	Interrupt Enable Register	IER	Write-only	-
0x10	Interrupt Disable Register	IDR	Write-only	-
0x14	Interrupt Mask Register	IMR	Read-only	0x00000000
0x18	Receive Holding Register	RHR	Read-only	0x00000000
0x1C	Transmit Holding Register	THR	Read/Write	0x00000000
0x20	Baud Rate Register	BRR	Read/Write	0x00000000
0x24	Version Register	VERSION	Read-only	-(1)
0x28	Clock Request Register	CLKR	Read/Write	0x00000000

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

38.7.1 Control Register

Name: CTRL

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

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

- **MODE: aWire UART user interface mode**

Table 38-3. aWire UART user interface Modes

MODE	Mode Description
0	Disabled
1	Receive
2	Transmit
3	Receive with resync.

38.7.2 Status Register

Name: SR

Access Type: Read-only

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	TRMIS	-	-	OVERRUN	DREADYINT	READYINT
7	6	5	4	3	2	1	0
-	-	-	-	-	CENABLED	-	BUSY

- **TRMIS: Transmit Mismatch**
  0: No transfers mismatches.
  1: The transceiver was active when receiving.
  This bit is set when the transceiver is active when receiving.
  This bit is cleared when corresponding bit in SCR is written to one.
- **OVERRUN: Data Overrun**
  0: No data overwritten in RHR.
  1: Data in RHR has been overwritten before it has been read.
  This bit is set when data in RHR is overwritten before it has been read.
  This bit is cleared when corresponding bit in SCR is written to one.
- **DREADYINT: Data Ready Interrupt**
  0: No new data in the RHR.
  1: New data received and placed in the RHR.
  This bit is set when new data is received and placed in the RHR.
  This bit is cleared when corresponding bit in SCR is written to one.
- **READYINT: Ready Interrupt**
  0: The interface has not generated an ready interrupt.
  1: The interface has had a transition from busy to not busy.
  This bit is set when the interface has transition from busy to not busy.
  This bit is cleared when corresponding bit in SCR is written to one.

- **CENABLED: Clock Enabled**
  0: The aWire clock is not enabled.
  1: The aWire clock is enabled.

This bit is set when the clock is disabled.

This bit is cleared when the clock is enabled.

- **BUSY: Synchronizer Busy**

  0: The asynchronous interface is ready to accept more data.

  1: The asynchronous interface is busy and will block writes to CTRL, BRR, and THR.

  This bit is set when the asynchronous interface becomes busy.

  This bit is cleared when the asynchronous interface becomes ready.

38.7.3 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	TRMIS	-	-	OVERRUN	DREADYINT	READYINT
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

38.7.4 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	TRMIS	-	-	OVERRUN	DREADYINT	READYINT
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

38.7.5 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	TRMIS	-	-	OVERRUN	DREADYINT	READYINT
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

38.7.6 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	TRMIS	-	-	OVERRUN	DREADYINT	READYINT
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

38.7.7 Receive Holding Register

Name: RHR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: Received Data**
  The last byte received.

38.7.8 Transmit Holding Register

Name: THR

Access Type: Read/Write

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
TXDATA							

- **TXDATA: Transmit Data**
  The data to send.

38.7.9 Baud Rate Register

Name: BRR

Access Type: Read/Write

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
BR[15:8]							

7	6	5	4	3	2	1	0
BR[7:0]							

- **BR: Baud Rate**

  The baud rate (f_{br}) of the transmission, calculated using the following formula (f_{aw} is the RC120M frequency):

$$f_{br} = \frac{8 f_{aw}}{BR}$$

  BR should not be set to a value smaller than 32.
  Writing a value to this field will update the baud rate of the transmission.
  Reading this field will give the current baud rate of the transmission.

38.7.10 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x24

Reset Value: 0x00000200

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	VERSION[11:8]			

7	6	5	4	3	2	1	0
VERSION[7:0]							

- **VERSION: Version Number**
  Version number of the module. No functionality associated.

38.7.11 Clock Request Register

Name: CLKR

Access Type: Read/Write

Offset: 0x28

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	CLKEN

- **CLKEN: Clock Enable**
  0: The aWire clock is disabled.
  1: The aWire clock is enabled.
  Writing a zero to this bit will disable the aWire clock.
  Writing a one to this bit will enable the aWire clock.

38.8 Module Configuration

The specific configuration for each aWire instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 38-4. Module clock name

Module name	Clock name	Description
aWire	CLK_AW	Peripheral Bus clock from the PBA clock domain

Table 38-5. Register Reset Values

Register	Reset Value
VERSION	0x00000230

39. Programming and Debugging

39.1 Overview

The AT32UC3C supports programming and debugging through two interfaces, JTAG or aWire™. JTAG is an industry standard interface and allows boundary scan for PCB testing, as well as daisy-chaining of multiple devices on the PCB. aWire is an Atmel proprietary protocol which offers higher throughput and robust communication, and does not require application pins to be reserved. Either interface provides access to the internal Service Access Bus (SAB), which offers a bridge to the High Speed Bus, giving access to memories and peripherals in the device. By using this bridge to the bus system, the flash and fuses can thus be programmed by accessing the Flash Controller in the same manner as the CPU.

The SAB also provides access to the Nexus-compliant On-Chip Debug (OCD) system in the device, which gives the user non-intrusive run-time control of the program execution. Additionally, trace information can be output on the Auxiliary (AUX) debug port or buffered in internal RAM for later retrieval by JTAG or aWire.

39.2 Service Access Bus

The AVR32 architecture offers a common interface for access to On-Chip Debug, programming, and test functions. These are mapped on a common bus called the Service Access Bus (SAB), which is linked to the JTAG and aWire port through a bus master module, which also handles synchronization between the debugger and SAB clocks.

When accessing the SAB through the debugger there are no limitations on debugger frequency compared to chip frequency, although there must be an active system clock in order for the SAB accesses to complete. If the system clock is switched off in sleep mode, activity on the debugger will restart the system clock automatically, without waking the device from sleep. Debuggers may optimize the transfer rate by adjusting the frequency in relation to the system clock. This ratio can be measured with debug protocol specific instructions.

The Service Access Bus uses 36 address bits to address memory or registers in any of the slaves on the bus. The bus supports sized accesses of bytes (8 bits), halfwords (16 bits), or words (32 bits). All accesses must be aligned to the size of the access, i.e. halfword accesses must have the lowest address bit cleared, and word accesses must have the two lowest address bits cleared.

39.2.1 SAB address map

The Service Access Bus (SAB) gives the user access to the internal address space and other features through a 36 bits address space. The 4 MSBs identify the slave number, while the 32 LSBs are decoded within the slave's address space. The SAB slaves are shown in Table 39-1 on page 1195.

Table 39-1. SAB Slaves, addresses and descriptions.

Slave	Address [35:32]	Description
Unallocated	0x0	Intentionally unallocated
OCD	0x1	OCD registers
HSB	0x4	HSB memory space, as seen by the CPU

Table 39-1. SAB Slaves, addresses and descriptions.

Slave	Address [35:32]	Description
HSB	0x5	Alternative mapping for HSB space, for compatibility with other 32-bit AVR devices.
Memory Service Unit	0x6	Memory Service Unit registers
Reserved	Other	Unused

39.2.2 SAB security restrictions

The Service Access bus can be restricted by internal security measures. A short description of the security measures are found in the table below.

39.2.2.1 Security measure and control location

A security measure is a mechanism to either block or allow SAB access to a certain address or address range. A security measure is enabled or disabled by one or several control signals. This is called the control location for the security measure.

These security measures can be used to prevent an end user from reading out the code programmed in the flash, for instance.

Table 39-2. SAB Security Measures

Security Measure	Control Location	Description
Security bit	FLASHC security bit set	Programming and debugging not possible, very restricted access.
User code programming	FLASHC UPROT + security bit set	Restricts all access except parts of the flash and the flash controller for programming user code. Debugging is not possible unless an OS running from the secure part of the flash supports it.

Below follows a more in depth description of what locations are accessible when the security measures are active.

Table 39-3. Security Bit SAB Restrictions

Name	Address start	Address end	Access
OCD DCCPU, OCD DCEMU, OCD DCSR	0x100000110	0x100000118	Read/Write
User page	0x580800000	0x581000000	Read
Other accesses	-	-	Blocked

Table 39-4. User Code Programming SAB Restrictions

Name	Address start	Address end	Access
OCD DCCPU, OCD DCEMU, OCD DCSR	0x100000110	0x100000118	Read/Write
User page	0x580800000	0x581000000	Read
FLASHCDW PB interface	0x5FFFE0000	0x5FFFE0400	Read/Write
FLASH pages outside BOOTPROT	0x580000000 + BOOTPROT size	0x580000000 + Flash size	Read/Write
Other accesses	-	-	Blocked

39.3 On-Chip Debug

Rev: 2.0.0.0

39.3.1 Features

- **Debug interface in compliance with IEEE-ISTO 5001-2003 (Nexus 2.0) Class 2+**
- **JTAG or aWire access to all on-chip debug functions**
- **Advanced Program, Data, Ownership, and Watchpoint trace supported**
- **NanoTrace aWire- or JTAG-based trace access**
- **Auxiliary port for high-speed trace information**
- **Hardware support for 6 Program and 2 Data breakpoints**
- **Unlimited number of software breakpoints supported**
- **Automatic CRC check of memory regions**

39.3.2 Overview

Debugging on the AT32UC3C is facilitated by a powerful On-Chip Debug (OCD) system. The user accesses this through an external debug tool which connects to the JTAG or aWire port and the Auxiliary (AUX) port if implemented. The AUX port is primarily used for trace functions, and an aWire- or JTAG-based debugger is sufficient for basic debugging.

The debug system is based on the Nexus 2.0 standard, class 2+, which includes:

- Basic run-time control
- Program breakpoints
- Data breakpoints
- Program trace
- Ownership trace
- Data trace

In addition to the mandatory Nexus debug features, the AT32UC3C implements several useful OCD features, such as:

- Debug Communication Channel between CPU and debugger
- Run-time PC monitoring
- CRC checking
- NanoTrace
- Software Quality Assurance (SQA) support

The OCD features are controlled by OCD registers, which can be accessed by the debugger, for instance when the NEXUS_ACCESS JTAG instruction is loaded. The CPU can also access OCD registers directly using mtdr/mfdr instructions in any privileged mode. The OCD registers are implemented based on the recommendations in the Nexus 2.0 standard, and are detailed in the AVR32UC Technical Reference Manual.

39.3.3 I/O Lines Description

The OCD AUX trace port contains a number of pins, as shown in Table 39-5 on page 1199. These are multiplexed with I/O Controller lines, and must explicitly be enabled by writing OCD registers before the debug session starts. The AUX port is mapped to two different locations,

selectable by OCD Registers, minimizing the chance that the AUX port will need to be shared with an application.

Table 39-5. Auxiliary Port Signals

Pin Name	Pin Description	Direction	Active Level	Type
MCKO	Trace data output clock	Output		Digital
MDO[5:0]	Trace data output	Output		Digital
MSEO[1:0]	Trace frame control	Output		Digital
EVTI_N	Event In	Input	Low	Digital
EVTO_N	Event Out	Output	Low	Digital

39.3.4 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

39.3.4.1 Power Management

The OCD clock operates independently of the CPU clock. If enabled in the Power Manager, the OCD clock (CLK_OCD) will continue running even if the CPU enters a sleep mode that disables the CPU clock.

39.3.4.2 Clocks

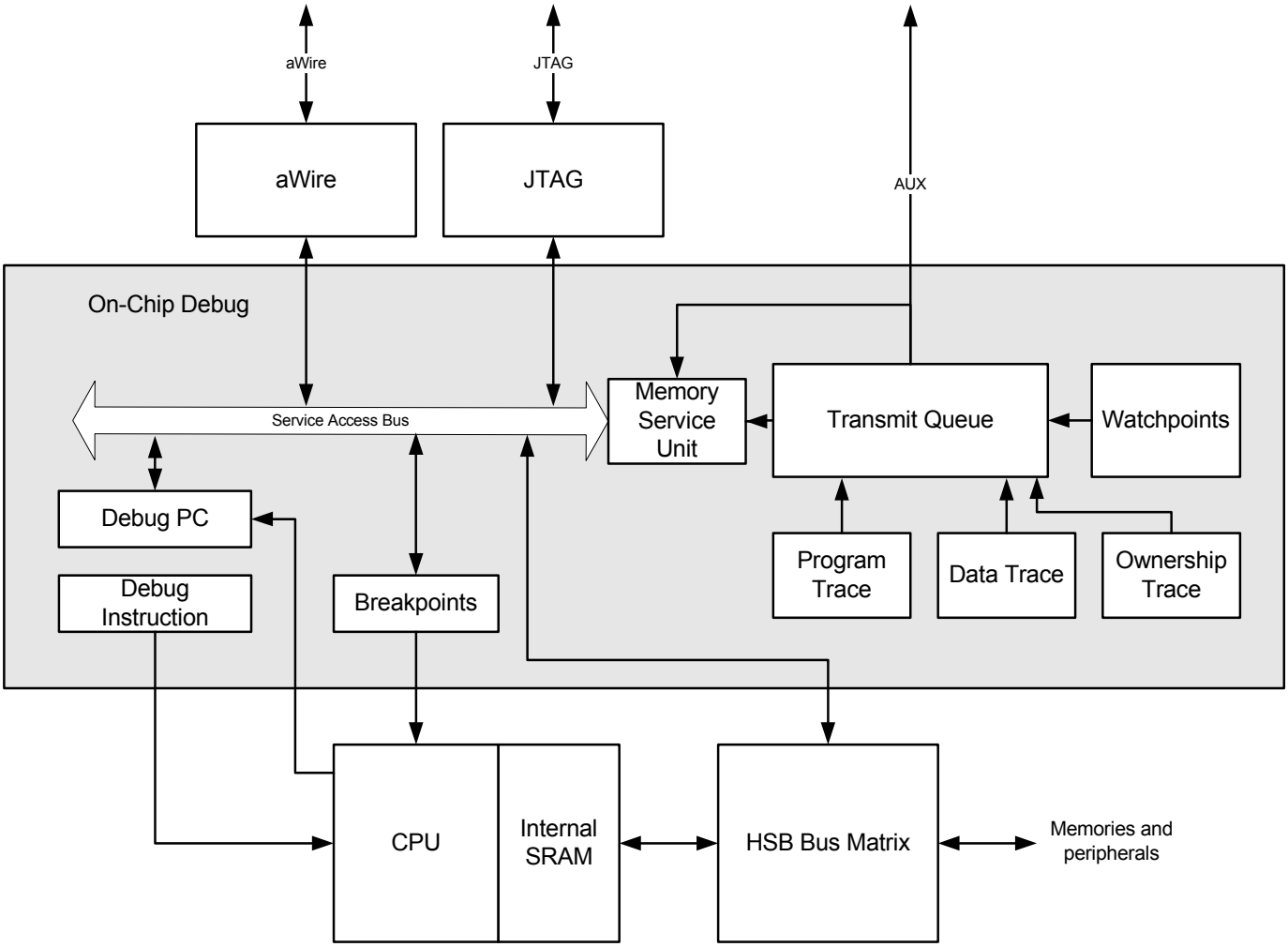
The OCD has a clock (CLK_OCD) running synchronously with the CPU clock. This clock is generated by the Power Manager. The clock is enabled at reset, and can be disabled by writing to the Power Manager.

39.3.4.3 Interrupt

The OCD system interrupt request lines are connected to the interrupt controller. Using the OCD interrupts requires the interrupt controller to be programmed first.

39.3.5 Block Diagram

Figure 39-1. On-Chip Debug Block Diagram



39.3.6 SAB-based Debug Features

A debugger can control all OCD features by writing OCD registers over the SAB interface. Many of these do not depend on output on the AUX port, allowing an aWire- or JTAG-based debugger to be used.

A JTAG-based debugger should connect to the device through a standard 10-pin IDC connector as described in the AVR32UC Technical Reference Manual.

An aWire-based debugger should connect to the device through the RESET_N pin.

Figure 39-2. JTAG-based Debugger

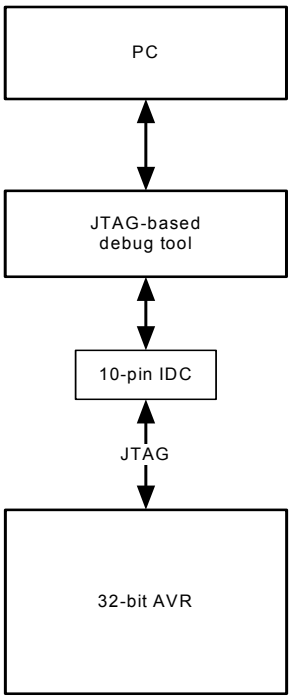
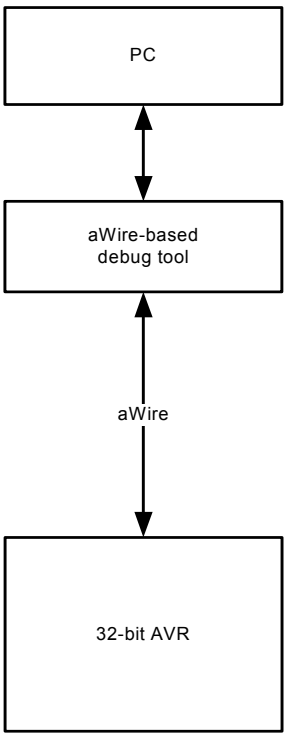


Figure 39-3. aWire-based Debugger



39.3.6.1 Debug Communication Channel

The Debug Communication Channel (DCC) consists of a pair OCD registers with associated handshake logic, accessible to both CPU and debugger. The registers can be used to exchange data between the CPU and the debugmaster, both runtime as well as in debug mode.

The OCD system can generate an interrupt to the CPU when DCCPU is read and when DCEMU is written. This enables the user to build a custum debug protocol using only these registers. The DCCPU and DCEMU registers are available even when the security bit in the flash is active.

For more information refer to the AVR32UC Technical Reference Manual.

39.3.6.2 Breakpoints

One of the most fundamental debug features is the ability to halt the CPU, to examine registers and the state of the system. This is accomplished by breakpoints, of which many types are available:

- Unconditional breakpoints are set by writing OCD registers by the debugger, halting the CPU immediately.
- Program breakpoints halt the CPU when a specific address in the program is executed.
- Data breakpoints halt the CPU when a specific memory address is read or written, allowing variables to be watched.
- Software breakpoints halt the CPU when the breakpoint instruction is executed.

When a breakpoint triggers, the CPU enters debug mode, and the D bit in the status register is set. This is a privileged mode with dedicated return address and return status registers. All privileged instructions are permitted. Debug mode can be entered as either OCD Mode, running instructions from the debugger, or Monitor Mode, running instructions from program memory.

39.3.6.3 OCD Mode

When a breakpoint triggers, the CPU enters OCD mode, and instructions are fetched from the Debug Instruction OCD register. Each time this register is written by the debugger, the instruction is executed, allowing the debugger to execute CPU instructions directly. The debug master can e.g. read out the register file by issuing mtdr instructions to the CPU, writing each register to the Debug Communication Channel OCD registers.

39.3.6.4 Monitor Mode

Since the OCD registers are directly accessible by the CPU, it is possible to build a software-based debugger that runs on the CPU itself. Setting the Monitor Mode bit in the Development Control register causes the CPU to enter Monitor Mode instead of OCD mode when a breakpoint triggers. Monitor Mode is similar to OCD mode, except that instructions are fetched from the debug exception vector in regular program memory, instead of issued by the debug master.

39.3.6.5 Program Counter Monitoring

Normally, the CPU would need to be halted for a debugger to examine the current PC value. However, the AT32UC3C also proves a Debug Program Counter OCD register, where the debugger can continuously read the current PC without affecting the CPU. This allows the debugger to generate a simple statistic of the time spent in various areas of the code, easing code optimization.

39.3.7 Memory Service Unit

The Memory Service Unit (MSU) is a block dedicated to test and debug functionality. It is controlled through a dedicated set of registers addressed through the Service Access Bus.

39.3.7.1 *Cyclic Redundancy Check (CRC)*

The MSU can be used to automatically calculate the CRC of a block of data in memory. The MSU will then read out each word in the specified memory block and report the CRC32-value in an MSU register.

39.3.7.2 *NanoTrace*

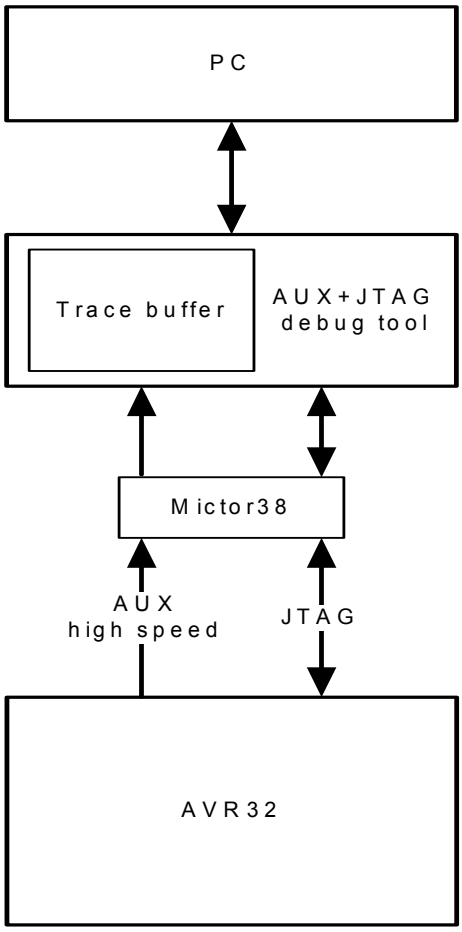
The MSU additionally supports NanoTrace. This is a 32-bit AVR-specific feature, in which trace data is output to memory instead of the AUX port. This allows the trace data to be extracted by the debugger through the SAB, enabling trace features for aWire- or JTAG-based debuggers. The user must write MSU registers to configure the address and size of the memory block to be used for NanoTrace. The NanoTrace buffer can be anywhere in the physical address range, including internal and external RAM, through an EBI, if present. This area may not be used by the application running on the CPU.

39.3.8 AUX-based Debug Features

Utilizing the Auxiliary (AUX) port gives access to a wide range of advanced debug features. Of prime importance are the trace features, which allow an external debugger to receive continuous information on the program execution in the CPU. Additionally, Event In and Event Out pins allow external events to be correlated with the program flow.

Debug tools utilizing the AUX port should connect to the device through a Nexus-compliant Mictor-38 connector, as described in the AVR32UC Technical Reference manual. This connector includes the JTAG signals and the RESET_N pin, giving full access to the programming and debug features in the device.

Figure 39-4. AUX+JTAG Based Debugger



39.3.8.1 *Trace Operation*

Trace features are enabled by writing OCD registers by the debugger. The OCD extracts the trace information from the CPU, compresses this information and formats it into variable-length messages according to the Nexus standard. The messages are buffered in a 16-frame transmit queue, and are output on the AUX port one frame at a time.

The trace features can be configured to be very selective, to reduce the bandwidth on the AUX port. In case the transmit queue overflows, error messages are produced to indicate loss of data. The transmit queue module can optionally be configured to halt the CPU when an overflow occurs, to prevent the loss of messages, at the expense of longer run-time for the program.

39.3.8.2 *Program Trace*

Program trace allows the debugger to continuously monitor the program execution in the CPU. Program trace messages are generated for every branch in the program, and contains compressed information, which allows the debugger to correlate the message with the source code to identify the branch instruction and target address.

39.3.8.3 *Data Trace*

Data trace outputs a message every time a specific location is read or written. The message contains information about the type (read/write) and size of the access, as well as the address and data of the accessed location. The AT32UC3C contains two data trace channels, each of

which are controlled by a pair of OCD registers which determine the range of addresses (or single address) which should produce data trace messages.

39.3.8.4 *Ownership Trace*

Program and data trace operate on virtual addresses. In cases where an operating system runs several processes in overlapping virtual memory segments, the Ownership Trace feature can be used to identify the process switch. When the O/S activates a process, it will write the process ID number to an OCD register, which produces an Ownership Trace Message, allowing the debugger to switch context for the subsequent program and data trace messages. As the use of this feature depends on the software running on the CPU, it can also be used to extract other types of information from the system.

39.3.8.5 *Watchpoint Messages*

The breakpoint modules normally used to generate program and data breakpoints can also be used to generate Watchpoint messages, allowing a debugger to monitor program and data events without halting the CPU. Watchpoints can be enabled independently of breakpoints, so a breakpoint module can optionally halt the CPU when the trigger condition occurs. Data trace modules can also be configured to produce watchpoint messages instead of regular data trace messages.

39.3.8.6 *Event In and Event Out Pins*

The AUX port also contains an Event In pin (EVTI_N) and an Event Out pin (EVTO_N). EVTI_N can be used to trigger a breakpoint when an external event occurs. It can also be used to trigger specific program and data trace synchronization messages, allowing an external event to be correlated to the program flow.

When the CPU enters debug mode, a Debug Status message is transmitted on the trace port. All trace messages can be timestamped when they are received by the debug tool. However, due to the latency of the transmit queue buffering, the timestamp will not be 100% accurate. To improve this, EVTO_N can toggle every time a message is inserted into the transmit queue, allowing trace messages to be timestamped precisely. EVTO_N can also toggle when a breakpoint module triggers, or when the CPU enters debug mode, for any reason. This can be used to measure precisely when the respective internal event occurs.

39.3.8.7 *Software Quality Analysis (SQA)*

Software Quality Analysis (SQA) deals with two important issues regarding embedded software development. *Code coverage* involves identifying untested parts of the embedded code, to improve test procedures and thus the quality of the released software. *Performance analysis* allows the developer to precisely quantify the time spent in various parts of the code, allowing bottlenecks to be identified and optimized.

Program trace must be used to accomplish these tasks without instrumenting (altering) the code to be examined. However, traditional program trace cannot reconstruct the current PC value without correlating the trace information with the source code, which cannot be done on-the-fly. This limits program trace to a relatively short time segment, determined by the size of the trace buffer in the debug tool.

The OCD system in AT32UC3C extends program trace with SQA capabilities, allowing the debug tool to reconstruct the PC value on-the-fly. Code coverage and performance analysis can thus be reported for an unlimited execution sequence.

39.4 JTAG and Boundary-scan (JTAG)

Rev: 2.3.0.4

39.4.1 Features

- **IEEE1149.1 compliant JTAG Interface**
- **Boundary-scan Chain for board-level testing**
- **Direct memory access and programming capabilities through JTAG Interface**

39.4.2 Overview

The JTAG Interface offers a four pin programming and debug solution, including boundary-scan support for board-level testing.

Figure 39-5 on page 1207 shows how the JTAG is connected in an 32-bit AVR device. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (shift register) between the TDI-input and TDO-output.

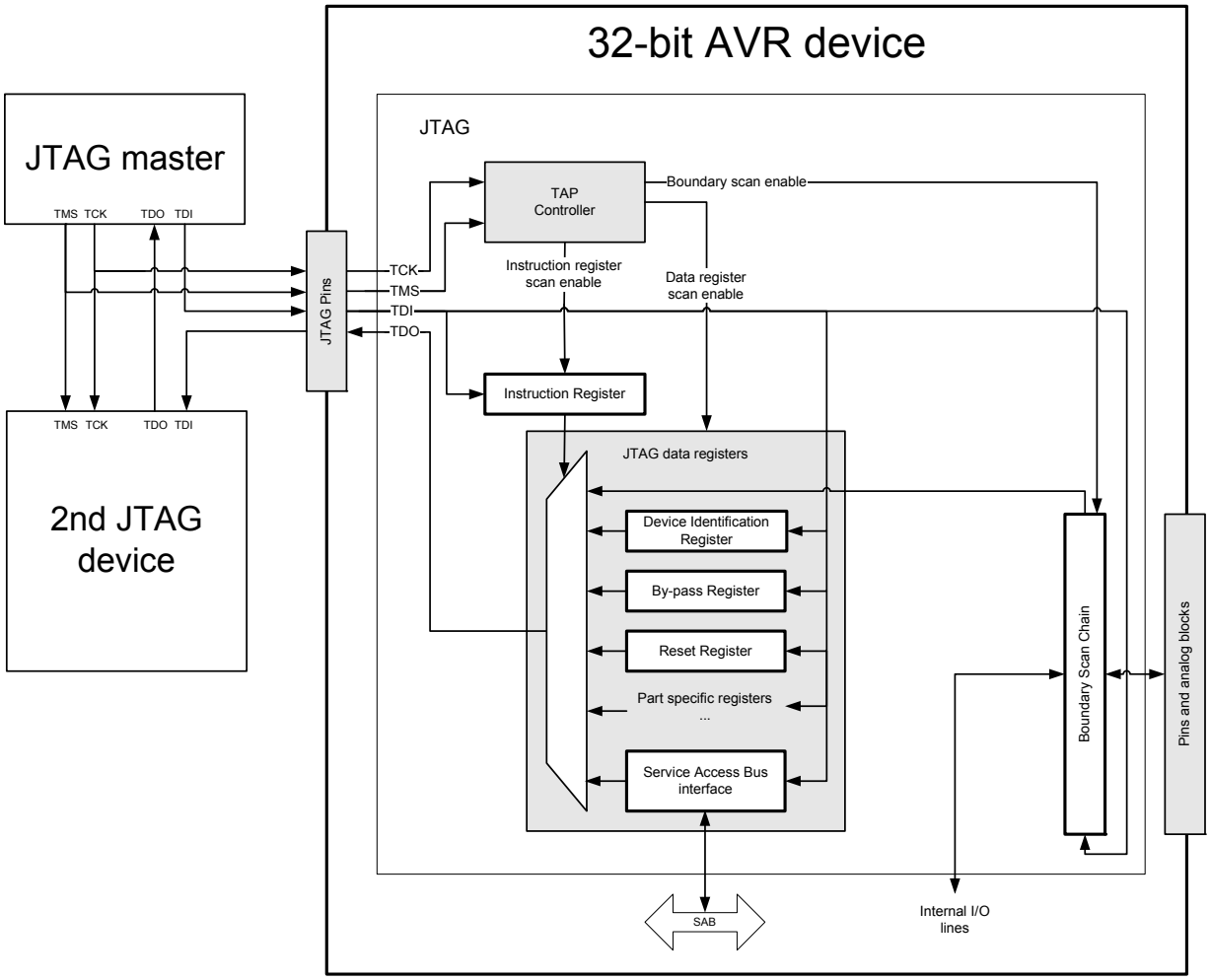
The Instruction Register holds JTAG instructions controlling the behavior of a Data Register. The Device Identification Register, Bypass Register, and the boundary-scan chain are the Data Registers used for board-level testing. The Reset Register can be used to keep the device reset during test or programming.

The Service Access Bus (SAB) interface contains address and data registers for the Service Access Bus, which gives access to On-Chip Debug, programming, and other functions in the device. The SAB offers several modes of access to the address and data registers, as described in Section 39.4.11.

Section 39.5 lists the supported JTAG instructions, with references to the description in this document.

39.4.3 Block Diagram

Figure 39-5. JTAG and Boundary-scan Access



39.4.4 I/O Lines Description

Table 39-6. I/O Line Description

Pin Name	Pin Description	Type	Active Level
RESET_N	External reset pin. Used when enabling and disabling the JTAG.	Input	Low
TCK	Test Clock Input. Fully asynchronous to system clock frequency.	Input	
TMS	Test Mode Select, sampled on rising TCK.	Input	
TDI	Test Data In, sampled on rising TCK.	Input	
TDO	Test Data Out, driven on falling TCK.	Output	

39.4.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

39.4.5.1 I/O Lines

The TMS, TDI, TDO, and TCK pins are multiplexed with I/O lines. When the JTAG is used the associated pins must be enabled. To enable the JTAG pins, refer to Section 39.4.7.

While using the multiplexed JTAG lines all normal peripheral activity on these lines is disabled. The user must make sure that no external peripheral is blocking the JTAG lines while debugging.

39.4.5.2 Power Management

When an instruction that accesses the SAB is loaded in the instruction register, before entering a sleep mode, the system clocks are not switched off to allow debugging in sleep modes. This can lead to a program behaving differently when debugging.

39.4.5.3 Clocks

The JTAG Interface uses the external TCK pin as clock source. This clock must be provided by the JTAG master.

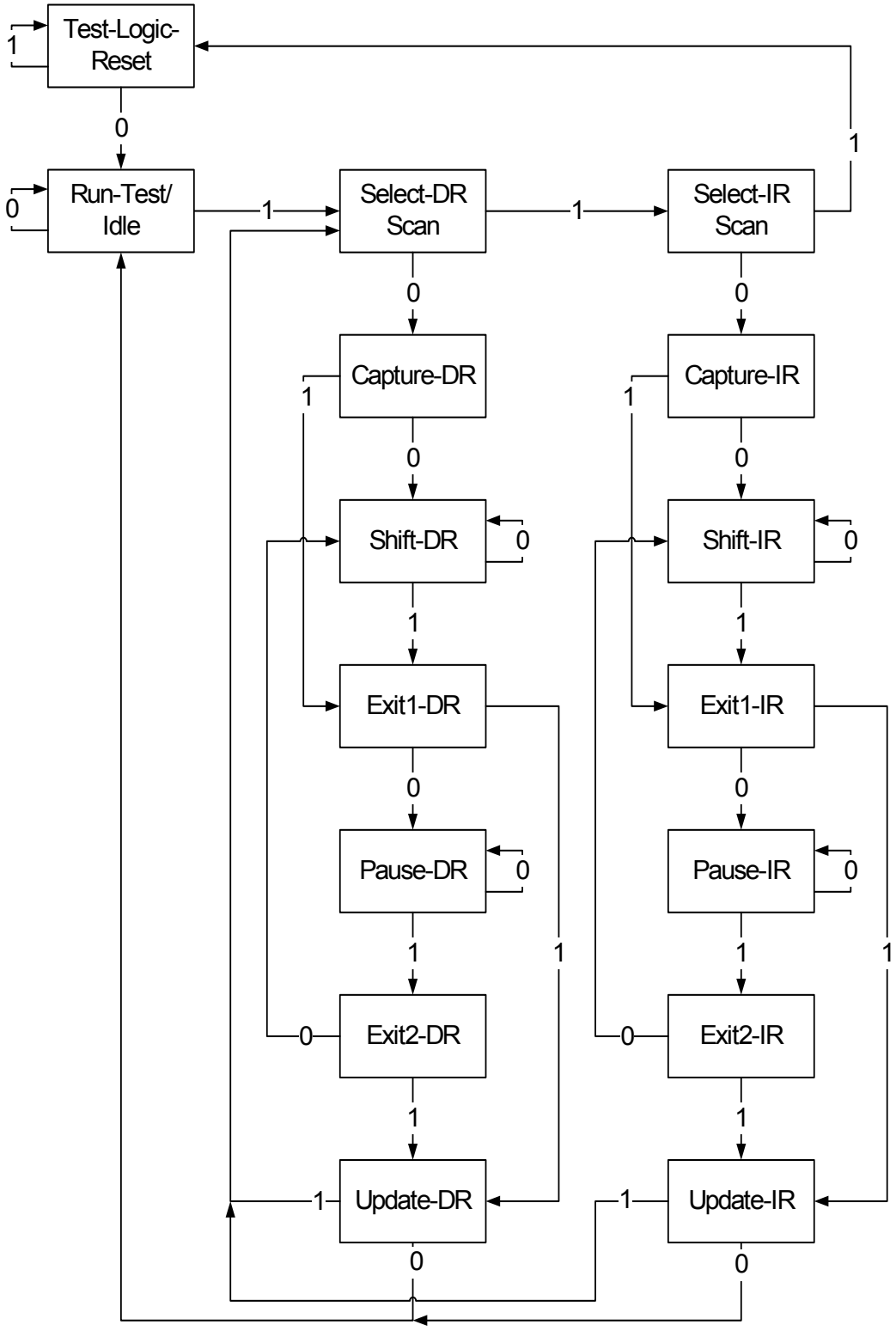
Instructions that use the SAB bus requires the internal main clock to be running.

39.4.6 JTAG Interface

The JTAG Interface is accessed through the dedicated JTAG pins shown in Table 39-6 on page 1207. The TMS control line navigates the TAP controller, as shown in Figure 39-6 on page 1209. The TAP controller manages the serial access to the JTAG Instruction and Data registers. Data is scanned into the selected instruction or data register on TDI, and out of the register on TDO, in the Shift-IR and Shift-DR states, respectively. The LSB is shifted in and out first. TDO is high-Z in other states than Shift-IR and Shift-DR.

The device implements a 5-bit Instruction Register (IR). A number of public JTAG instructions defined by the JTAG standard are supported, as described in Section 39.5.2, as well as a number of 32-bit AVR-specific private JTAG instructions described in Section 39.5.3. Each instruction selects a specific data register for the Shift-DR path, as described for each instruction.

Figure 39-6. TAP Controller State Diagram



39.4.7 How to Initialize the Module

To enable the JTAG pins the TCK pin must be held low while the RESET_N pin is released. After enabling the JTAG interface the halt bit is set automatically to prevent the system from running code after the interface is enabled. To make the CPU run again set halt to zero using the HALT command..

JTAG operation when RESET_N is pulled low is not possible.

Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for 5 TCK clock periods. This sequence should always be applied at the start of a JTAG session and after enabling the JTAG pins to bring the TAP Controller into a defined state before applying JTAG commands. Applying a 0 on TMS for 1 TCK period brings the TAP Controller to the Run-Test/Idle state, which is the starting point for JTAG operations.

39.4.8 How to disable the module

To disable the JTAG pins the TCK pin must be held high while RESET_N pin is released.

39.4.9 Typical Sequence

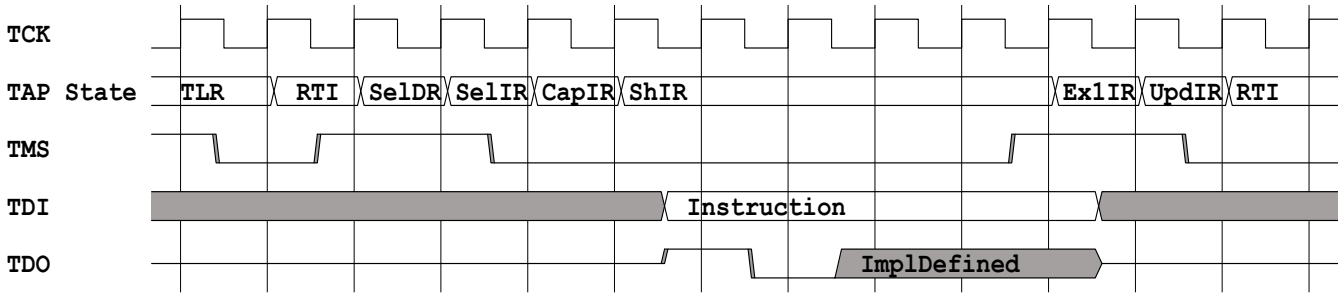
Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG Interface follows.

39.4.9.1 Scanning in JTAG Instruction

At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register (Shift-IR) state. While in this state, shift the 5 bits of the JTAG instructions into the JTAG instruction register from the TDI input at the rising edge of TCK. During shifting, the JTAG outputs status bits on TDO, refer to Section 39.5 for a description of these. The TMS input must be held low during input of the 4 LSBs in order to remain in the Shift-IR state. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the shift register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.

Figure 39-7. Scanning in JTAG Instruction



39.4.9.2 Scanning in/out Data

At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register (Shift-DR) state. While in this state, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge

of TCK. In order to remain in the Shift-DR state, the TMS input must be held low. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the TDO pin.

Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers.

39.4.10 Boundary-scan

The boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections. At system level, all ICs having JTAG capabilities are connected serially by the TDI/TDO signals to form a long shift register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, boundary-scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the 4 TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST can be used for testing the Printed Circuit Board. Initial scanning of the data register path will show the ID-code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the 32-bit AVR device in reset during test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an undetermined state when exiting the test mode. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The device can be set in the reset state either by pulling the external RESETn pin low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the external pins during normal operation of the part.

When using the JTAG Interface for boundary-scan, the JTAG TCK clock is independent of the internal chip clock. The internal chip clock is not required to run during boundary-scan operations.

NOTE: For pins connected to 5V lines care should be taken to not drive the pins to a logic one using boundary-scan, as this will create a current flowing from the 3,3V driver to the 5V pull-up on the line. Optionally a series resistor can be added between the line and the pin to reduce the current.

Details about the boundary-scan chain can be found in the BSDL file for the device. This can be found on the Atmel website.

39.4.11 Service Access Bus

The AVR32 architecture offers a common interface for access to On-Chip Debug, programming, and test functions. These are mapped on a common bus called the Service Access Bus (SAB),

which is linked to the JTAG through a bus master module, which also handles synchronization between the TCK and SAB clocks.

For more information about the SAB and a list of SAB slaves see the Service Access Bus chapter.

39.4.11.1 SAB Address Mode

The MEMORY_SIZED_ACCESS instruction allows a sized read or write to any 36-bit address on the bus. MEMORY_WORD_ACCESS is a shorthand instruction for 32-bit accesses to any 36-bit address, while the NEXUS_ACCESS instruction is a Nexus-compliant shorthand instruction for accessing the 32-bit OCD registers in the 7-bit address space reserved for these. These instructions require two passes through the Shift-DR TAP state: one for the address and control information, and one for data.

39.4.11.2 Block Transfer

To increase the transfer rate, consecutive memory accesses can be accomplished by the MEMORY_BLOCK_ACCESS instruction, which only requires a single pass through Shift-DR for data transfer only. The address is automatically incremented according to the size of the last SAB transfer.

39.4.11.3 Canceling a SAB Access

It is possible to abort an ongoing SAB access by the CANCEL_ACCESS instruction, to avoid hanging the bus due to an extremely slow slave.

39.4.11.4 Busy Reporting

As the time taken to perform an access may vary depending on system activity and current chip frequency, all the SAB access JTAG instructions can return a busy indicator. This indicates whether a delay needs to be inserted, or an operation needs to be repeated in order to be successful. If a new access is requested while the SAB is busy, the request is ignored.

The SAB becomes busy when:

- Entering Update-DR in the address phase of any read operation, e.g., after scanning in a NEXUS_ACCESS address with the read bit set.
- Entering Update-DR in the data phase of any write operation, e.g., after scanning in data for a NEXUS_ACCESS write.
- Entering Update-DR during a MEMORY_BLOCK_ACCESS.
- Entering Update-DR after scanning in a counter value for SYNC.
- Entering Update-IR after scanning in a MEMORY_BLOCK_ACCESS if the previous access was a read and data was scanned after scanning the address.

The SAB becomes ready again when:

- A read or write operation completes.
- A SYNC countdown completed.
- A operation is cancelled by the CANCEL_ACCESS instruction.

What to do if the busy bit is set:

- During Shift-IR: The new instruction is selected, but the previous operation has not yet completed and will continue (unless the new instruction is CANCEL_ACCESS). You may

continue shifting the same instruction until the busy bit clears, or start shifting data. If shifting data, you must be prepared that the data shift may also report busy.

- During Shift-DR of an address: The new address is ignored. The SAB stays in address mode, so no data must be shifted. Repeat the address until the busy bit clears.
- During Shift-DR of read data: The read data is invalid. The SAB stays in data mode. Repeat scanning until the busy bit clears.
- During Shift-DR of write data: The write data is ignored. The SAB stays in data mode. Repeat scanning until the busy bit clears.

39.4.11.5 Error Reporting

The Service Access Bus may not be able to complete all accesses as requested. This may be because the address is invalid, the addressed area is read-only or cannot handle byte/halfword accesses, or because the chip is set in a protected mode where only limited accesses are allowed.

The error bit is updated when an access completes, and is cleared when a new access starts.

What to do if the error bit is set:

- During Shift-IR: The new instruction is selected. The last operation performed using the old instruction did not complete successfully.
- During Shift-DR of an address: The previous operation failed. The new address is accepted. If the read bit is set, a read operation is started.
- During Shift-DR of read data: The read operation failed, and the read data is invalid.
- During Shift-DR of write data: The previous write operation failed. The new data is accepted and a write operation started. This should only occur during block writes or stream writes. No error can occur between scanning a write address and the following write data.
- While polling with CANCEL_ACCESS: The previous access was cancelled. It may or may not have actually completed.
- After power-up: The error bit is set after power up, but there has been no previous SAB instruction so this error can be discarded.

39.4.11.6 Protected Reporting

A protected status may be reported during Shift-IR or Shift-DR. This indicates that the security bit in the Flash Controller is set and that the chip is locked for access, according to Section 39.5.1.

The protected state is reported when:

- The Flash Controller is under reset. This can be due to the AVR_RESET command or the RESET_N line.
- The Flash Controller has not read the security bit from the flash yet (This will take a a few ms). Happens after the Flash Controller reset has been released.
- The security bit in the Flash Controller is set.

What to do if the protected bit is set:

- Release all active AVR_RESET domains, if any.
- Release the RESET_N line.
- Wait a few ms for the security bit to clear. It can be set temporarily due to a reset.

- Perform a CHIP_ERASE to clear the security bit. **NOTE**: This will erase all the contents of the non-volatile memory.

39.5 JTAG Instruction Summary

The implemented JTAG instructions in the 32-bit AVR are shown in the table below.

Table 39-7. JTAG Instruction Summary

Instruction OPCODE	Instruction	Description
0x01	IDCODE	Select the 32-bit Device Identification register as data register.
0x02	SAMPLE_PRELOAD	Take a snapshot of external pin values without affecting system operation.
0x03	EXTEST	Select boundary-scan chain as data register for testing circuitry external to the device.
0x04	INTEST	Select boundary-scan chain for internal testing of the device.
0x06	CLAMP	Bypass device through Bypass register, while driving outputs from boundary-scan register.
0x0C	AVR_RESET	Apply or remove a static reset to the device
0x0F	CHIP_ERASE	Erase the device
0x10	NEXUS_ACCESS	Select the SAB Address and Data registers as data register for the TAP. The registers are accessed in Nexus mode.
0x11	MEMORY_WORD_ACCESS	Select the SAB Address and Data registers as data register for the TAP.
0x12	MEMORY_BLOCK_ACCESS	Select the SAB Data register as data register for the TAP. The address is auto-incremented.
0x13	CANCEL_ACCESS	Cancel an ongoing Nexus or Memory access.
0x14	MEMORY_SERVICE	Select the SAB Address and Data registers as data register for the TAP. The registers are accessed in Memory Service mode.
0x15	MEMORY_SIZED_ACCESS	Select the SAB Address and Data registers as data register for the TAP.
0x17	SYNC	Synchronization counter
0x1C	HALT	Halt the CPU for safe programming.
0x1F	BYPASS	Bypass this device through the bypass register.
Others	N/A	Acts as BYPASS

39.5.1 Security Restrictions

When the security fuse in the Flash is programmed, the following JTAG instructions are restricted:

- NEXUS_ACCESS
- MEMORY_WORD_ACCESS
- MEMORY_BLOCK_ACCESS
- MEMORY_SIZED_ACCESS

For description of what memory locations remain accessible, please refer to the SAB address map.

Full access to these instructions is re-enabled when the security fuse is erased by the CHIP_ERASE JTAG instruction.

Note that the security bit will read as programmed and block these instructions also if the Flash Controller is statically reset.

Other security mechanisms can also restrict these functions. If such mechanisms are present they are listed in the SAB address map section.

39.5.1.1 Notation

Table 39-9 on page 1215 shows bit patterns to be shifted in a format like "**peb01**". Each character corresponds to one bit, and eight bits are grouped together for readability. The least significantbit is always shifted first, and the most significant bit shifted last. The symbols used are shown in Table 39-8.

Table 39-8. Symbol Description

Symbol	Description
0	Constant low value - always reads as zero.
1	Constant high value - always reads as one.
a	An address bit - always scanned with the least significant bit first
b	A busy bit. Reads as one if the SAB was busy, or zero if it was not. See Section 39.4.11.4 for details on how the busy reporting works.
d	A data bit - always scanned with the least significant bit first.
e	An error bit. Reads as one if an error occurred, or zero if not. See Section 39.4.11.5 for details on how the error reporting works.
p	The chip protected bit. Some devices may be set in a protected state where access to chip internals are severely restricted. See the documentation for the specific device for details. On devices without this possibility, this bit always reads as zero.
r	A direction bit. Set to one to request a read, set to zero to request a write.
s	A size bit. The size encoding is described where used.
x	A don't care bit. Any value can be shifted in, and output data should be ignored.

In many cases, it is not required to shift all bits through the data register. Bit patterns are shown using the full width of the shift register, but the suggested or required bits are emphasized using **bold** text. I.e. given the pattern "**aaaaaaar** xxxxxxxx xxxxxxxx xxxxxxxx xx", the shift register is 34 bits, but the test or debug unit may choose to shift only 8 bits "**aaaaaaar**".

The following describes how to interpret the fields in the instruction description tables:

Table 39-9. Instruction Description

Instruction	Description
IR input value	Shows the bit pattern to shift into IR in the Shift-IR state in order to select this instruction. The pattern is show both in binary and in hexadecimal form for convenience. Example: **10000** (0x10)
IR output value	Shows the bit pattern shifted out of IR in the Shift-IR state when this instruction is active. Example: peb01

Table 39-9. Instruction Description (Continued)

Instruction	Description
DR Size	Shows the number of bits in the data register chain when this instruction is active. Example: 34 bits
DR input value	Shows which bit pattern to shift into the data register in the Shift-DR state when this instruction is active. Multiple such lines may exist, e.g., to distinguish between reads and writes. Example: aaaaaaar xxxxxxxx xxxxxxxx xxxxxxxx xx
DR output value	Shows the bit pattern shifted out of the data register in the Shift-DR state when this instruction is active. Multiple such lines may exist, e.g., to distinguish between reads and writes. Example: xx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

39.5.2 Public JTAG Instructions

The JTAG standard defines a number of public JTAG instructions. These instructions are described in the sections below.

39.5.2.1 IDCODE

This instruction selects the 32 bit Device Identification register (DID) as Data Register. The DID register consists of a version number, a device number, and the manufacturer code chosen by JEDEC. This is the default instruction after a JTAG reset. Details about the DID register can be found in the module configuration section at the end of this chapter.

Starting in Run-Test/Idle, the Device Identification register is accessed in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.
6. In Capture-DR: The IDCODE value is latched into the shift register.
7. In Shift-DR: The IDCODE scan chain is shifted by the TCK input.
8. Return to Run-Test/Idle.

Table 39-10. IDCODE Details

Instructions	Details
IR input value	**00001** (0x01)
IR output value	p0001
DR Size	32
DR input value	xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
DR output value	Device Identification Register

39.5.2.2 SAMPLE_PRELOAD

This instruction takes a snap-shot of the input/output pins without affecting the system operation, and pre-loading the scan chain without updating the DR-latch. The boundary-scan chain is selected as Data Register.

Starting in Run-Test/Idle, the Device Identification register is accessed in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.
6. In Capture-DR: The Data on the external pins are sampled into the boundary-scan chain.
7. In Shift-DR: The boundary-scan chain is shifted by the TCK input.
8. Return to Run-Test/Idle.

Table 39-11. SAMPLE_PRELOAD Details

Instructions	Details
IR input value	**00010** (0x02)
IR output value	p0001
DR Size	Depending on boundary-scan chain, see BSDL-file.
DR input value	Depending on boundary-scan chain, see BSDL-file.
DR output value	Depending on boundary-scan chain, see BSDL-file.

39.5.2.3 EXTEST

This instruction selects the boundary-scan chain as Data Register for testing circuitry external to the 32-bit AVR package. The contents of the latched outputs of the boundary-scan chain is driven out as soon as the JTAG IR-register is loaded with the EXTEST instruction.

Starting in Run-Test/Idle, the EXTEST instruction is accessed the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. In Update-IR: The data from the boundary-scan chain is applied to the output pins.
5. Return to Run-Test/Idle.
6. Select the DR Scan path.
7. In Capture-DR: The data on the external pins is sampled into the boundary-scan chain.
8. In Shift-DR: The boundary-scan chain is shifted by the TCK input.
9. In Update-DR: The data from the scan chain is applied to the output pins.
10. Return to Run-Test/Idle.

Table 39-12. EXTEST Details

Instructions	Details
IR input value	**00011** (0x03)
IR output value	p0001
DR Size	Depending on boundary-scan chain, see BSDL-file.
DR input value	Depending on boundary-scan chain, see BSDL-file.
DR output value	Depending on boundary-scan chain, see BSDL-file.

39.5.2.4 INTEST

This instruction selects the boundary-scan chain as Data Register for testing internal logic in the device. The logic inputs are determined by the boundary-scan chain, and the logic outputs are captured by the boundary-scan chain. The device output pins are driven from the boundary-scan chain.

Starting in Run-Test/Idle, the INTEST instruction is accessed the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. In Update-IR: The data from the boundary-scan chain is applied to the internal logic inputs.
5. Return to Run-Test/Idle.
6. Select the DR Scan path.
7. In Capture-DR: The data on the internal logic is sampled into the boundary-scan chain.
8. In Shift-DR: The boundary-scan chain is shifted by the TCK input.
9. In Update-DR: The data from the boundary-scan chain is applied to internal logic inputs.
10. Return to Run-Test/Idle.

Table 39-13. INTEST Details

Instructions	Details
IR input value	**00100** (0x04)
IR output value	p0001
DR Size	Depending on boundary-scan chain, see BSDL-file.
DR input value	Depending on boundary-scan chain, see BSDL-file.
DR output value	Depending on boundary-scan chain, see BSDL-file.

39.5.2.5 CLAMP

This instruction selects the Bypass register as Data Register. The device output pins are driven from the boundary-scan chain.

Starting in Run-Test/Idle, the CLAMP instruction is accessed the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. In Update-IR: The data from the boundary-scan chain is applied to the output pins.
5. Return to Run-Test/Idle.
6. Select the DR Scan path.
7. In Capture-DR: A logic ‘0’ is loaded into the Bypass Register.
8. In Shift-DR: Data is scanned from TDI to TDO through the Bypass register.

9. Return to Run-Test/Idle.

Table 39-14. CLAMP Details

Instructions	Details
IR input value	**00110** (0x06)
IR output value	p0001
DR Size	1
DR input value	x
DR output value	x

39.5.2.6 BYPASS

This instruction selects the 1-bit Bypass Register as Data Register.

Starting in Run-Test/Idle, the CLAMP instruction is accessed the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.
6. In Capture-DR: A logic ‘0’ is loaded into the Bypass Register.
7. In Shift-DR: Data is scanned from TDI to TDO through the Bypass register.
8. Return to Run-Test/Idle.

Table 39-15. BYPASS Details

Instructions	Details
IR input value	**11111** (0x1F)
IR output value	p0001
DR Size	1
DR input value	x
DR output value	x

39.5.3 Private JTAG Instructions

The 32-bit AVR defines a number of private JTAG instructions, not defined by the JTAG standard. Each instruction is briefly described in text, with details following in table form.

39.5.3.1 NEXUS_ACCESS

This instruction allows Nexus-compliant access to the On-Chip Debug registers through the SAB. The 7-bit register index, a read/write control bit, and the 32-bit data is accessed through the JTAG port.

The data register is alternately interpreted by the SAB as an address register and a data register. The SAB starts in address mode after the NEXUS_ACCESS instruction is selected, and toggles between address and data mode each time a data scan completes with the busy bit cleared.

NOTE: The polarity of the direction bit is inverse of the Nexus standard.

Starting in Run-Test/Idle, OCD registers are accessed in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.
6. In Shift-DR: Scan in the direction bit (1=read, 0=write) and the 7-bit address for the OCD register.
7. Go to Update-DR and re-enter Select-DR Scan.
8. In Shift-DR: For a read operation, scan out the contents of the addressed register. For a write operation, scan in the new contents of the register.
9. Return to Run-Test/Idle.

For any operation, the full 7 bits of the address must be provided. For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 39-16. NEXUS_ACCESS Details

Instructions	Details
IR input value	**10000** (0x10)
IR output value	peb01
DR Size	34 bits
DR input value (Address phase)	**aaaaaaar** xxxxxxxx xxxxxxxx xxxxxxxx xx
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xx
DR output value (Address phase)	xx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb
DR output value (Data read phase)	eb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

39.5.3.2 MEMORY_SERVICE

This instruction allows access to registers in an optional Memory Service Unit. The 7-bit register index, a read/write control bit, and the 32-bit data is accessed through the JTAG port.

The data register is alternately interpreted by the SAB as an address register and a data register. The SAB starts in address mode after the MEMORY_SERVICE instruction is selected, and toggles between address and data mode each time a data scan completes with the busy bit cleared.

Starting in Run-Test/Idle, Memory Service registers are accessed in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.
6. In Shift-DR: Scan in the direction bit (1=read, 0=write) and the 7-bit address for the Memory Service register.

7. Go to Update-DR and re-enter Select-DR Scan.
8. In Shift-DR: For a read operation, scan out the contents of the addressed register. For a write operation, scan in the new contents of the register.
9. Return to Run-Test/Idle.

For any operation, the full 7 bits of the address must be provided. For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 39-17. MEMORY_SERVICE Details

Instructions	Details
IR input value	**10100** (0x14)
IR output value	peb01
DR Size	34 bits
DR input value (Address phase)	**aaaaaaar** xxxxxxxx xxxxxxxx xxxxxxxx xx
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xx
DR output value (Address phase)	xx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb
DR output value (Data read phase)	eb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

39.5.3.3 MEMORY_SIZED_ACCESS

This instruction allows access to the entire Service Access Bus data area. Data is accessed through a 36-bit byte index, a 2-bit size, a direction bit, and 8, 16, or 32 bits of data. Not all units mapped on the SAB bus may support all sizes of accesses, e.g., some may only support word accesses.

The data register is alternately interpreted by the SAB as an address register and a data register. The SAB starts in address mode after the MEMORY_SIZED_ACCESS instruction is selected, and toggles between address and data mode each time a data scan completes with the busy bit cleared.

The size field is encoded as i Table 39-18.

Table 39-18. Size Field Semantics

Size field value	Access size	Data alignment
00	Byte (8 bits)	Address modulo 4 : data alignment 0: **dddddddd** xxxxxxxx xxxxxxxx xxxxxxxx 1: xxxxxxxx **dddddddd** xxxxxxxx xxxxxxxx 2: xxxxxxxx xxxxxxxx **dddddddd** xxxxxxxx 3: xxxxxxxx xxxxxxxx xxxxxxxx **dddddddd**
01	Halfword (16 bits)	Address modulo 4 : data alignment 0: **dddddddd dddddddd** xxxxxxxx xxxxxxxx 1: Not allowed 2: xxxxxxxx xxxxxxxx **dddddddd dddddddd** 3: Not allowed
10	Word (32 bits)	Address modulo 4 : data alignment 0: **dddddddd dddddddd dddddddd dddddddd** 1: Not allowed 2: Not allowed 3: Not allowed
11	Reserved	N/A

Starting in Run-Test/Idle, SAB data is accessed in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.
6. In Shift-DR: Scan in the direction bit (1=read, 0=write), 2-bit access size, and the 36-bit address of the data to access.
7. Go to Update-DR and re-enter Select-DR Scan.
8. In Shift-DR: For a read operation, scan out the contents of the addressed area. For a write operation, scan in the new contents of the area.
9. Return to Run-Test/Idle.

For any operation, the full 36 bits of the address must be provided. For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 39-19. MEMORY_SIZED_ACCESS Details

Instructions	Details
IR input value	**10101** (0x15)
IR output value	peb01
DR Size	39 bits
DR input value (Address phase)	aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaassr
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xxxxxxx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xxxxxxx

Table 39-19. MEMORY_SIZED_ACCESS Details (Continued)

Instructions	Details
DR output value (Address phase)	xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb
DR output value (Data read phase)	xxxxxeb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xxxxxxx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

39.5.3.4 *MEMORY_WORD_ACCESS*

This instruction allows access to the entire Service Access Bus data area. Data is accessed through the 34 MSB of the SAB address, a direction bit, and 32 bits of data. This instruction is identical to MEMORY_SIZED_ACCESS except that it always does word sized accesses. The size field is implied, and the two lowest address bits are removed and not scanned in.

Note: This instruction was previously known as MEMORY_ACCESS, and is provided for backwards compatibility.

The data register is alternately interpreted by the SAB as an address register and a data register. The SAB starts in address mode after the MEMORY_WORD_ACCESS instruction is selected, and toggles between address and data mode each time a data scan completes with the busy bit cleared.

Starting in Run-Test/Idle, SAB data is accessed in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.
6. In Shift-DR: Scan in the direction bit (1=read, 0=write) and the 34-bit address of the data to access.
7. Go to Update-DR and re-enter Select-DR Scan.
8. In Shift-DR: For a read operation, scan out the contents of the addressed area. For a write operation, scan in the new contents of the area.
9. Return to Run-Test/Idle.

For any operation, the full 34 bits of the address must be provided. For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 39-20. MEMORY_WORD_ACCESS Details

Instructions	Details
IR input value	**10001** (0x11)
IR output value	peb01
DR Size	35 bits
DR input value (Address phase)	aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aar
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xxx
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xxx

Table 39-20. MEMORY_WORD_ACCESS Details (Continued)

Instructions	Details
DR output value (Address phase)	xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xeb
DR output value (Data read phase)	xeb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xxx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

39.5.3.5 MEMORY_BLOCK_ACCESS

This instruction allows access to the entire SAB data area. Up to 32 bits of data is accessed at a time, while the address is sequentially incremented from the previously used address.

In this mode, the SAB address, size, and access direction is not provided with each access. Instead, the previous address is auto-incremented depending on the specified size and the previous operation repeated. The address must be set up in advance with MEMORY_SIZE_ACCESS or MEMORY_WORD_ACCESS. It is allowed, but not required, to shift data after shifting the address.

This instruction is primarily intended to speed up large quantities of sequential word accesses. It is possible to use it also for byte and halfword accesses, but the overhead in this is case much larger as 32 bits must still be shifted for each access.

The following sequence should be used:

1. Use the MEMORY_SIZE_ACCESS or MEMORY_WORD_ACCESS to read or write the first location.
2. Return to Run-Test/Idle.
3. Select the IR Scan path.
4. In Capture-IR: The IR output value is latched into the shift register.
5. In Shift-IR: The instruction register is shifted by the TCK input.
6. Return to Run-Test/Idle.
7. Select the DR Scan path. The address will now have incremented by 1, 2, or 4 (corresponding to the next byte, halfword, or word location).
8. In Shift-DR: For a read operation, scan out the contents of the next addressed location. For a write operation, scan in the new contents of the next addressed location.
9. Go to Update-DR.
10. If the block access is not complete, return to Select-DR Scan and repeat the access.
11. If the block access is complete, return to Run-Test/Idle.

For write operations, 32 data bits must be provided, or the result will be undefined. For read operations, shifting may be terminated once the required number of bits have been acquired.

Table 39-21. MEMORY_BLOCK_ACCESS Details

Instructions	Details
IR input value	**10010** (0x12)
IR output value	peb01
DR Size	34 bits
DR input value (Data read phase)	**xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx** xx

Table 39-21. MEMORY_BLOCK_ACCESS Details (Continued)

Instructions	Details
DR input value (Data write phase)	**dddddddd dddddddd dddddddd dddddddd** xx
DR output value (Data read phase)	eb **dddddddd dddddddd dddddddd dddddddd**
DR output value (Data write phase)	xx **xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxeb**

The overhead using block word access is 4 cycles per 32 bits of data, resulting in an 88% transfer efficiency, or 2.1 MBytes per second with a 20 MHz TCK frequency.

39.5.3.6 CANCEL_ACCESS

If a very slow memory location is accessed during a SAB memory access, it could take a very long time until the busy bit is cleared, and the SAB becomes ready for the next operation. The CANCEL_ACCESS instruction provides a possibility to abort an ongoing transfer and report a timeout to the JTAG master.

When the CANCEL_ACCESS instruction is selected, the current access will be terminated as soon as possible. There are no guarantees about how long this will take, as the hardware may not always be able to cancel the access immediately. The SAB is ready to respond to a new command when the busy bit clears.

Starting in Run-Test/Idle, CANCEL_ACCESS is accessed in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.

Table 39-22. CANCEL_ACCESS Details

Instructions	Details
IR input value	**10011** (0x13)
IR output value	peb01
DR Size	1
DR input value	x
DR output value	0

39.5.3.7 SYNC

This instruction allows external debuggers and testers to measure the ratio between the external JTAG clock and the internal system clock. The SYNC data register is a 16-bit counter that counts down to zero using the internal system clock. The busy bit stays high until the counter reaches zero.

Starting in Run-Test/Idle, SYNC instruction is used in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.

6. Scan in an 16-bit counter value.
7. Go to Update-DR and re-enter Select-DR Scan.
8. In Shift-DR: Scan out the busy bit, and until the busy bit clears goto 7.
9. Calculate an approximation to the internal clock speed using the elapsed time and the counter value.
10. Return to Run-Test/Idle.

The full 16-bit counter value must be provided when starting the synch operation, or the result will be undefined. When reading status, shifting may be terminated once the required number of bits have been acquired.

Table 39-23. SYNC_ACCESS Details

Instructions	Details
IR input value	**10111** (0x17)
IR output value	peb01
DR Size	16 bits
DR input value	dddddddd dddddddd
DR output value	xxxxxxxx xxxxxxeb

39.5.3.8 AVR_RESET

This instruction allows a debugger or tester to directly control separate reset domains inside the chip. The shift register contains one bit for each controllable reset domain. Setting a bit to one resets that domain and holds it in reset. Setting a bit to zero releases the reset for that domain.

The AVR_RESET instruction can be used in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.
6. In Shift-DR: Scan in the value corresponding to the reset domains the JTAG master wants to reset into the data register.
7. Return to Run-Test/Idle.
8. Stay in run test idle for at least 10 TCK clock cycles to let the reset propagate to the system.

See the device specific documentation for the number of reset domains, and what these domains are.

For any operation, all bits must be provided or the result will be undefined.

Table 39-24. AVR_RESET Details

Instructions	Details
IR input value	**01100** (0x0C)
IR output value	p0001

Table 39-24. AVR_RESET Details (Continued)

Instructions	Details
DR Size	Device specific.
DR input value	Device specific.
DR output value	Device specific.

39.5.3.9 CHIP_ERASE

This instruction allows a programmer to completely erase all nonvolatile memories in a chip. This will also clear any security bits that are set, so the device can be accessed normally. In devices without non-volatile memories this instruction does nothing, and appears to complete immediately.

The erasing of non-volatile memories starts as soon as the CHIP_ERASE instruction is selected. The CHIP_ERASE instruction selects a 1 bit bypass data register.

A chip erase operation should be performed as:

1. Reset the system and stop the CPU from executing.
2. Select the IR Scan path.
3. In Capture-IR: The IR output value is latched into the shift register.
4. In Shift-IR: The instruction register is shifted by the TCK input.
5. Check the busy bit that was scanned out during Shift-IR. If the busy bit was set goto 2.
6. Return to Run-Test/Idle.

Table 39-25. CHIP_ERASE Details

Instructions	Details
IR input value	**01111** (0x0F)
IR output value	p0b01 Where b is the *busy* bit.
DR Size	1 bit
DR input value	x
DR output value	0

39.5.3.10 HALT

This instruction allows a programmer to easily stop the CPU to ensure that it does not execute invalid code during programming.

This instruction selects a 1-bit halt register. Setting this bit to one resets the device and halts the CPU. Setting this bit to zero resets the device and releases the CPU to run normally. The value shifted out from the data register is one if the CPU is halted.

The HALT instruction can be used in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.

6. In Shift-DR: Scan in the value 1 to halt the CPU, 0 to start CPU execution.
7. Return to Run-Test/Idle.

Table 39-26. HALT Details

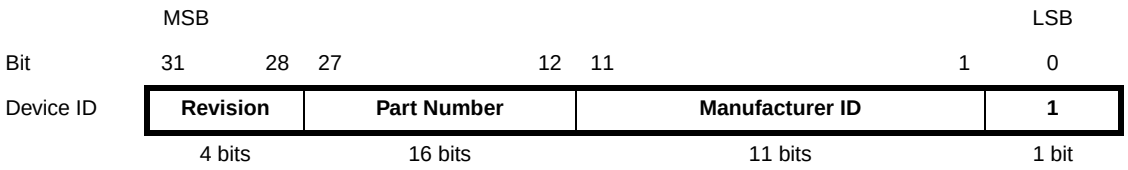
Instructions	Details
IR input value	**11100** (0x1C)
IR output value	p0001
DR Size	1 bit
DR input value	d
DR output value	d

39.5.4 JTAG Data Registers

The following device specific registers can be selected as JTAG scan chain depending on the instruction loaded in the JTAG Instruction Register. Additional registers exist, but are implicitly described in the functional description of the relevant instructions.

39.5.4.1 Device Identification Register

The Device Identification Register contains a unique identifier for each product. The register is selected by the IDCODE instruction, which is the default instruction after a JTAG reset.



Revision This is a 4 bit number identifying the revision of the component. Rev A = 0x0, B = 0x1, etc.

Part Number The part number is a 16 bit code identifying the component.

Manufacturer ID The Manufacturer ID is a 11 bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is 0x01F.

Device specific ID codes

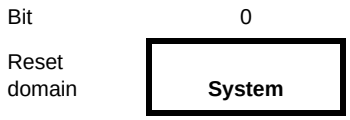
The different device configurations have different JTAG ID codes, as shown in Table 39-27. Note that if the flash controller is statically reset, the ID code will be undefined.

Table 39-27. Device and JTAG ID

Device Name	JTAG ID Code (R is the revision number)
AT32UC3C0512C	0xr200003F
AT32UC3C0256C	0xr200103F
AT32UC3C0128C	0xr200203F
AT32UC3C064C	0xr200303F
AT32UC3C1512C	0xr200403F
AT32UC3C1256C	0xr200503F
AT32UC3C1128C	0xr200603F
AT32UC3C164C	0xr200703F
AT32UC3C2512C	0xr200803F
AT32UC3C2256C	0xr200903F
AT32UC3C2128C	0xr200A03F
AT32UC3C264C	0xr200B03F

39.5.4.2 *Reset Register*

The reset register is selected by the AVR_RESET instruction and contains one bit for each reset domain in the device. Setting each bit to one will keep that domain reset until the bit is cleared.



System	Resets the whole chip, except the JTAG itself.

39.5.4.3 *Boundary--scan Chain*

The boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as driving and observing the logic levels between the digital I/O pins and the internal logic. Typically, output value, output enable, and input data are all available in the boundary-scan chain.

The boundary-scan chain is described in the BSDL (Boundary Scan Description Language) file available at the Atmel web site.

39.6 aWire Debug Interface (AW)

Rev.: 2.3.0.1

39.6.1 Features

- **Single pin debug system.**
- **Half Duplex asynchronous communication (UART compatible).**
- **Full duplex mode for direct UART connection.**
- **Compatible with JTAG functionality, except boundary scan.**
- **Failsafe packet-oriented protocol.**
- **Read and write on-chip memory and program on-chip flash and fuses through SAB interface.**
- **On-Chip Debug access through SAB interface.**
- **Asynchronous receiver or transmitter when the aWire system is not used for debugging.**

39.6.2 Overview

The aWire Debug Interface (AW) offers a single pin debug solution that is fully compatible with the functionality offered by the JTAG interface, except boundary scan. This functionality includes memory access, programming capabilities, and On-Chip Debug access.

Figure 39-8 on page 1232 shows how the AW is connected in a 32-bit AVR device. The RESET_N pin is used both as reset and debug pin. A special sequence on RESET_N is needed to block the normal reset functionality and enable the AW.

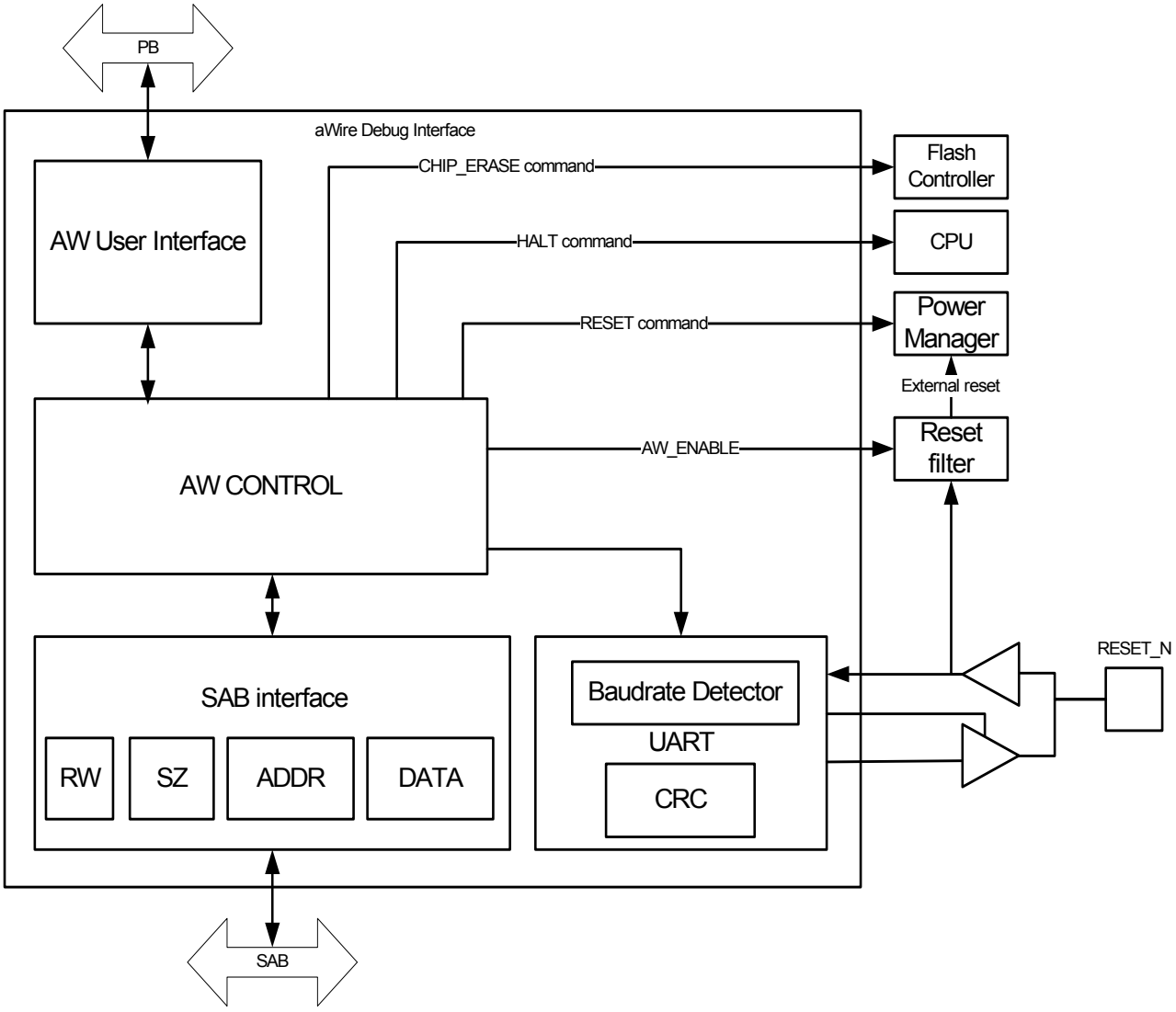
The Service Access Bus (SAB) interface contains address and data registers for the Service Access Bus, which gives access to On-Chip Debug, programming, and other functions in the device. The SAB offers several modes of access to the address and data registers, as discussed in Section 39.6.6.8.

Section 39.6.7 lists the supported aWire commands and responses, with references to the description in this document.

If the AW is not used for debugging, the aWire UART can be used by the user to send or receive data with one stop bit, eight data bits, no parity bits, and one stop bit. This can be controlled through the aWire user interface.

39.6.3 Block Diagram

Figure 39-8. aWire Debug Interface Block Diagram



39.6.4 I/O Lines Description

Table 39-28. I/O Lines Description

Name	Description	Type
DATA	aWire data multiplexed with the RESET_N pin.	Input/Output
DATAOUT	aWire data output in 2-pin mode.	Output

39.6.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

39.6.5.1 I/O Lines

The pin used by AW is multiplexed with the RESET_N pin. The reset functionality is the default function of this pin. To enable the aWire functionality on the RESET_N pin the user must enable the AW either by sending the enable sequence over the RESET_N pin from an external aWire master or by enabling the aWire user interface.

In 2-pin mode data is received on the RESET_N line, but transmitted on the DATAOUT line. After sending the 2_PIN_MODE command the DATAOUT line is automatically enabled. All other peripheral functions on this pin is disabled.

39.6.5.2 Power Management

When debugging through AW the system clocks are automatically turned on to allow debugging in sleep modes.

39.6.5.3 Clocks

The aWire UART uses the internal 120 MHz RC oscillator (RC120M) as clock source for its operation. When enabling the AW the RC120M is automatically started.

39.6.6 Functional Description

39.6.6.1 aWire Communication Protocol

The AW is accessed through the RESET_N pin shown in Table 39-28 on page 1232. The AW communicates through a UART operating at variable baud rate (depending on a sync pattern) with one start bit, 8 data bits (LSB first), one stop bit, and no parity bits. The aWire protocol is based upon command packets from an externalmaster and response packets from the slave (AW). The master always initiates communication and decides the baud rate.

The packet contains a sync byte (0x55), a command/response byte, two length bytes (optional), a number of data bytes as defined in the length field (optional), and two CRC bytes. If the command/response has the most significant bit set, the command/response also carries the optional length and data fields. The CRC field is not checked if the CRC value transmitted is 0x0000.

Table 39-29. aWire Packet Format

Field	Number of bytes	Description	Comment	Optional
SYNC	1	Sync pattern (0x55).	Used by the receiver to set the baud rate clock.	No
COMMAND/ RESPONSE	1	Command from the master or response from the slave.	When the most significant bit is set the command/response has a length field. A response has the next most significant bit set. A command does not have this bit set.	No
LENGTH	2	The number of bytes in the DATA field.		Yes
DATA	LENGTH	Data according to command/ response.		Yes
CRC	2	CRC calculated with the FCS16 polynomial.	CRC value of 0x0000 makes the aWire disregard the CRC if the master does not support it.	No

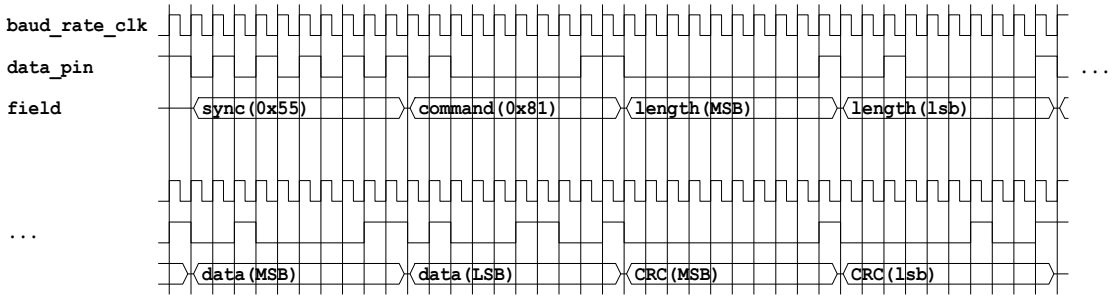
CRC calculation

The CRC is calculated from the command/response, length, and data fields. The polynomial used is the FCS16 (or CRC-16-CCIT) in reverse mode (0x8408) and the starting value is 0x0000.

Example command

Below is an example command from the master with additional data.

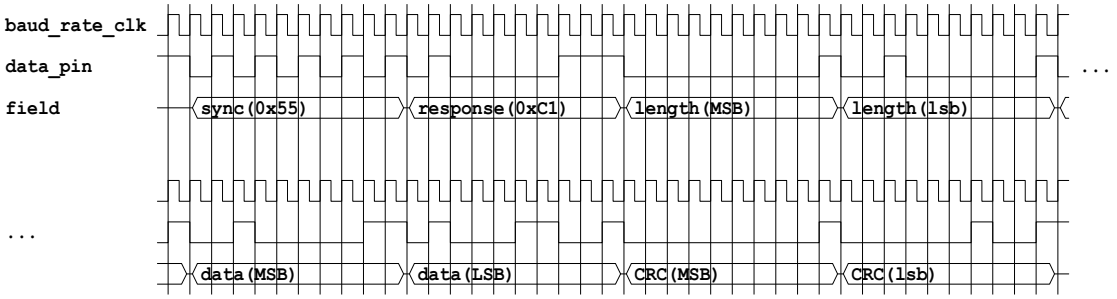
Figure 39-9. Example Command



Example response

Below is an example response from the slave with additional data.

Figure 39-10. Example Response



Avoiding drive contention when changing direction

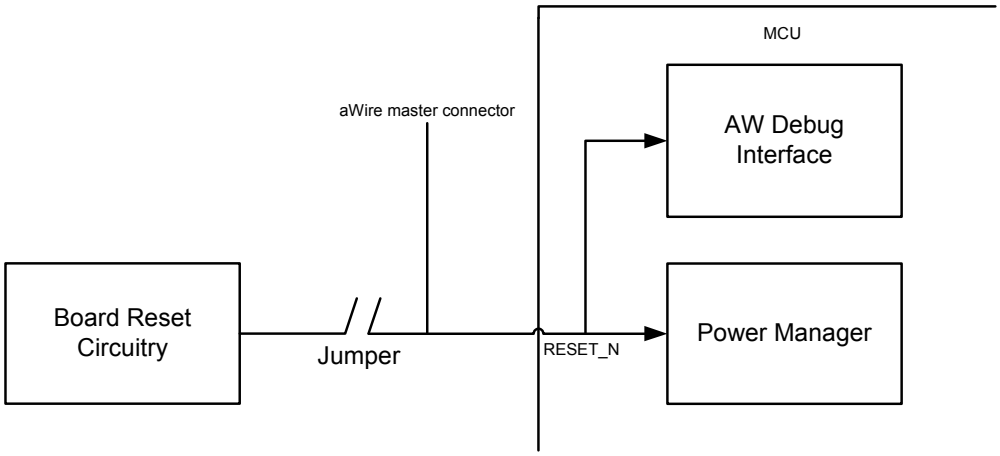
The aWire debug protocol uses one dataline in both directions. To avoid both the master and the slave to drive this line when changing direction the AW has a built in guard time before it starts to drive the line. At reset this guard time is set to maximum (128 bit cycles), but can be lowered by the master upon command.

The AW will release the line immediately after the stop character has been transmitted.

39.6.6.2 During the direction change there can be a period when the line is not driven. The internal pullup of the RESET_N pin keeps the signal stable when neither master or slave is actively driving the line. *The RESET_N pin*

Normal reset functionality on the RESET_N pin is disabled when using aWire. However, the user can reset the system through the RESET aWire command. During aWire operation the RESET_N pin should not be connected to an external reset circuitry, but disconnected via a switch or a jumper to avoid drive contention and speed problems.

Figure 39-11. Reset Circuitry and aWire.



39.6.6.3 Initializing the AW

To enable AW, the user has to send a 0x55 pattern with a baudrate of 1 kHz on the RESET_N pin. The AW is enabled after transmitting this pattern and the user can start transmitting commands. This pattern is not the sync pattern for the first command.

After enabling the awire debug interface the halt bit is set automatically to prevent the system from running code after the interface is enabled. To make the CPU run again set halt to zero using the HALT command.

39.6.6.4 Disabling the AW

To disable AW, the user can keep the RESET_N pin low for 100 ms. This will disable the AW, return RESET_N to its normal function, and reset the device.

An aWire master can also disable aWire by sending the DISABLE command. After acking the command the AW will be disabled and RESET_N returns to its normal function.

39.6.6.5 Resetting the AW

The aWire master can reset the AW slave by pulling the RESET_N pin low for 20 ms. This is equivalent to disabling and then enabling AW.

39.6.6.6 2-pin Mode

To avoid using special hardware when using a normal UART device as aWire master, the aWire slave has a 2-pin mode where one pin is used as input and on pin is used as output. To enable this mode the 2_PIN_MODE command must be sent. After sending the command, all responses will be sent on the DATAOUT pin instead of the RESET_N pin. Commands are still received on the RESET_N pin.

39.6.6.7 Baud Rate Clock

The communication speed is set by the master in the sync field of the command. The AW will use this to resynchronize its baud rate clock and reply on this frequency. The minimum frequency of the communication is 1 kHz. The maximum frequency depends on the internal clock source for the AW (RC120M). The baud rate clock is generated by AW with the following formula:

$$f_{aw} = \frac{TUNE \times f_{br}}{8}$$

Where f_{br} is the baud rate frequency and f_{aw} is the frequency of the internal RC120M. TUNE is the value returned by the BAUD_RATE response.

To find the max frequency the user can issue the TUNE command to the AW to make it return the TUNE value. This value can be used to compute the f_{aw}. The maximum operational frequency (f_{brmax}) is then:

$$f_{brmax} = \frac{f_{aw}}{4}$$

39.6.6.8 Service Access Bus

The AVR32 architecture offers a common interface for access to On-Chip Debug, programming, and test functions. These are mapped on a common bus called the Service Access Bus (SAB), which is linked to the aWire through a bus master module, which also handles synchronization between the aWire and SAB clocks.

For more information about the SAB and a list of SAB slaves see the Service Access Bus chapter.

SAB Clock

When accessing the SAB through the aWire there are no limitations on baud rate frequency compared to chip frequency, although there must be an active system clock in order for the SAB accesses to complete. If the system clock (CLK_SYS) is switched off in sleep mode, activity on the aWire pin will restart the CLK_SYS automatically, without waking the device from sleep. aWire masters may optimize the transfer rate by adjusting the baud rate frequency in relation to the CLK_SYS. This ratio can be measured with the MEMORY_SPEED_REQUEST command.

When issuing the MEMORY_SPEED_REQUEST command a counter value CV is returned. CV can be used to calculate the SAB speed (f_{sab}) using this formula:

$$f_{sab} = \frac{3f_{aw}}{CV - 3}$$

SAB Address Mode

The Service Access Bus uses 36 address bits to address memory or registers in any of the slaves on the bus. The bus supports sized accesses of bytes (8 bits), halfwords (16 bits), or words (32 bits). All accesses must be aligned to the size of the access, i.e. halfword accesses must have the lowest address bit cleared, and word accesses must have the two lowest address bits cleared.

Two instructions exist to access the SAB: MEMORY_WRITE and MEMORY_READ. These two instructions write and read words, halfwords, and bytes from the SAB.

Busy Reporting

If the aWire master, during a MEMORY_WRITE or a MEMORY_READ command, transmit another byte when the aWire is still busy sending the previous byte to the SAB, the AW will respond with a MEMORY_READ_WRITE_STATUS error. See chapter Section 39.6.8.5 for more details.

The aWire master should adjust its baudrate or delay between bytes when doing SAB accesses to ensure that the SAB is not overwhelmed with data.

Error Reporting

If a write is performed on a non-existing memory location the SAB interface will respond with an error. If this happens, all further writes in this command will not be performed and the error and number of bytes written is reported in the MEMORY_READWRITE_STATUS message from the AW after the write.

If a read is performed on a non-existing memory location, the SAB interface will respond with an error. If this happens, the data bytes read after this event are not valid. The AW will include three extra bytes at the end of the transfer to indicate if the transfer was successful, or in the case of an error, how many valid bytes were received.

39.6.6.9 CRC Errors/NACK Response

The AW will calculate a CRC value when receiving the command, length, and data fields of the command packets. If this value differs from the value from the CRC field of the packet, the AW will reply with a NACK response. Otherwise the command is carried out normally.

An unknown command will be replied with a NACK response.

In worst case a transmission error can happen in the length or command field of the packet. This can lead to the aWire slave trying to receive a command with or without length (opposite of what the master intended) or receive an incorrect number of bytes. The aWire slave will then either wait for more data when the master has finished or already have transmitted the NACK response in congestion with the master. The master can implement a timeout on every command and reset the slave if no response is returned after the timeout period has ended.

39.6.7 aWire Command Summary

The implemented aWire commands are shown in the table below. The responses from the AW are listed in Section 39.6.8.

Table 39-30. aWire Command Summary

COMMAND	Instruction	Description
0x01	AYA	“Are you alive”.
0x02	JTAG_ID	Asks AW to return the JTAG IDCODE.
0x03	STATUS_REQUEST	Request a status message from the AW.
0x04	TUNE	Tell the AW to report the current baud rate.
0x05	MEMORY_SPEED_REQUEST	Reports the speed difference between the aWire control and the SAB clock domains.
0x06	CHIP_ERASE	Erases the flash and all volatile memories.

Table 39-30. aWire Command Summary

COMMAND	Instruction	Description
0x07	DISABLE	Disables the AW.
0x08	2_PIN_MODE	Enables the DATAOUT pin and puts the aWire in 2-pin mode, where all responses are sent on the DATAOUT pin.
0x80	MEMORY_WRITE	Writes words, halfwords, or bytes to the SAB.
0x81	MEMORY_READ	Reads words, halfwords, or bytes from the SAB.
0x82	HALT	Issues a halt command to the device.
0x83	RESET	Issues a reset to the Reset Controller.
0x84	SET_GUARD_TIME	Sets the guard time for the AW.

All aWire commands are described below, with a summary in table form.

Table 39-31. Command/Response Description Notation

Command/Response	Description
Command/Response value	Shows the command/response value to put into the command/response field of the packet.
Additional data	Shows the format of the optional data field if applicable.
Possible responses	Shows the possible responses for this command.

39.6.7.1 AYA

This command asks the AW: “Are you alive”, where the AW should respond with an acknowledge.

Table 39-32. AYA Details

Command	Details
Command value	0x01
Additional data	N/A
Possible responses	0x40: ACK (Section 39.6.8.1) 0x41: NACK (Section 39.6.8.2)

39.6.7.2 JTAG_ID

This command instructs the AW to output the JTAG idcode in the following response.

Table 39-33. JTAG_ID Details

Command	Details
Command value	0x02
Additional data	N/A
Possible responses	0xC0: IDCODE (Section 39.6.8.3) 0x41: NACK (Section 39.6.8.2)

39.6.7.3 STATUS_REQUEST

Asks the AW for a status message.

Table 39-34. STATUS_REQUEST Details

Command	Details
Command value	0x03
Additional data	N/A
Possible responses	0xC4: STATUS_INFO (Section 39.6.8.7) 0x41: NACK (Section 39.6.8.2)

39.6.7.4 TUNE

Asks the AW for the current baud rate counter value.

Table 39-35. TUNE Details

Command	Details
Command value	0x04
Additional data	N/A
Possible responses	0xC3: BAUD_RATE (Section 39.6.8.6) 0x41: NACK (Section 39.6.8.2)

39.6.7.5 MEMORY_SPEED_REQUEST

Asks the AW for the relative speed between the aWire clock (RC120M) and the SAB interface.

Table 39-36. MEMORY_SPEED_REQUEST Details

Command	Details
Command value	0x05
Additional data	N/A
Possible responses	0xC5: MEMORY_SPEED (Section 39.6.8.8) 0x41: NACK (Section 39.6.8.2)

39.6.7.6 CHIP_ERASE

This instruction allows a programmer to completely erase all nonvolatile memories in the chip. This will also clear any security bits that are set, so the device can be accessed normally. The command is acked immediately, but the status of the command can be monitored by checking the Chip Erase ongoing bit in the status bytes received after the STATUS_REQUEST command.

Table 39-37. CHIP_ERASE Details

Command	Details
Command value	0x06
Additional data	N/A
Possible responses	0x40: ACK (Section 39.6.8.1) 0x41: NACK (Section 39.6.8.2)

39.6.7.7 DISABLE

Disables the AW. The AW will respond with an ACK response and then disable itself.

Table 39-38. DISABLE Details

Command	Details
Command value	0x07
Additional data	N/A
Possible responses	0x40: ACK (Section 39.6.8.1) 0x41: NACK (Section 39.6.8.2)

39.6.7.8 2_PIN_MODE

Enables the DATAOUT pin as an output pin. All responses sent from the aWire slave will be sent on this pin, instead of the RESET_N pin, starting with the ACK for the 2_PIN_MODE command.

Table 39-39. DISABLE Details

Command	Details
Command value	0x07
Additional data	N/A
Possible responses	0x40: ACK (Section 39.6.8.1) 0x41: NACK (Section 39.6.8.2)

39.6.7.9 MEMORY_WRITE

This command enables programming of memory/writing to registers on the SAB. The MEMORY_WRITE command allows words, halfwords, and bytes to be programmed to a continuous sequence of addresses in one operation. Before transferring the data, the user must supply:

1. The number of data **bytes** to write + 5 (size and starting address) in the length field.
2. The size of the transfer: words, halfwords, or bytes.
3. The starting address of the transfer.

The 4 MSB of the 36 bit SAB address are submitted together with the size field (2 bits). Then follows the 4 remaining address bytes and finally the data bytes. The size of the transfer is specified using the values from the following table:

Table 39-40. Size Field Decoding

Size field	Description
00	Byte transfer
01	Halfword transfer
10	Word transfer
11	Reserved

Below is an example write command:

1. 0x55 (sync)
2. 0x80 (command)
3. 0x00 (length MSB)

4. 0x09 (length LSB)
5. 0x25 (size and address MSB, the two MSB of this byte are unused and set to zero)
6. 0x00
7. 0x00
8. 0x00
9. 0x04 (address LSB)
10. 0xCA
11. 0xFE
12. 0xBA
13. 0xBE
14. 0xXX (CRC MSB)
15. 0xXX (CRC LSB)

The length field is set to 0x0009 because there are 9 bytes of additional data: 5 address and size bytes and 4 bytes of data. The address and size field indicates that words should be written to address 0x500000004. The data written to 0x500000004 is 0xCAFEBABE.

Table 39-41. MEMORY_WRITE Details

Command	Details
Command value	0x80
Additional data	Size, Address and Data
Possible responses	0xC2: MEMORY_READWRITE_STATUS (Section 39.6.8.5) 0x41: NACK (Section 39.6.8.2)

39.6.7.10 MEMORY_READ

This command enables reading of memory/registers on the Service Access Bus (SAB). The MEMORY_READ command allows words, halfwords, and bytes to be read from a continuous sequence of addresses in one operation. The user must supply:

1. The size of the data field: 7 (size and starting address + read length indicator) in the length field.
2. The size of the transfer: Words, halfwords, or bytes.
3. The starting address of the transfer.
4. The number of **bytes** to read (max 65532).

The 4 MSB of the 36 bit SAB address are submitted together with the size field (2 bits). The 4 remaining address bytes are submitted before the number of bytes to read. The size of the transfer is specified using the values from the following table:

Table 39-42. Size Field Decoding

Size field	Description
00	Byte transfer
01	Halfword transfer
10	Word transfer
11	Reserved

Below is an example read command:

1. 0x55 (sync)
2. 0x81 (command)
3. 0x00 (length MSB)
4. 0x07 (length LSB)
5. 0x25 (size and address MSB, the two MSB of this byte are unused and set to zero)
6. 0x00
7. 0x00
8. 0x00
9. 0x04 (address LSB)
10. 0x00
11. 0x04
12. 0xXX (CRC MSB)
13. 0xXX (CRC LSB)

The length field is set to 0x0007 because there are 7 bytes of additional data: 5 bytes of address and size and 2 bytes with the number of bytes to read. The address and size field indicates one word (four bytes) should be read from address 0x500000004.

Table 39-43. MEMORY_READ Details

Command	Details
Command value	0x81
Additional data	Size, Address and Length
Possible responses	0xC1: MEMDATA (Section 39.6.8.4) 0xC2: MEMORY_READWRITE_STATUS (Section 39.6.8.5) 0x41: NACK (Section 39.6.8.2)

39.6.7.11 HALT

This command tells the CPU to halt code execution for safe programming. If the CPU is not halted during programming it can start executing partially loaded programs. To halt the processor, the aWire master should send 0x01 in the data field of the command. After programming the halting can be released by sending 0x00 in the data field of the command.

Table 39-44. HALT Details

Command	Details
Command value	0x82
Additional data	0x01 to halt the CPU 0x00 to release the halt and reset the device.
Possible responses	0x40: ACK (Section 39.6.8.1) 0x41: NACK (Section 39.6.8.2)

39.6.7.12 RESET

This command resets different domains in the part. The aWire master sends a byte with the reset value. Each bit in the reset value byte corresponds to a reset domain in the chip. If a bit is set the reset is activated and if a bit is not set the reset is released. The number of reset domains

and their destinations are identical to the resets described in the JTAG data registers chapter under reset register.

Table 39-45. RESET Details

Command	Details
Command value	0x83
Additional data	Reset value for each reset domain. The number of reset domains is part specific.
Possible responses	0x40: ACK (Section 39.6.8.1) 0x41: NACK (Section 39.6.8.2)

39.6.7.13 SET_GUARD_TIME

Sets the guard time value in the AW, i.e. how long the AW will wait before starting its transfer after the master has finished.

The guard time can be either 0x00 (128 bit lengths), 0x01 (16 bit lengths), 0x2 (4 bit lengths) or 0x3 (1 bit length).

Table 39-46. SET_GUARD_TIME Details

Command	Details
Command value	0x84
Additional data	Guard time
Possible responses	0x40: ACK (Section 39.6.8.1) 0x41: NACK (Section 39.6.8.2)

39.6.8 aWire Response Summary

The implemented aWire responses are shown in the table below.

Table 39-47. aWire Response Summary

RESPONSE	Instruction	Description
0x40	ACK	Acknowledge.
0x41	NACK	Not acknowledge. Sent after CRC errors and after unknown commands.
0xC0	IDCODE	The JTAG idcode.
0xC1	MEMDATA	Values read from memory.
0xC2	MEMORY_READWRITE_STATUS	Status after a MEMORY_WRITE or a MEMORY_READ command. OK, busy, error.
0xC3	BAUD_RATE	The current baudrate.
0xC4	STATUS_INFO	Status information.
0xC5	MEMORY_SPEED	SAB to aWire speed information.

39.6.8.1 ACK

The AW has received the command successfully and performed the operation.

Table 39-48. ACK Details

Response	Details
Response value	0x40
Additional data	N/A

39.6.8.2 NACK

The AW has received the command, but got a CRC mismatch.

Table 39-49. NACK Details

Response	Details
Response value	0x41
Additional data	N/A

39.6.8.3 IDCODE

The JTAG idcode for this device.

Table 39-50. IDCODE Details

Response	Details
Response value	0xC0
Additional data	JTAG idcode

39.6.8.4 MEMDATA

The data read from the address specified by the MEMORY_READ command. The last 3 bytes are status bytes from the read. The first status byte is the status of the command described in the table below. The last 2 bytes are the number of remaining data bytes to be sent in the data field of the packet when the error occurred. If the read was not successful all data bytes after the failure are undefined. A successful word read (4 bytes) will look like this:

1. 0x55 (sync)
2. 0xC1 (command)
3. 0x00 (length MSB)
4. 0x07 (length LSB)
5. 0xCA (Data MSB)
6. 0xFE
7. 0xBA
8. 0xBE (Data LSB)
9. 0x00 (Status byte)
10. 0x00 (Bytes remaining MSB)
11. 0x00 (Bytes remaining LSB)
12. 0xXX (CRC MSB)
13. 0xXX (CRC LSB)

The status is 0x00 and all data read are valid. An unsuccessful four byte read can look like this:

1. 0x55 (sync)
2. 0xC1 (command)
3. 0x00 (length MSB)
4. 0x07 (length LSB)
5. 0xCA (Data MSB)
6. 0xFE
7. 0xXX (An error has occurred. Data read is undefined. 5 bytes remaining of the Data field)
8. 0xXX (More undefined data)
9. 0x02 (Status byte)
10. 0x00 (Bytes remaining MSB)
11. 0x05 (Bytes remaining LSB)
12. 0xXX (CRC MSB)
13. 0xXX (CRC LSB)

The error occurred after reading 2 bytes on the SAB. The rest of the bytes read are undefined. The status byte indicates the error and the bytes remaining indicates how many bytes were remaining to be sent of the data field of the packet when the error occurred.

Table 39-51. MEMDATA Status Byte

status byte	Description
0x00	Read successful
0x01	SAB busy
0x02	Bus error (wrong address)
Other	Reserved

Table 39-52. MEMDATA Details

Response	Details
Response value	0xC1
Additional data	Data read, status byte, and byte count (2 bytes)

39.6.8.5 MEMORY_READWRITE_STATUS

After a MEMORY_WRITE command this response is sent by AW. The response can also be sent after a MEMORY_READ command if AW encountered an error when receiving the address. The response contains 3 bytes, where the first is the status of the command and the 2 next contains the byte count when the first error occurred. The first byte is encoded this way:

Table 39-53. MEMORY_READWRITE_STATUS Status Byte

status byte	Description
0x00	Write successful
0x01	SAB busy
0x02	Bus error (wrong address)
Other	Reserved

Table 39-54. MEMORY_READWRITE_STATUS Details

Response	Details
Response value	0xC2
Additional data	Status byte and byte count (2 bytes)

39.6.8.6 *BAUD_RATE*

The current baud rate in the AW. See Section 39.6.6.7 for more details.

Table 39-55. BAUD_RATE Details

Response	Details
Response value	0xC3
Additional data	Baud rate

39.6.8.7 *STATUS_INFO*

A status message from AW.

Table 39-56. STATUS_INFO Contents

Bit number	Name	Description
15-9	Reserved	
8	Protected	The protection bit in the internal flash is set. SAB access is restricted. This bit will read as one during reset.
7	SAB busy	The SAB bus is busy with a previous transfer. This could indicate that the CPU is running on a very slow clock, the CPU clock has stopped for some reason or that the part is in constant reset.
6	Chip erase ongoing	The Chip erase operation has not finished.
5	CPU halted	This bit will be set if the CPU is halted. This bit will read as zero during reset.
4-1	Reserved	
0	Reset status	This bit will be set if AW has reset the CPU using the RESET command.

Table 39-57. STATUS_INFO Details

Response	Details
Response value	0xC4
Additional data	2 status bytes

39.6.8.8 *MEMORY_SPEED*

Counts the number of RC120M clock cycles it takes to sync one message to the SAB interface and back again. The SAB clock speed (f_{sab}) can be calculated using the following formula:

$$f_{sab} = \frac{3f_{aw}}{CV - 3}$$

Table 39-58. MEMORY_SPEED Details

Response	Details
Response value	0xC5
Additional data	Clock cycle count (MS)

39.6.9 Security Restrictions

When the security fuse in the Flash is programmed, the following aWire commands are limited:

- MEMORY_WRITE
- MEMORY_READ

Unlimited access to these instructions is restored when the security fuse is erased by the CHIP_ERASE aWire command.

Note that the security bit will read as programmed and block these instructions also if the Flash Controller is statically reset.

39.7 Module Configuration

The bit mapping of the Peripheral Debug Register (PDBG) is described in the following table. Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual for details.

Table 39-59. Bit mapping of the Peripheral Debug Register (PDBG)

bit	Peripheral
0	AST
1	WDT
2	CANIF
3	QDEC0
4	QDEC1
5	ADCIFA
6	ACIFA0
7	ACIFA1
8	DACIFB0
9	DACIFB1
10	PEVC
11-31	Reserved

40. Electrical Characteristics

40.1 Absolute Maximum Ratings*

Operating temperature	-40°C to +85°C
Storage temperature	-60°C to +150°C
Voltage on any pin except DM/DP/VBUS with respect to ground	-0.3V to V_{VDD}[1]+0.3V
Voltage on DM/DP with respect to ground	-0.3V to +3.6V
Voltage on VBUS with respect to ground	-0.3V to +5.5V
Maximum operating voltage (VDDIN_5)	5.5V
Maximum operating voltage (VDDIO1, VDDIO2, VDDIO3, VDDANA)	5.5V
Maximum operating voltage (VDDIN_33)	3.6V
Total DC output current on all I/O pins- VDDIO1	120 mA
Total DC output current on all I/O pins- VDDIO2	120 mA
Total DC output current on all I/O pins- VDDIO3	120 mA
Total DC output current on all I/O pins- VDDANA	120 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Notes: 1. V_{VDD} corresponds to either V_{VDDIO1}, V_{VDDIO2}, V_{VDDIO3}, or V_{VDDANA}, depending on the supply for the pin. Refer to Section 3-1 on page 11 for details.

40.2 Supply Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified and are valid for a junction temperature up to T_J = 100°C. Please refer to Section 6. ”Supply and Startup Considerations” on page 46.

Table 40-1. Supply Characteristics

Symbol	Parameter	Condition	Voltage		
			Min	Max	Unit
V_{VDDIN_5}	DC supply internal regulators	3V range	3.0	3.6	V
		5V range	4.5	5.5	
V_{VDDIN_33}	DC supply USB I/O	only in 3V range	3.0	3.6	V
V_{VDDANA}	DC supply peripheral I/O and analog part	3V range	3.0	3.6	V
		5V range	4.5	5.5	
V_{VDDIO1} V_{VDDIO2} V_{VDDIO2}	DC supply peripheral I/O	3V range	3.0	3.6	V
		5V range	4.5	5.5	

Table 40-2. Supply Rise Rates and Order

Symbol	Parameter	Rise Rate		
		Min	Max	Comment
V_{VDDIN_5}	DC supply internal 3.3V regulator	0.01 V/ms	1.25 V/us	
V_{VDDIN_33}	DC supply internal 1.8V regulator	0.01 V/ms	1.25 V/us	
V_{VDDIO1} V_{VDDIO2} V_{VDDIO3}	DC supply peripheral I/O	0.01 V/ms	1.25 V/us	Rise after or at the same time as VDDIN_5, VDDIN_33
V_{VDDANA}	DC supply peripheral I/O and analog part	0.01 V/ms	1.25 V/us	Rise after or at the same time as VDDIN_5, VDDIN_33

40.3 Maximum Clock Frequencies

These parameters are given in the following conditions:

- $V_{VDDCORE}$ > 1.85V
- Temperature = -40°C to 85°C

Table 40-3. Clock Frequencies

Symbol	Parameter	Conditions	Min	Max	Units
f_{CPU}	CPU clock frequency			66	MHz
f_{PBA}	PBA clock frequency			66	MHz
f_{PBB}	PBB clock frequency			66	MHz
f_{PBC}	PBC clock frequency			66	MHz
f_{GCLK0}	GCLK0 clock frequency	Generic clock for USBC		50(1)	MHz
f_{GCLK1}	GCLK1 clock frequency	Generic clock for CANIF		66(1)	MHz
f_{GCLK2}	GCLK2 clock frequency	Generic clock for AST		80(1)	MHz
f_{GCLK4}	GCLK4 clock frequency	Generic clock for PWM		133(1)	MHz
f_{GCLK11}	GCLK11 clock frequency	Generic clock for IISC		50(1)	MHz

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

40.4 Power Consumption

The values in Table 40-4 are measured values of power consumption under the following conditions, except where noted:

- Operating conditions core supply (Figure 40-1)
  - V_{VDDIN_5} = V_{VDDIN_33} = 3.3V
  - $V_{VDDCORE}$ = 1.85V, supplied by the internal regulator
  - V_{VDDIO1} = V_{VDDIO2} = V_{VDDIO3} = 3.3V
  - V_{VDDANA} = 3.3V

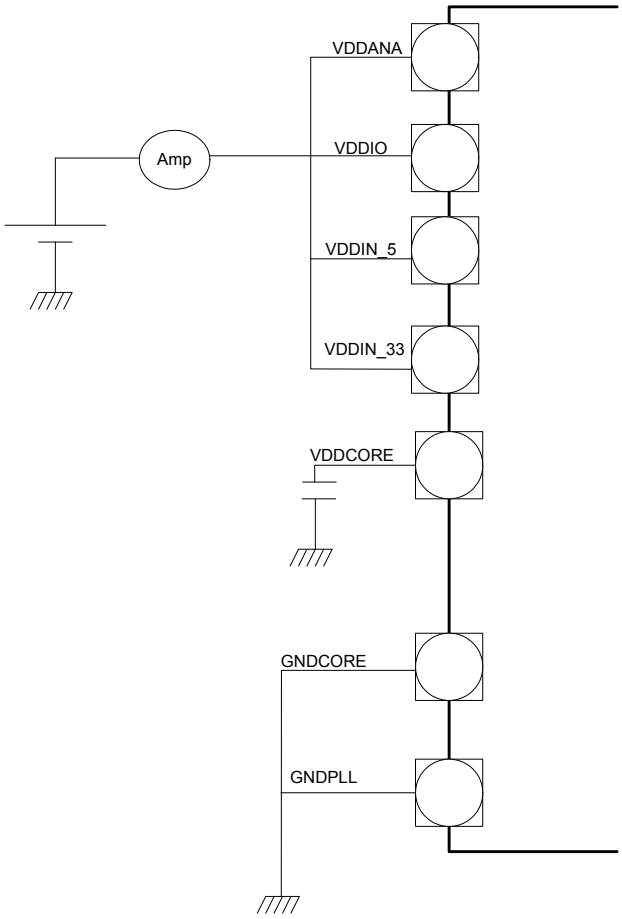
  - Internal 3.3V regulator is off
- T_A = 25°C
- I/Os are configured as inputs, with internal pull-up enabled.
- Oscillators
  - OSC0/1 (crystal oscillator) stopped
  - OSC32K (32KHz crystal oscillator) stopped
  - PLL0 running
  - PLL1 stopped
- Clocks
  - External clock on XIN0 as main clock source **(**10MHz**)**
  - CPU, HSB, and PBB clocks undivided
  - PBA, PBC clock divided by 4
  - All peripheral clocks running

Table 40-4. Power Consumption for Different Operating Modes

Mode	Conditions	Measured on	Consumption Typ	Unit
Active[1]	CPU running a recursive Fibonacci algorithm	Amp	512	µA/MHz
Idle[1]			258	
Frozen[1]			106	
Standby[1]			48	
Stop			73	µA
DeepStop			43	
Static	OSC32K and AST running		32	
	AST and OSC32K stopped		31	

Note: 1. These numbers are valid for the measured condition only and must not be extrapolated to other frequencies.

Figure 40-1. Measurement Schematic



40.4.1 Peripheral Power Consumption

The values in Table 40-5 are measured values of power consumption under the following conditions.

- Operating conditions core supply (Figure 40-1)
  - $V_{VDDIN\_5} = V_{DDIN\_33} = 3.3V$
  - $V_{VDDCORE} = 1.85V$, supplied by the internal regulator
  - $V_{VDDIO1} = V_{VDDIO2} = V_{VDDIO3} = 3.3V$
  - $V_{VDDANA} = 3.3V$
  - Internal 3.3V regulator is off.
- $T_A = 25°C$
- I/Os are configured as inputs, with internal pull-up enabled.
- Oscillators
  - OSC0/1 (crystal oscillator) stopped
  - OSC32K (32KHz crystal oscillator) stopped
  - PLL0 running

  - PLL1 stopped
- Clocks
  - External clock on XIN0 as main clock source.
  - CPU, HSB, and PB clocks undivided

Consumption active is the added current consumption when the module clock is turned on and when the module is doing a typical set of operations.

Table 40-5. Typical Current Consumption by Peripheral[2]

Peripheral	Typ Consumption Active	Unit
ACIFA[1]	3	µA/MHz
ADCIFA[1]	7	
AST	3	
CANIF	25	
DACIFB[1]	3	
EBI	23	
EIC	0.5	
FREQM	0.5	
GPIO	37	
INTC	3	
MDMA	4	
PDCA	24	
PEVC	15	
PWM	40	
QDEC	3	
SAU	3	
SDRAMC	2	
SMC	9	
SPI	5	
TC	8	
TWIM	2	
TWIS	2	
USART	10	
USBC	5	
WDT	2	

Notes: 1. Includes the current consumption on VDDANA.
2. These numbers are valid for the measured condition only and must not be extrapolated to other frequencies.

40.5 I/O Pin Characteristics

Table 40-6. Normal I/O Pin Characteristics[1]

Symbol	Parameter	Condition		Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance			5		16	kOhm
$R_{PULLDOWN}$	Pull-down resistance			2		16	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3V				0.8	V
		V_{VDD} = 4.5V				1.3	
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V		2.0			V
		V_{VDD} = 5.5V		3.0			
V_{OL}	Output low-level voltage	I_{OL} = -3.5mA, pin drive x1[2]				0.45	V
		I_{OL} = -7mA, pin drive x2[2]					
		I_{OL} = -14mA, pin drive x4[2]					
V_{OH}	Output high-level voltage	I_{OH} = 3.5mA, pin drive x1[2]		V_{VDD} - 0.8			V
		I_{OH} = 7mA, pin drive x2[2]					
		I_{OH} = 14mA, pin drive x4[2]					
f_{MAX}	Output frequency[3]	V_{VDD} = 3.0V	load = 10pF, pin drive x1[2]			35	MHz
			load = 10pF, pin drive x2[2]			55	
			load = 10pF, pin drive x4[2]			70	
			load = 30pF, pin drive x1[2]			15	
			load = 30pF, pin drive x2[2]			30	
			load = 30pF, pin drive x4[2]			45	
		V_{VDD} =4.5V	load = 10pF, pin drive x1[2]			50	
			load = 10pF, pin drive x2[2]			80	
			load = 10pF, pin drive x4[2]			95	
			load = 30pF, pin drive x1[2]			25	
			load = 30pF, pin drive x2[2]			40	
			load = 30pF, pin drive x4[2]			65	

Table 40-6. Normal I/O Pin Characteristics[1]

Symbol	Parameter	Condition		Min	Typ	Max	Units
t_{RISE}	Rise time[3]	V_{VDD} = 3.0V	load = 10pF, pin drive x1[2]			7.7	ns
			load = 10pF, pin drive x2[2]			3.4	
			load = 10pF, pin drive x4[2]			1.9	
			load = 30pF, pin drive x1[2]			16	
			load = 30pF, pin drive x2[2]			7.5	
			load = 30pF, pin drive x4[2]			3.8	
		V_{VDD} = 4.5V	load = 10pF, pin drive x1[2]			5.3	
			load = 10pF, pin drive x2[2]			2.4	
			load = 10pF, pin drive x4[2]			1.3	
			load = 30pF, pin drive x1[2]			11.1	
			load = 30pF, pin drive x2[2]			5.2	
			load = 30pF, pin drive x4[2]			2.7	
t_{FALL}	Fall time[3]	V_{VDD} = 3.0V	load = 10pF, pin drive x1[2]			7.6	ns
			load = 10pF, pin drive x2[2]			3.5	
			load = 10pF, pin drive x4[2]			1.9	
			load = 30pF, pin drive x1[2]			15.8	
			load = 30pF, pin drive x2[2]			7.3	
			load = 30pF, pin drive x4[2]			3.8	
		V_{VDD} = 4.5V	load = 10pF, pin drive x1[2]			5.2	
			load = 10pF, pin drive x2[2]			2.4	
			load = 10pF, pin drive x4[2]			1.4	
			load = 30pF, pin drive x1[2]			10.9	
			load = 30pF, pin drive x2[2]			5.1	
			load = 30pF, pin drive x4[2]			2.7	
I_{LEAK}	Input leakage current	Pull-up resistors disabled				1.0	µA
C_{IN}	Input capacitance	PA00-PA29, PB00-PB31, PC00-PC01, PC08-PC31, PD00-PD30			7.5		pF
		PC02, PC03, PC04, PC05, PC06, PC07			2		

Note:
1. V_{VDD} corresponds to either V_{VDDIO1}, V_{VDDIO2}, V_{VDDIO3}, or V_{VDDANA}, depending on the supply for the pin. Refer to Section 3-1 on page 11 for details.
2. drive x1 capability pins are: PB00, PB01, PB02, PB03, PB30, PB31, PC02, PC03, PC04, PC05, PC06, PC07 - drive x2 /x4 capability pins are: PB06, PB21, PB26, PD02, PD06, PD13 - drive x1/x2 capability pins are the remaining PA, PB, PC, PD pins. The drive strength is programmable through ODCR0, ODCR0S, ODCR0C, ODCR0T registers of GPIO.
3. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

40.6 Oscillator Characteristics

40.6.1 Oscillator (OSC0 and OSC1) Characteristics

40.6.1.1 Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN0 or XIN1.

Table 40-7. Digital Clock Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{CPXIN}	XIN clock frequency				50	MHz
t_{CPXIN}	XIN clock period		20			ns
t_{CHXIN}	XIN clock high half-priod		0.4 x t_{CPXIN}		0.6 x t_{CPXIN}	ns
t_{CLXIN}	XIN clock low half-priod		0.4 x t_{CPXIN}		0.6 x t_{CPXIN}	ns
C_{IN}	XIN input capacitance			2		pF

40.6.1.2 Crystal Oscillator Characteristics

The following table describes the characteristics for the oscillator when a crystal is connected between XIN and XOUT as shown in Figure 40-2. The user must choose a crystal oscillator where the crystal load capacitance C_L is within the range given in the table. The exact value of C_L can be found in the crystal datasheet. The capacitance of the external capacitors (C_{LEXT}) can then be computed as follows:

$$C_{LEXT} = 2(C_L - C_i) - C_{PCB}$$

where C_{PCB} is the capacitance of the PCB and C_i is the internal equivalent load capacitance.

Figure 40-2. Oscillator Connection

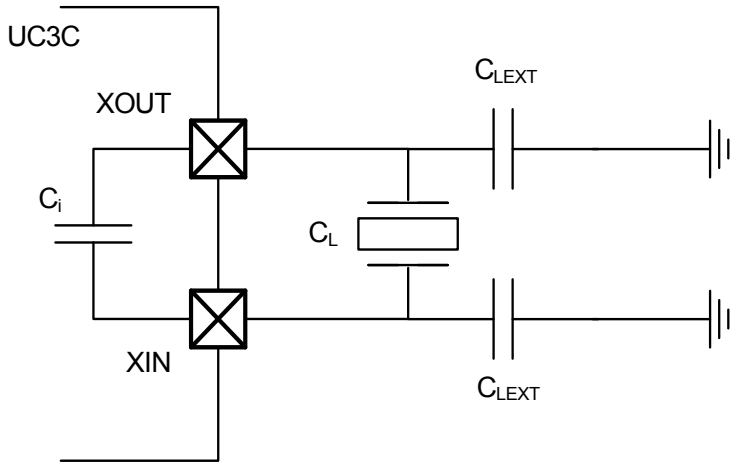


Table 40-8. Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Crystal oscillator frequency		0.4		20	MHz
C_i	Internal equivalent load capacitance			1.7		pF
$t_{STARTUP}$	Startup time	f_{OUT} = 8MHz SCIF.OSCCTRL.GAIN = 1(1)		975		us
		f_{OUT} = 16MHz SCIF.OSCCTRL.GAIN = 2(1)		1100		us

Notes: 1. Please refer to the SCIF chapter for details.

40.6.2 32KHz Crystal Oscillator (OSC32K) Characteristics

40.6.2.1 Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN32.

Table 40-9. Digital 32KHz Clock Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{CPXIN}	XIN32 clock frequency			32.768	5000	KHz
t_{CPXIN}	XIN32 clock period		200			ns
t_{CHXIN}	XIN32 clock high half-priod		0.4 x t_{CPXIN}		0.6 x t_{CPXIN}	ns
t_{CLXIN}	XIN32 clock low half-priod		0.4 x t_{CPXIN}		0.6 x t_{CPXIN}	ns
C_{IN}	XIN32 input capacitance			2		pF

40.6.2.2 Crystal Oscillator Characteristics

Figure 40-2 and the equation above also applies to the 32KHz oscillator connection. The user must choose a crystal oscillator where the crystal load capacitance C_L is within the range given in the table. The exact value of C_L can then be found in the crystal datasheet..

Table 40-10. 32 KHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Crystal oscillator frequency			32 768		Hz
$t_{STARTUP}$	Startup time	R_S = 50 kOhm, C_L = 12.5pF		2		s
C_L	Crystal load capacitance		6		15	pF
C_i	Internal equivalent load capacitance			1.4		pF

40.6.3 Phase Lock Loop (PLL0 and PLL1) Characteristics

Table 40-11. PLL Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{VCO}	Output frequency		80		240	MHz
f_{IN}	Input frequency		4		16	MHz
I_{PLL}	Current consumption	Active mode, f_{VCO} = 80MHz		250		µA
		Active mode, f_{VCO} = 240MHz		600		
$t_{STARTUP}$	Startup time, from enabling the PLL until the PLL is locked	Wide Bandwidth mode disabled		15		µs
		Wide Bandwidth mode enabled		45		

40.6.4 120MHz RC Oscillator (RC120M) Characteristics

Table 40-12. Internal 120MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency[1]		88	120	152	MHz
I_{RC120M}	Current consumption			1.85		mA
$t_{STARTUP}$	Startup time			3		µs

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

40.6.5 System RC Oscillator (RCSYS) Characteristics

Table 40-13. System RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency	Calibrated at T_A = 85°C	110	115.2	120	kHz
		T_A = 25°C	105	109	115	
		T_A = -40°C	100	104	108	

40.6.6 8MHz/1MHz RC Oscillator (RC8M) Characteristics

Table 40-14. 8MHz/1MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency	SCIF.RCCR8.FREQMODE = 0[1]	7.6	8	8.4	MHz
		SCIF.RCCR8.FREQMODE = 1[1]	0.955	1	1.045	
$t_{STARTUP}$	Startup time				20	µs

Notes: 1. Please refer to the SCIF chapter for details.

40.7 Flash Characteristics

Table 40-15 gives the device maximum operating frequency depending on the number of flash wait states. The FSW bit in the FLASHC FSR register controls the number of wait states used when accessing the flash memory.

Table 40-15. Maximum Operating Frequency

Flash Wait States	Read Mode	Maximum Operating Frequency
0	1 cycle	33 MHz
1	2 cycles	66 MHz

Table 40-16. Flash Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FPP}	Page programming time	f_{CLK_HSB} = 66 MHz		4.3		ms
t_{FPE}	Page erase time			4.3		
t_{FFP}	Fuse programming time			0.6		
t_{FEA}	Full chip erase time (EA)			4.9		
t_{FCE}	JTAG chip erase time (CHIP_ERASE)	f_{CLK_HSB} = 115 kHz		640		

Table 40-17. Flash Endurance and Data Retention

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
N_{FARRAY}	Array endurance (write/page)		100k			cycles
N_{FFUSE}	General Purpose fuses endurance (write/bit)		1k			cycles
t_{RET}	Data retention		15			years

40.8 Analog Characteristics

40.8.1 1.8V Voltage Regulator Characteristics

Table 40-18. 1.8V Voltage Regulator Electrical Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{VDDIN_5}	Input voltage range	5V range	4.5		5.5	V
		3V range	3.0		3.6	
$V_{VDDCORE}$	Output voltage, calibrated value			1.85		V
$V_{ACCURACY}$	Output voltage accuracy	I_{OUT} = 0.1mA to 100mA, V_{VDDIN_5} > 3V		5.8		%
I_{OUT}	DC output current				80	mA
I_{VREG}	Static current of regulator	Low power mode		34		µA

Table 40-19. Decoupling Requirements

Symbol	Parameter	Condition	Typ	Techno.	Units
C_{IN1}	Input regulator capacitor 1		1	NPO	nF
C_{IN2}	Input regulator capacitor 2		4.7	X7R	uF
C_{OUT1}	Output regulator capacitor 1		470	NPO	pf
C_{OUT2}	Output regulator capacitor 2		2.2	X7R	uF

40.8.2 3.3V Voltage Regulator Characteristics

Table 40-20. 3.3V Voltage Regulator Electrical Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{VDDIN_5}	Input voltage range		4.5		5.5	V
V_{VDDIN_33}	Output voltage, calibrated value			3.4		V
	Output voltage accuracy	I_{OUT} = 0.1mA to 40mA, V_{VDDIN} > 4.5V		3.3		%
I_{OUT}	DC output current				35	mA
I_{VREG}	Static current of regulator	Low power mode		45		µA

40.8.3 1.8V Brown Out Detector (BOD18) Characteristics

The values in Table 40-21 describe the values of the BOD.LEVEL in the SCIF module.

Table 40-21. BODLEVEL Values

BODLEVEL Value	Parameter	Min	Max	Units
0		1.38	1.52	V
20		1.44	1.58	
26	threshold at power-up sequence	1.48	1.66	
28		1.51	1.68	
32		1.55	1.73	
36		1.59	1.79	
40		1.64	1.83	

40.8.4 3.3V Brown Out Detector (BOD33) Characteristics

The values in Table 40-23 describe the values of the BOD33.LEVEL field in the SCIF module.

Table 40-23. BOD33.LEVEL Values

BOD33.LEVEL Value	Parameter	Min	Max	Units
17		2.26	2.51	V
22		2.33	2.63	
27		2.44	2.72	
31	threshold at power-up sequence	2.50	2.80	
33		2.54	2.84	
39		2.66	2.95	
44		2.74	3.06	
49		2.84	3.16	
53		2.91	3.24	

40.8.5 5V Brown Out Detector (BOD50) Characteristics

The values in Table 40-25 describe the values of the BOD50.LEVEL field in the SCIF module.

Table 40-25. BOD50.LEVEL Values

BOD50.LEVEL Value	Parameter	Min	Max	Units
16		3.31	3.62	V
25		3.54	3.88	
35		3.82	4.16	
44		4.05	4.43	
53		4.30	4.69	
61		4.52	4.92	

40.8.6 Analog to Digital Converter (ADC) and sample and hold (S/H) Characteristics

Table 40-27. ADC and S/H characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{ADC}	ADC clock frequency	12-bit resolution mode, V_{VDDANA} = 3V			1.2	MHz
		10-bit resolution mode, V_{VDDANA} = 3V			1.6	
		8-bit resolution mode, V_{VDDANA} = 3V			2.2	
		12-bit resolution mode, V_{VDDANA} = 4.5V			1.5	
		10-bit resolution mode, V_{VDDANA} = 4.5V			2	
		8-bit resolution mode, V_{VDDANA} = 4.5V			2.4	
$t_{STARTUP}$	Startup time	ADC cold start-up			1	ms
		ADC hot start-up			24	ADC clock cycles
t_{CONV}	Conversion time (latency)	(ADCIFA.SEQCFGn.SRES)/2 + 2, ADCIFA.CFG.SHD = 1	6		8	ADC clock cycles
		(ADCIFA.SEQCFGn.SRES)/2 + 3, ADCIFA.CFG.SHD = 0	7		9	
	Throughput rate	12-bit resolution, ADC clock = 1.2 MHz, V_{VDDANA} = 3V			1.2	MSPS
		10-bit resolution, ADC clock = 1.6 MHz, V_{VDDANA} = 3V			1.6	
		12-bit resolution, ADC clock = 1.5 MHz, V_{VDDANA} = 4.5V			1.5	
		10-bit resolution, ADC clock = 2 MHz, V_{VDDANA} = 4.5V			2	

Table 40-28. ADC Reference Voltage

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{ADCREF0}$	ADCREF0 input voltage range	5V Range	1		3.5	V
		3V Range	1		V_{VDDANA}-0.7	
$V_{ADCREF1}$	ADCREF1 input voltage range	5V Range	1		3.5	V
		3V Range	1		V_{VDDANA}-0.7	
	Internal 1V reference			1.0		V
	Internal 0.6*VDDANA reference			0.6*V_{VDDANA}		V

Table 40-29. ADC Decoupling requirements

Symbol	Parameter	Conditions	Min	Typ	Max	Units	Units
$C_{ADCREFPN}$	ADCREFP-ADCREFN capacitance			100			nF

Table 40-30. ADC Inputs

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{ADCINn}	ADC input voltage range		0		V_{VDDANA}	V
C_{ONCHIP}	Internal Capacitance	ADC used without S/H			5	pF
		ADC used with S/H			4	
R_{ONCHIP}	Switch resistance	ADC used without S/H			5.1	kΩ
		ADC used with S/H			4.6	

Figure 40-3. ADC input

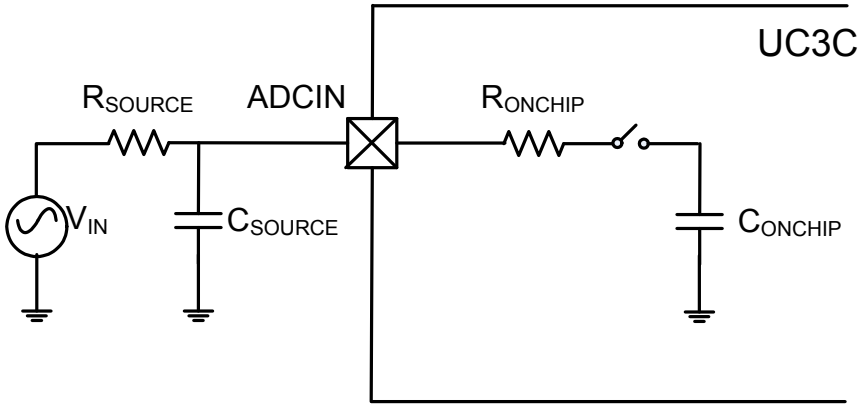


Table 40-31. ADC Transfer Characteristics 12-bit Resolution Mode[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 0			12	Bit
INL	Integral Non-Linearity			2.5		LSB
DNL	Differential Non-Linearity			1		LSB
	Offset error		-5		5	mV
	Gain error		-25		0	mV
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$ = 3V, ADCFIA.SEQCFGn.SRES = 0			12	Bit
INL	Integral Non-Linearity			3		LSB
DNL	Differential Non-Linearity			1.2		LSB
	Offset error		-30		30	mV
	Gain error		-25		0	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

Table 40-32. ADC Transfer Characteristics 10-bit Resolution Mode[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 1			10	Bit
INL	Integral Non-Linearity			0.7		LSB
DNL	Differential Non-Linearity			0.25		LSB
	Offset error		-5		5	mV
	Gain error		-25		0	mV
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$= 3V, ADCFIA.SEQCFGn.SRES = 1			10	Bit
INL	Integral Non-Linearity			0.8		LSB
DNL	Differential Non-Linearity			0.3		LSB
	Offset error		-30		30	mV
	Gain error		-25		0	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

Table 40-33. ADC Transfer Characteristics 8-bit Resolution Mode[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 2			8	Bit
INL	Integral Non-Linearity			0.4		LSB
DNL	Differential Non-Linearity			0.2		LSB
	Offset error		-5		5	mV
	Gain error		-25		0	mV
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$ = 3V, ADCFIA.SEQCFGn.SRES = 2			8	Bit
INL	Integral Non-Linearity			0.4		LSB
DNL	Differential Non-Linearity			0.2		LSB
	Offset error		-5		5	mV
	Gain error		-25		0	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

Table 40-34. ADC and S/H Transfer Characteristics 12-bit Resolution Mode and S/H gain = 1[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 0, S/H gain = 1			12	Bit
INL	Integral Non-Linearity			2.5		LSB
DNL	Differential Non-Linearity			1		LSB
	Offset error		-5		5	mV
	Gain error		-25		0	mV

Table 40-34. ADC and S/H Transfer Characteristics (Continued)12-bit Resolution Mode and S/H gain = 1[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$ = 3V, ADCFIA.SEQCFGn.SRES = 0, S/H gain = 1			12	Bit
INL	Integral Non-Linearity			3		LSB
DNL	Differential Non-Linearity			1.2		LSB
	Offset error		-30		30	mV
	Gain error		-25		0	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

40.8.7 Digital to Analog Converter (DAC) Characteristics

Table 40-35. Channel Conversion Time and DAC Clock

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{DAC}	DAC clock frequency				1	MHz
$t_{STARTUP}$	Startup time				3	µs
t_{CONV}	Conversion time (latency)	No S/H enabled, internal DAC			1	µs
		One S/H			1.5	µs
		Two S/H			2	µs
	Throughput rate				$1/t_{CONV}$	MSPS

Table 40-36. External Voltage Reference Input

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{DACREF}	DACREF input voltage range		1.2		V_{VDDANA}-0.7	V

Table 40-37. DAC Outputs

Symbol	Parameter	Conditions	Min	Typ	Max	Units
	Output range		0.2		V_{DACREF}[1]	
C_{LOAD}	Output capacitance		0		100	pF
R_{LOAD}	Output resitance		2			kΩ

Note: 1. DACREF corresponds to the internal or external DAC reference voltage depending on the DACREF settings

Figure 40-4. DAC output

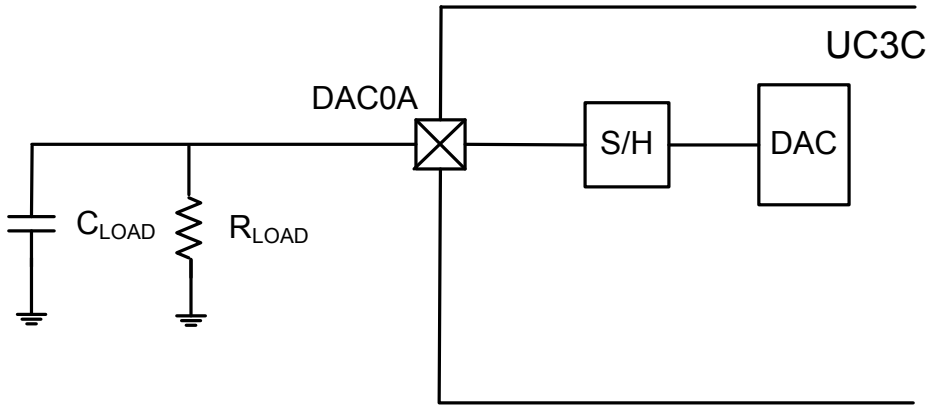


Table 40-38. Transfer Characteristics[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	V_{VDDANA} = 3V, V_{DACREF} = 2V, One S/H			12	Bit
INL	Integral Non-Linearity			8		LSB
DNL	Differential Non-linearity			6		LSB
	Offset error		-30		30	mV
	Gain error		-30		30	mV
RES	Resolution	V_{VDDANA} = 5V, V_{DACREF} = 3V, One S/H			12	Bit
INL	Integral Non-Linearity			12		LSB
DNL	Differential Non-linearity			6		LSB
	Offset error		-30		30	mV
	Gain error		-30		30	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

40.8.8 Analog Comparator Characteristics

Table 40-39. Analog Comparator Characteristics[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
	Positive input voltage range		0		V_{VDDANA}	V
	Negative input voltage range		0		V_{VDDANA}	V
V_{OFFSET}	Offset	No hysteresis, Low Power mode	-90		90	mV
		No hysteresis, High Speed mode	-25		25	mV
V_{HYST}	Hysteresis	Low hysteresis, Low Power mode			100	mV
		Low hysteresis, High Speed mode			90	
		High hysteresis, Low Power mode			230	mV
		High hysteresis, High Speed mode			150	
t_{DELAY}	Propagation delay	Low Power mode			10	us
		High Speed mode			0.7	
$t_{STARTUP}$	Start-up time				20	µs

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

Table 40-40. VDDANA scaled reference

Symbol	Parameter	Min	Typ	Max	Units
SCF	ACIFA.SCFi.SCF range	0		48	
V_{VDDANA} scaled			(64 - SCF) * V_{VDDANA} / 65		V
	V_{VDDANA} voltage accuracy			2.5	%

40.8.9 USB Transceiver Characteristics

40.8.9.1 Electrical Characteristics

Table 40-41. Electrical Parameters

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
R_{EXT}	Recommended external USB series resistor	In series with each USB pin with ±5%		39		Ω

The USB on-chip buffers comply with the Universal Serial Bus (USB) v2.0 standard. All AC parameters related to these buffers can be found within the USB 2.0 electrical specifications.

40.9 Timing Characteristics

40.9.1 Startup, Reset, and Wake-up Timing

The startup, reset, and wake-up timings are calculated using the following formula:

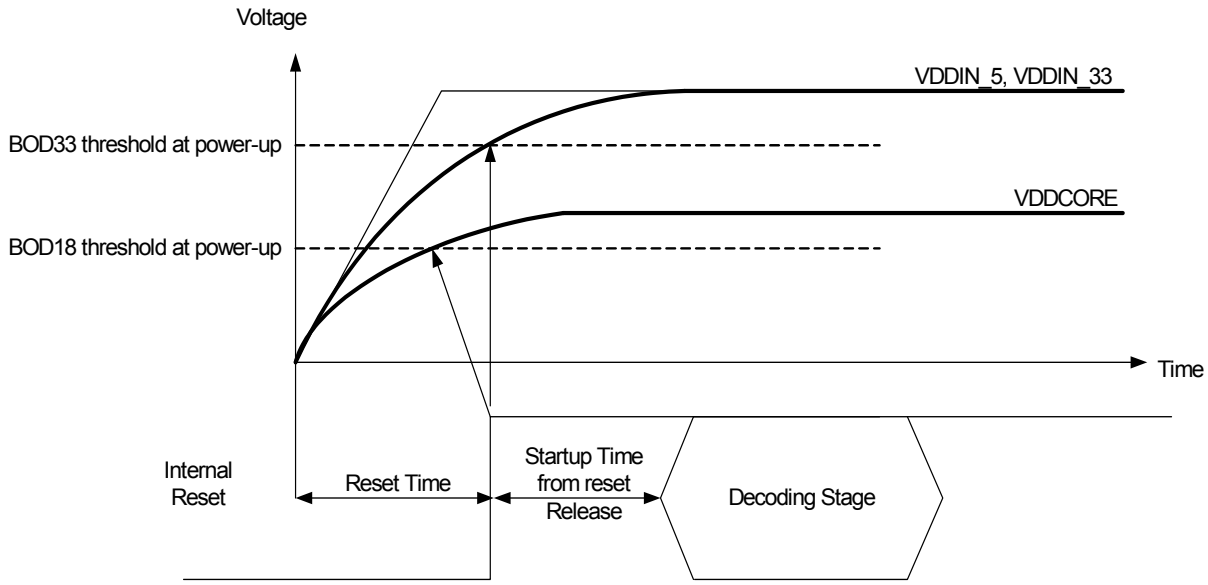
$$t = t_{CONST} + N_{CPU} \times t_{CPU}$$

Where t_{CONST} and N_{CPU} are found in Table 40-42. t_{CONST} is the delay relative to RCSYS, t_{CPU} is the period of the CPU clock. If another clock source than RCSYS is selected as CPU clock the startup time of the oscillator, $t_{OSCSTART}$, must be added to the wake-up time in the stop, deepstop, and static sleep modes. Please refer to the source for the CPU clock in the "Oscillator Characteristics" on page 1256 for more details about oscillator startup times.

Table 40-42. Maximum Reset and Wake-up Timing

Parameter		Measuring	Max t_{CONST} (in µs)	Max N_{CPU}
Startup time from power-up, using regulator		VDDIN_5 rising (10 mV/ms) Time from V_{VDDIN_5}=0 to the first instruction entering the decode stage of CPU. VDDCORE is supplied by the internal regulator.	2600	0
Startup time from reset release		Time from releasing a reset source (except POR, BOD18, and BOD33) to the first instruction entering the decode stage of CPU.	1240	0
Wake-up	Idle	From wake-up event to the first instruction entering the decode stage of the CPU.	0	19
	Frozen		268	209
	Standby		268	209
	Stop		268+ $t_{OSCSTART}$	212
	Deepstop		268+ $t_{OSCSTART}$	212
	Static		268+ $t_{OSCSTART}$	212

Figure 40-5. Startup and Reset Time



40.9.2 RESET_N characteristics

Table 40-43. RESET_N Clock Waveform Parameters

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{RESET}	RESET_N minimum pulse length		2 * T_{RCSYS}			clock cycles

40.9.3 USART in SPI Mode Timing

40.9.3.1 Master mode

Figure 40-6. USART in SPI Master Mode With (CPOL= CPHA= 0) or (CPOL= CPHA= 1)

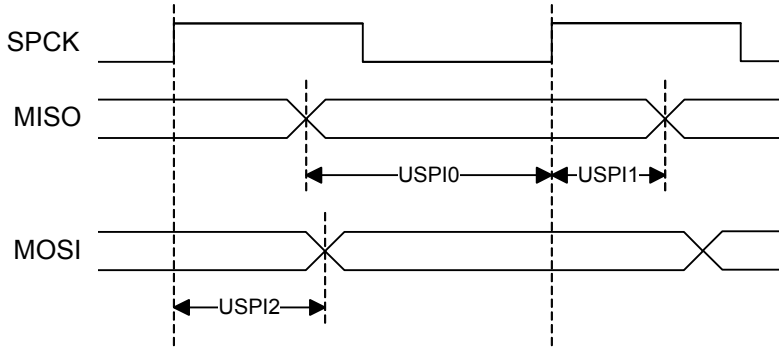


Figure 40-7. USART in SPI Master Mode With (CPOL= 0 and CPHA= 1) or (CPOL= 1 and CPHA= 0)

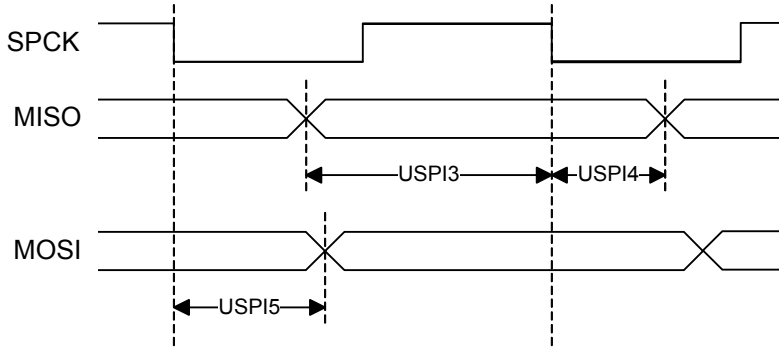


Table 40-44. USART in SPI Mode Timing, Master Mode[1]

Symbol	Parameter	Conditions	Min	Max	Units
USPI0	MISO setup time before SPCK rises	external capacitor = 40pF	26+ t_{SAMPLE}[2]		ns
USPI1	MISO hold time after SPCK rises		0		ns
USPI2	SPCK rising to MOSI delay			11	ns
USPI3	MISO setup time before SPCK falls		26+ t_{SAMPLE}[2]		ns
USPI4	MISO hold time after SPCK falls		0		ns
USPI5	SPCK falling to MOSI delay			11.5	ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

2. Where: $t_{SAMPLE} = t_{SPCK} - \left(\left\lfloor \frac{t_{SPCK}}{2 \times t_{CLKUSART}} \right\rfloor \frac{1}{2} \right) \times t_{CLKUSART}$

Maximum SPI Frequency, Master Output

The maximum SPI master output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{PINMAX}, \frac{1}{SPIn}, \frac{f_{CLKSPI} \times 2}{9})$$

Where $SPIn$ is the MOSI delay, USPI2 or USPI5 depending on CPOL and NCPHA. f_{PINMAX} is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins. f_{CLKSPI} is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

Maximum SPI Frequency, Master Input

The maximum SPI master input frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(\frac{1}{SPIn + t_{VALID}}, \frac{f_{CLKSPI} \times 2}{9})$$

Where $SPIn$ is the MISO setup and hold time, USPI0 + USPI1 or USPI3 + USPI4 depending on CPOL and NCPHA. T_{VALID} is the SPI slave response time. Please refer to the SPI slave datasheet for T_{VALID} . f_{CLKSPI} is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

40.9.3.2 Slave mode

Figure 40-8. USART in SPI Slave Mode With (CPOL= 0 and CPHA= 1) or (CPOL= 1 and CPHA= 0)

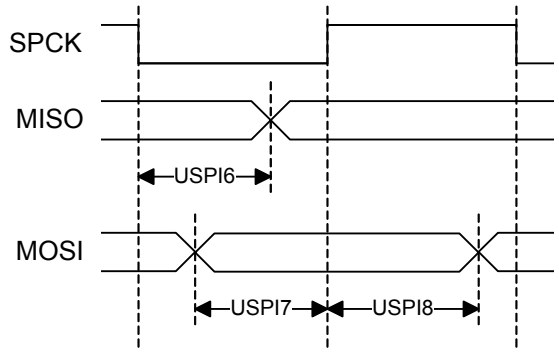


Figure 40-9. USART in SPI Slave Mode With (CPOL= CPHA= 0) or (CPOL= CPHA= 1)

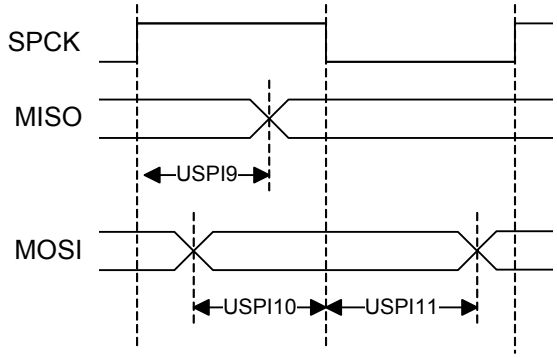


Figure 40-10. USART in SPI Slave Mode NPCS Timing

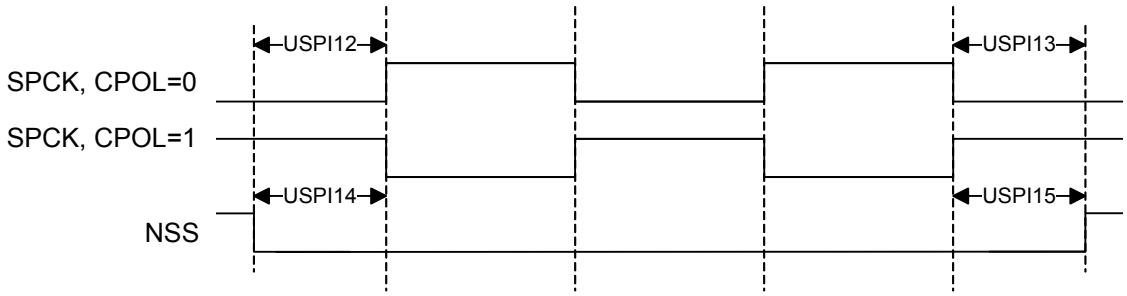


Table 40-45. USART in SPI mode Timing, Slave Mode[1]

Symbol	Parameter	Conditions	Min	Max	Units
USPI6	SPCK falling to MISO delay	external capacitor = 40pF		27	ns
USPI7	MOSI setup time before SPCK rises		t_{SAMPLE}[2] + t_{CLK_USART}		ns
USPI8	MOSI hold time after SPCK rises		0		ns
USPI9	SPCK rising to MISO delay			28	ns
USPI10	MOSI setup time before SPCK falls		t_{SAMPLE}[2] + t_{CLK_USART}		ns
USPI11	MOSI hold time after SPCK falls		0		ns
USPI12	NSS setup time before SPCK rises		33		ns
USPI13	NSS hold time after SPCK falls		0		ns
USPI14	NSS setup time before SPCK falls		33		ns
USPI15	NSS hold time after SPCK rises		0		ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

2. Where: $t_{SAMPLE} = t_{SPCK} - \left(\left\lfloor \frac{t_{SPCK}}{2 \times t_{CLKUSART}} \right\rfloor + \frac{1}{2} \right) \times t_{CLKUSART}$

Maximum SPI Frequency, Slave Input Mode

The maximum SPI slave input frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(\frac{f_{CLKSPI} \times 2}{9}, \frac{1}{SPIn})$$

Where $SPIn$ is the MOSI setup and hold time, USPI7 + USPI8 or USPI10 + USPI11 depending on CPOL and NCPHA. f_{CLKSPI} is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

Maximum SPI Frequency, Slave Output Mode

The maximum SPI slave output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(\frac{f_{CLKSPI} \times 2}{9}, f_{PINMAX}, \frac{1}{SPIn + t_{SETUP}})$$

Where $SPIn$ is the MISO delay, USPI6 or USPI9 depending on CPOL and NCPHA. T_{SETUP} is the SPI master setup time. Please refer to the SPI masterdatasheet for T_{SETUP} . f_{CLKSPI} is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock. f_{PINMAX} is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins.

40.9.4 SPI Timing

40.9.4.1 Master mode

Figure 40-11. SPI Master Mode With (CPOL= NCPHA= 0) or (CPOL= NCPHA= 1)

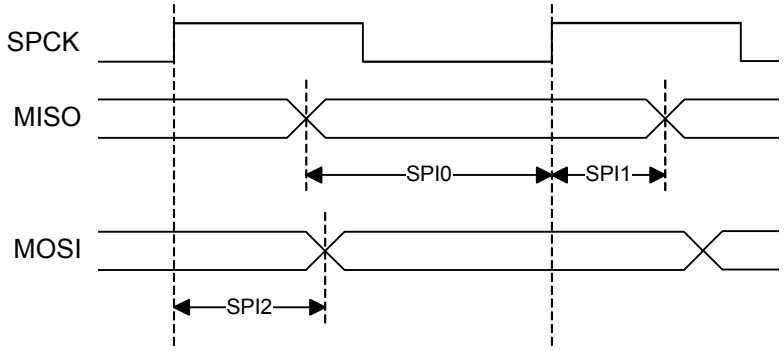


Figure 40-12. SPI Master Mode With (CPOL= 0 and NCPHA= 1) or (CPOL= 1 and NCPHA= 0)

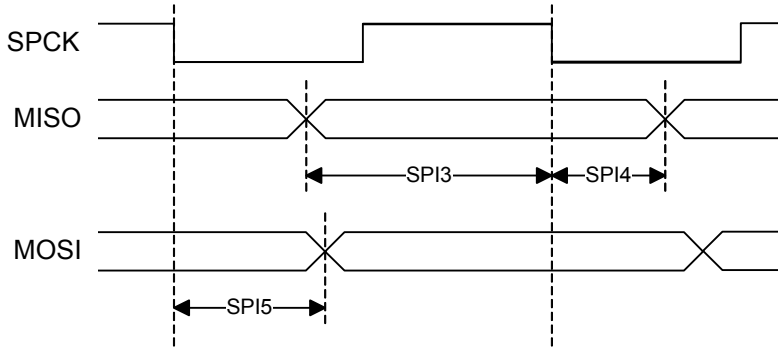


Table 40-46. SPI Timing, Master Mode[1]

Symbol	Parameter	Conditions	Min	Max	Units
SPI0	MISO setup time before SPCK rises	external capacitor = 40pF	28.5+ (t_{CLK_SPI})/2		ns
SPI1	MISO hold time after SPCK rises		0		ns
SPI2	SPCK rising to MOSI delay			10.5	ns
SPI3	MISO setup time before SPCK falls		28.5 + (t_{CLK_SPI})/2		ns
SPI4	MISO hold time after SPCK falls		0		ns
SPI5	SPCK falling to MOSI delay			10.5	ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Maximum SPI Frequency, Master Output

The maximum SPI master output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{PINMAX}, \frac{1}{SPIn})$$

Where $SPIn$ is the MOSI delay, SPI2 or SPI5 depending on CPOL and NCPHA. f_{PINMAX} is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins.

Maximum SPI Frequency, Master Input

The maximum SPI master input frequency is given by the following formula:

$$f_{SPCKMAX} = \frac{1}{SPIn + t_{VALID}}$$

Where $SPIn$ is the MISO setup and hold time, SPI0 + SPI1 or SPI3 + SPI4 depending on CPOL and NCPHA. t_{VALID} is the SPI slave response time. Please refer to the SPI slave datasheet for t_{VALID} .

40.9.4.2 Slave mode

Figure 40-13. SPI Slave Mode With (CPOL= 0 and NCPHA= 1) or (CPOL= 1 and NCPHA= 0)

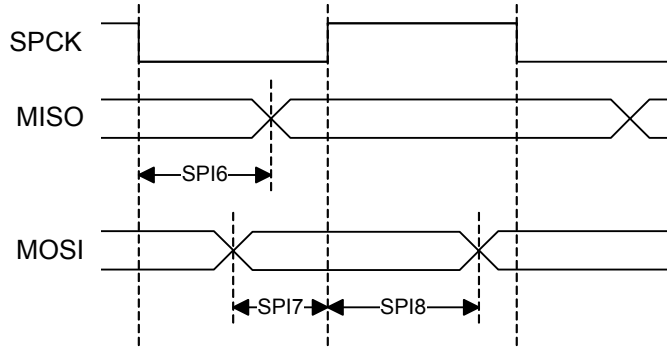


Figure 40-14. SPI Slave Mode With (CPOL= NCPHA= 0) or (CPOL= NCPHA= 1)

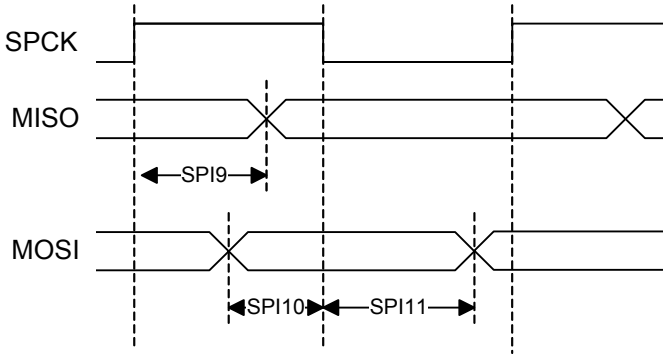


Figure 40-15. SPI Slave Mode NPCS Timing

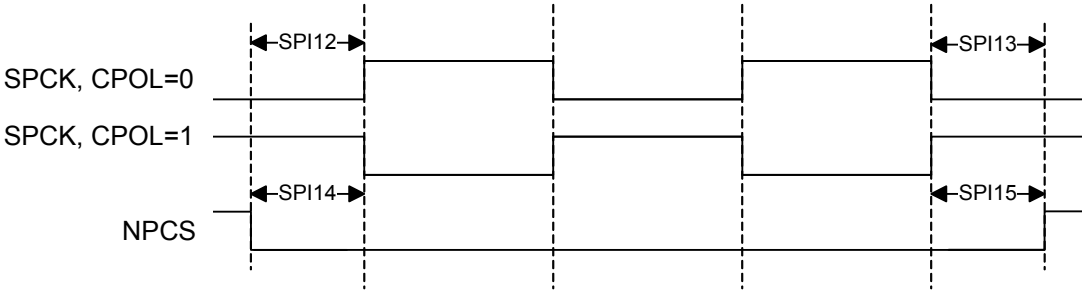


Table 40-47. SPI Timing, Slave Mode[1]

Symbol	Parameter	Conditions	Min	Max	Units
SPI6	SPCK falling to MISO delay	external capacitor = 40pF		29	ns
SPI7	MOSI setup time before SPCK rises		0		ns
SPI8	MOSI hold time after SPCK rises		6.5		ns
SPI9	SPCK rising to MISO delay			30	ns
SPI10	MOSI setup time before SPCK falls		0		ns
SPI11	MOSI hold time after SPCK falls		5		ns
SPI12	NPCS setup time before SPCK rises		0		ns
SPI13	NPCS hold time after SPCK falls		1.5		ns
SPI14	NPCS setup time before SPCK falls		0		ns
SPI15	NPCS hold time after SPCK rises		1.5		ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Maximum SPI Frequency, Slave Input Mode

The maximum SPI slave input frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{CLKSPI}, \frac{1}{SPIn})$$

Where $SPIn$ is the MOSI setup and hold time, SPI7 + SPI8 or SPI10 + SPI11 depending on CPOL and NCPHA. f_{CLKSPI} is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

Maximum SPI Frequency, Slave Output Mode

The maximum SPI slave output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{PINMAX}, \frac{1}{SPIn + t_{SETUP}})$$

Where $SPIn$ is the MISO delay, SPI6 or SPI9 depending on CPOL and NCPHA. t_{SETUP} is the SPI master setup time. Please refer to the SPI masterdatasheet for t_{SETUP} .f_{PINMAX} is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins.

40.9.5 TWIM/TWIS Timing

Figure 40-48 shows the TWI-bus timing requirements and the compliance of the device with them. Some of these requirements (t_r and t_f) are met by the device without requiring user intervention. Compliance with the other requirements (t_{HD-STA}, t_{SU-STA}, t_{SU-STO}, t_{HD-DAT}, $t_{SU-DAT-I2C}$, $t_{LOW-I2C}$, t_{HIGH}, and f_{TWCK}) requires user intervention through appropriate programming of the relevant

TWIM and TWIS user interface registers. Please refer to the TWIM and TWIS sections for more information.

Table 40-48. TWI-Bus Timing Requirements

Symbol	Parameter	Mode	Minimum		Maximum		Unit
			Requirement	Device	Requirement	Device	
t_r	TWCK and TWD rise time	Standard(1)	-		1000		ns
		Fast(1)	20 + 0.1 C_b		300		
t_f	TWCK and TWD fall time	Standard(1)	-		300		ns
		Fast(1)	20 + 0.1 C_b		300		
t_{HD-STA}	(Repeated) START hold time	Standard(1)	4.0	t_{clkpb}	-		µs
		Fast(1)	0.6				
t_{SU-STA}	(Repeated) START set-up time	Standard(1)	4.7	t_{clkpb}	-		µs
		Fast(1)	0.6				
t_{SU-STO}	STOP set-up time	Standard(1)	4.0	$4t_{clkpb}$	-		µs
		Fast(1)	0.6				
t_{HD-DAT}	Data hold time	Standard(1)	0.3(2)	$2t_{clkpb}$	3.45	??	µs
		Fast(1)			0.9		
$t_{SU-DAT-I2C}$	Data set-up time	Standard(1)	250	$2t_{clkpb}$	-		ns
		Fast(1)	100				
t_{SU-DAT}		-	-	t_{clkpb}	-		-
$t_{LOW-I2C}$	TWCK LOW period	Standard(1)	4.7	$4t_{clkpb}$	-		µs
		Fast(1)	1.3				
t_{LOW}		-	-	t_{clkpb}	-		-
t_{HIGH}	TWCK HIGH period	Standard(1)	4.0	$8t_{clkpb}$	-		µs
		Fast(1)	0.6				
f_{TWCK}	TWCK frequency	Standard(1)	-		100	$\frac{1}{12t_{clkpb}}$	kHz
		Fast(1)			400		

Notes: 1. Standard mode: $f_{TWCK} \leq 100$ kHz ; fast mode: $f_{TWCK} > 100$ kHz .
2. A device must internally provide a hold time of at least 300 ns for TWD with reference to the falling edge of TWCK.

Notations:

C_b = total capacitance of one bus line in pF
t_{clkpb} = period of TWI peripheral bus clock
$t_{prescaled}$ = period of TWI internal prescaled clock (see chapters on TWIM and TWIS)
The maximum $t_{HD;DAT}$ has only to be met if the device does not stretch the LOW period ($t_{LOW-I2C}$) of TWCK.

40.9.6 JTAG Timing

Figure 40-16. JTAG Interface Signals

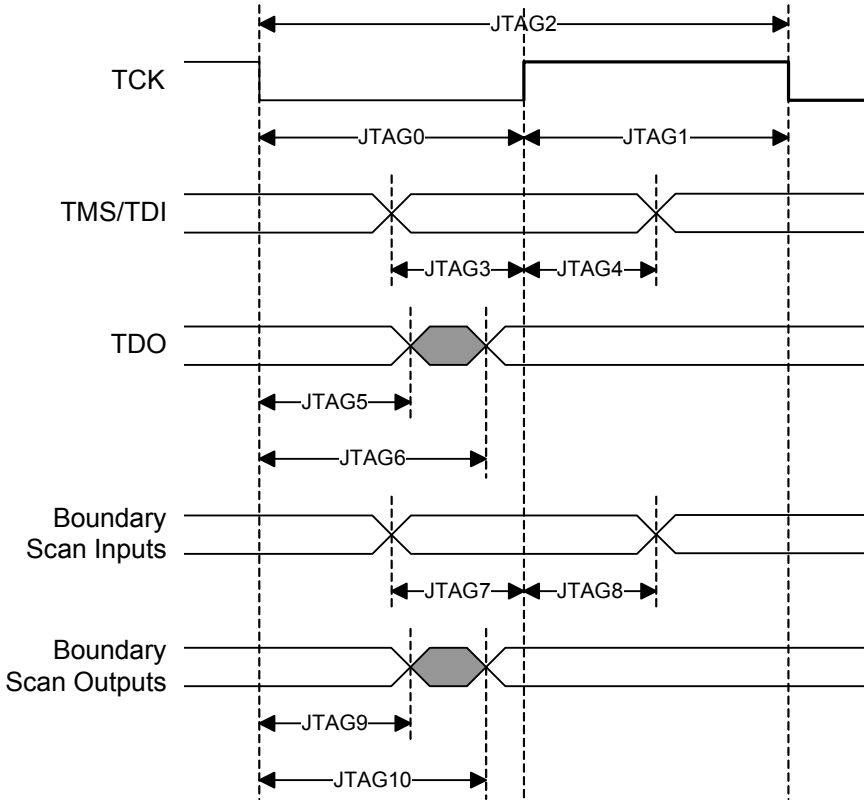


Table 40-49. JTAG Timings[1]

Symbol	Parameter	Conditions	Min	Max	Units
JTAG0	TCK Low Half-period	external capacitor = 40pF	21.5		ns
JTAG1	TCK High Half-period		8.5		ns
JTAG2	TCK Period		29		ns
JTAG3	TDI, TMS Setup before TCK High		6.5		ns
JTAG4	TDI, TMS Hold after TCK High		0		ns
JTAG5	TDO Hold Time		12.5		ns
JTAG6	TCK Low to TDO Valid			21.5	ns
JTAG7	Boundary Scan Inputs Setup Time		0		ns
JTAG8	Boundary Scan Inputs Hold Time		4.5		ns
JTAG9	Boundary Scan Outputs Hold Time		11		ns
JTAG10	TCK to Boundary Scan Outputs Valid			18	ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

40.9.7 EBI Timings

See EBI I/O lines description for more details.

Table 40-50. SMC Clock Signal.

Symbol	Parameter	Max[1]	Units
$1/(t_{CPSMC})$	SMC Controller clock frequency	f_{cpu}	MHz

Note: 1. The maximum frequency of the SMC interface is the same as the max frequency for the HSB.

Table 40-51. SMC Read Signals with Hold Settings[1]

Symbol	Parameter	Conditions	Min	Units
NRD Controlled (READ_MODE = 1)				
SMC_1	Data setup before NRD high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	32.5	ns
SMC_2	Data hold after NRD high		0	
SMC_3	NRD high to NBS0/A0 change[2]		nrd hold length * t_{CPSMC} - 1.5	
SMC_4	NRD high to NBS1 change[2]		nrd hold length * t_{CPSMC} - 0	
SMC_5	NRD high to NBS2/A1 change[2]		nrd hold length * t_{CPSMC} - 0	
SMC_7	NRD high to A2 - A25 change[2]		nrd hold length * t_{CPSMC} - 5.6	
SMC_8	NRD high to NCS inactive[2]		(nrd hold length - ncs rd hold length) * t_{CPSMC} - 1.3	
SMC_9	NRD pulse width		nrd pulse length * t_{CPSMC} - 0.6	
NRD Controlled (READ_MODE = 0)				
SMC_{10}	Data setup before NCS high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	34.1	ns
SMC_{11}	Data hold after NCS high		0	
SMC_{12}	NCS high to NBS0/A0 change[2]		ncs rd hold length * t_{CPSMC} - 3	
SMC_{13}	NCS high to NBS0/A0 change[2]		ncs rd hold length * t_{CPSMC} - 2	
SMC_{14}	NCS high to NBS2/A1 change[2]		ncs rd hold length * t_{CPSMC} - 1.1	
SMC_{16}	NCS high to A2 - A25 change[2]		ncs rd hold length * t_{CPSMC} - 7.2	
SMC_{17}	NCS high to NRD inactive[2]		(ncs rd hold length - nrd hold length) * t_{CPSMC} - 2.2	
SMC_{18}	NCS pulse width		ncs rd pulse length * t_{CPSMC} - 3	

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

2. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs rd hold length” or “nrd hold length”.

Table 40-52. SMC Read Signals with no Hold Settings[1]

Symbol	Parameter	Conditions	Min	Units
NRD Controlled (READ_MODE = 1)				
SMC_{19}	Data setup before NRD high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	32.5	ns
SMC_{20}	Data hold after NRD high		0	
NRD Controlled (READ_MODE = 0)				
SMC_{21}	Data setup before NCS high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	28.5	ns
SMC_{22}	Data hold after NCS high		0	

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Table 40-53. SMC Write Signals with Hold Settings[1]

Symbol	Parameter	Conditions	Min	Units
NRD Controlled (READ_MODE = 1)				
SMC_{23}	Data Out valid before NWE high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	(nwe pulse length - 1) * t_{CPSMC} - 1.4	ns
SMC_{24}	Data Out valid after NWE high[2]		nwe pulse length * t_{CPSMC} - 4.7	
SMC_{25}	NWE high to NBS0/A0 change[2]		nwe pulse length * t_{CPSMC} - 2.7	
SMC_{29}	NWE high to NBS2/A1 change[2]		nwe pulse length * t_{CPSMC} - 0.7	
SMC_{31}	NWE high to A2 - A25 change[2]		nwe pulse length * t_{CPSMC} - 6.8	
SMC_{32}	NWE high to NCS inactive[2]		(nwe hold pulse - ncs wr hold length) * t_{CPSMC} - 2.5	
SMC_{33}	NWE pulse width		nwe pulse length * t_{CPSMC} - 0.2	
NRD Controlled (READ_MODE = 0)				
SMC_{34}	Data Out valid before NCS high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	(ncs wr pulse length - 1) * t_{CPSMC} - 2.2	ns
SMC_{35}	Data Out valid after NCS high[2]		ncs wr hold length * t_{CPSMC} - 5.1	
SMC_{36}	NCS high to NWE inactive[2]		(ncs wr hold length - nwe hold length) * t_{CPSMC} - 2	

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

2. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “nwe hold length”

Table 40-54. SMC Write Signals with No Hold Settings (NWE Controlled only)[1]

Symbol	Parameter	Conditions	Min	Units
SMC_{37}	NWE rising to A2-A25 valid	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	8.7	ns
SMC_{38}	NWE rising to NBS0/A0 valid		7.6	
SMC_{40}	NWE rising to A1/NBS2 change		8.7	
SMC_{42}	NWE rising to NCS rising		8.4	
SMC_{43}	Data Out valid before NWE rising		(nwe pulse length - 1) * t_{CPSMC} - 1.2	
SMC_{44}	Data Out valid after NWE rising		8.4	
SMC_{45}	NWE pulse width		nwe pulse length * t_{CPSMC} - 0	

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Figure 40-17. SMC Signals for NCS Controlled Accesses

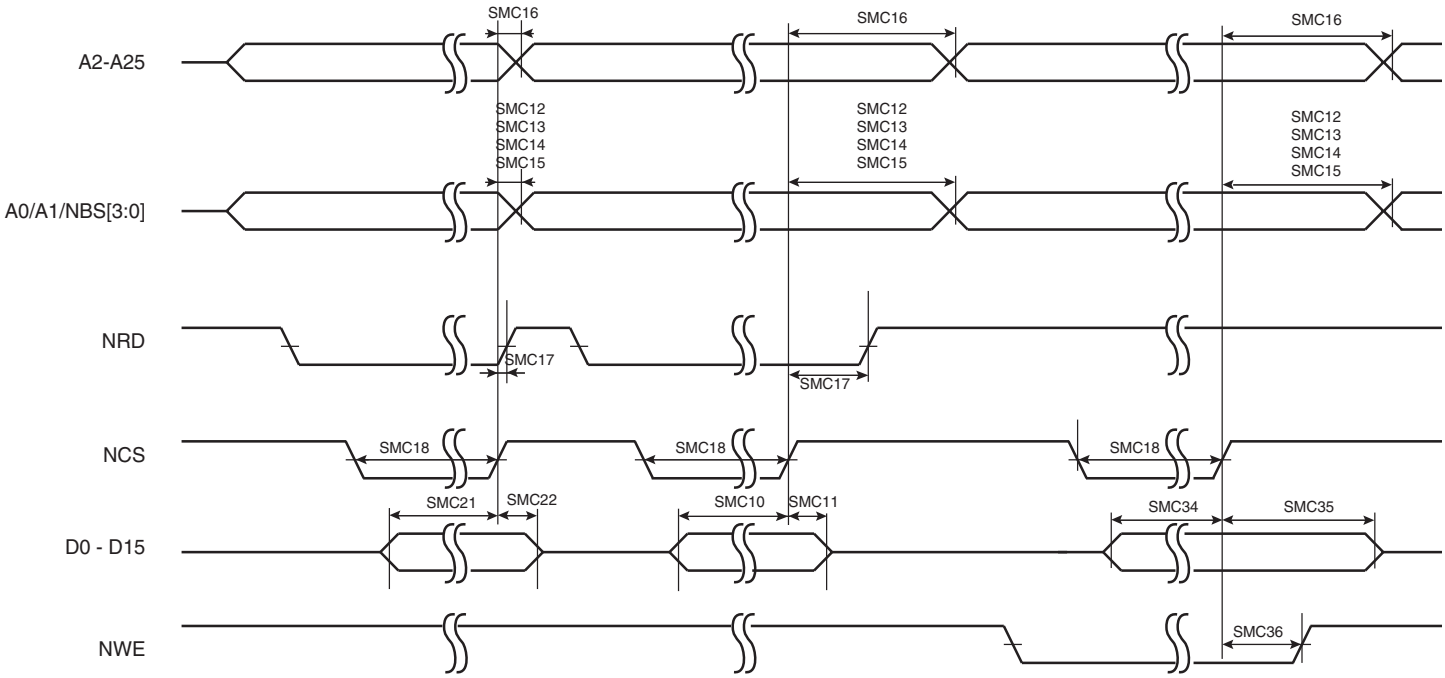
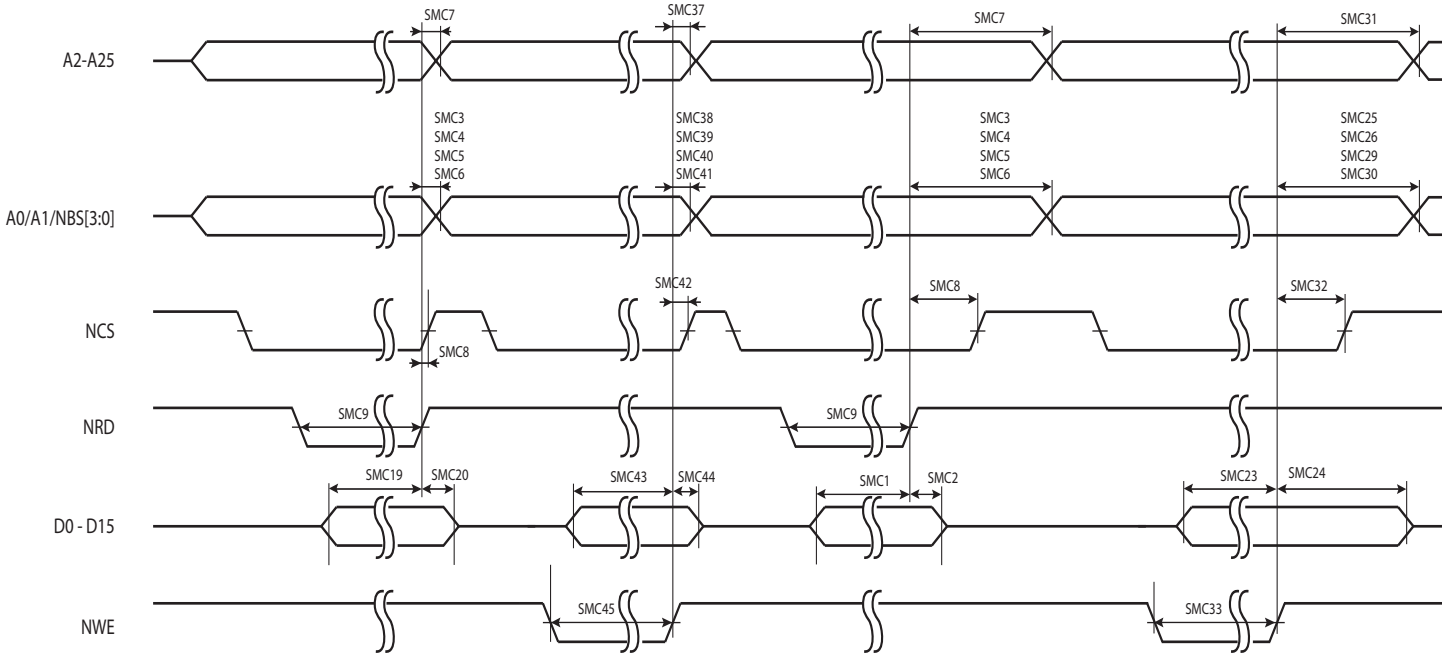


Figure 40-18. SMC Signals for NRD and NRW Controlled Accesses[1]



Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

40.9.8 SDRAM Signals

Table 40-55. SDRAM Clock Signal

Symbol	Parameter	Max[1]	Units
$1/(t_{CPSDCK})$	SDRAM Controller clock frequency	f_{cpu}	MHz

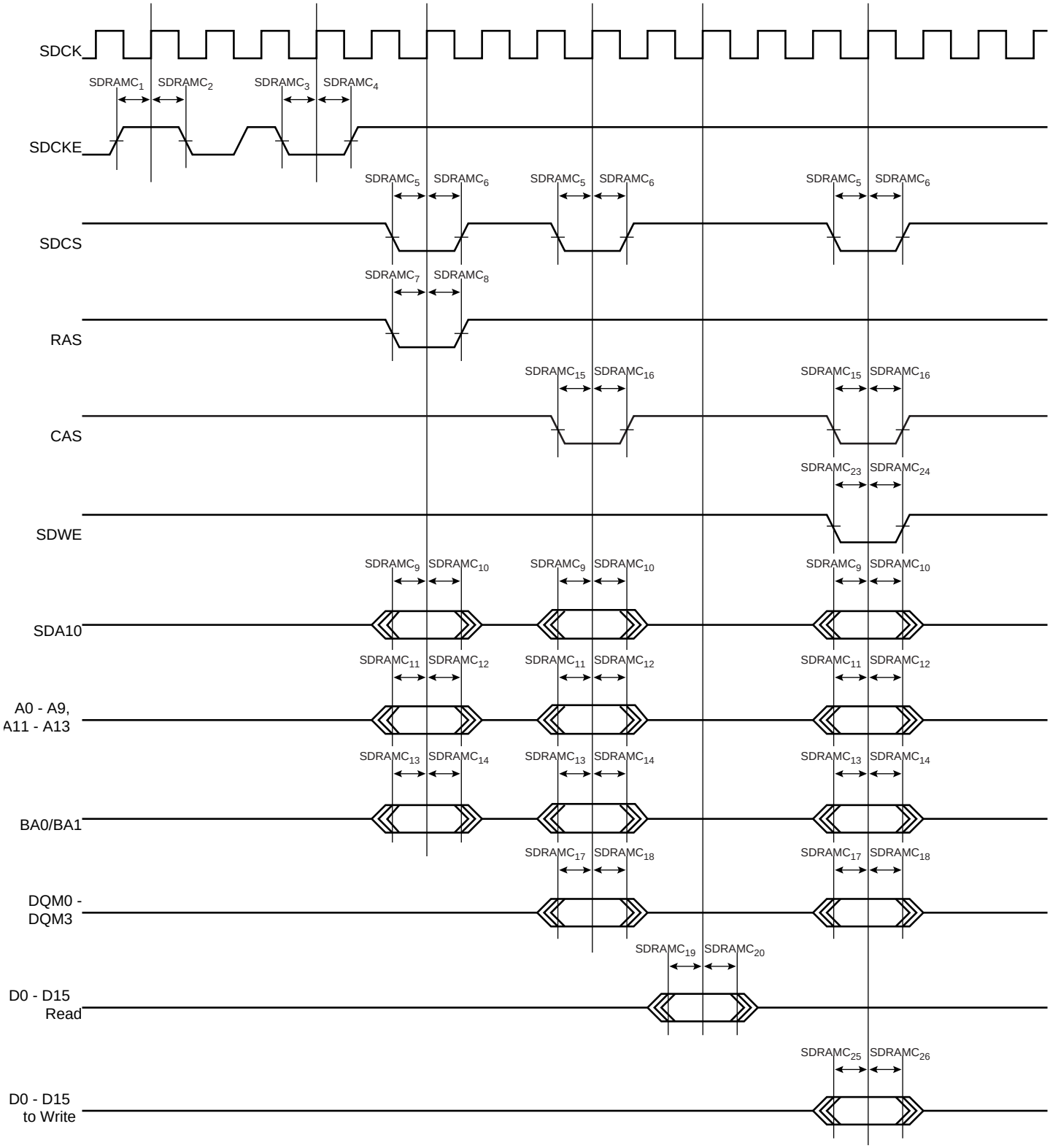
Note: 1. The maximum frequency of the SDRAMC interface is the same as the max frequency for the HSB.

Table 40-56. SDRAM Signal[1]

Symbol	Parameter	Conditions	Min	Units
$SDRAMC_1$	SDCKE high before SDCK rising edge	V_{VDD} = 3.0V, drive strength of the pads set to the highest, external capacitor = 40 pF on SDRAM pins except 8 pF on SDCK pins	5.6	ns
$SDRAMC_2$	SDCKE low after SDCK rising edge		7.3	
$SDRAMC_3$	SDCKE low before SDCK rising edge		6.8	
$SDRAMC_4$	SDCKE high after SDCK rising edge		8.3	
$SDRAMC_5$	SDCS low before SDCK rising edge		6.1	
$SDRAMC_6$	SDCS high after SDCK rising edge		8.4	
$SDRAMC_7$	RAS low before SDCK rising edge		7	
$SDRAMC_8$	RAS high after SDCK rising edge		7.7	
$SDRAMC_9$	SDA10 change before SDCK rising edge		6.4	
$SDRAMC_{10}$	SDA10 change after SDCK rising edge		7.1	
$SDRAMC_{11}$	Address change before SDCK rising edge		4.7	
$SDRAMC_{12}$	Address change after SDCK rising edge		4.4	
$SDRAMC_{13}$	Bank change before SDCK rising edge		6.2	
$SDRAMC_{14}$	Bank change after SDCK rising edge		6.9	
$SDRAMC_{15}$	CAS low before SDCK rising edge		6.6	
$SDRAMC_{16}$	CAS high after SDCK rising edge		7.8	
$SDRAMC_{17}$	DQM change before SDCK rising edge		6	
$SDRAMC_{18}$	DQM change after SDCK rising edge		6.7	
$SDRAMC_{19}$	D0-D15 in setup before SDCK rising edge		6.4	
$SDRAMC_{20}$	D0-D15 in hold after SDCK rising edge		0	
$SDRAMC_{23}$	SDWE low before SDCK rising edge		7	
$SDRAMC_{24}$	SDWE high after SDCK rising edge		7.4	
$SDRAMC_{25}$	D0-D15 Out valid before SDCK rising edge		5.2	
$SDRAMC_{26}$	D0-D15 Out valid after SDCK rising edge		5.6	

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Figure 40-19. SDRAMC Signals relative to SDCK.



40.9.9 MACB Characteristics

Table 40-57. Ethernet MAC Signals[1]

Symbol	Parameter	Conditions	Min.	Max.	Unit
MAC_1	Setup for MDIO from MDC rising	V_{VDD} = 3.0V, drive strength of the pads set to the highest, external capacitor = 10pF on MACB pins	0	2.5	ns
MAC_2	Hold for MDIO from MDC rising		0	0.7	ns
MAC_3	MDIO toggling from MDC falling		0	1.1	ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Table 40-58. Ethernet MAC MII Specific Signals[1]

Symbol	Parameter	Conditions	Min.	Max.	Unit
MAC_4	Setup for COL from TX_CLK rising	V_{VDD} = 3.0V, drive strength of the pads set to the highest, external capacitor = 10pF on MACB pins	0		ns
MAC_5	Hold for COL from TX_CLK rising		0		ns
MAC_6	Setup for CRS from TX_CLK rising		0.5		ns
MAC_7	Hold for CRS from TX_CLK rising		0.5		ns
MAC_8	TX_ER toggling from TX_CLK rising		16.4	18.6	ns
MAC_9	TX_EN toggling from TX_CLK rising		14.5	15.3	ns
MAC_{10}	TXD toggling from TX_CLK rising		13.9	18.2	ns
MAC_{11}	Setup for RXD from RX_CLK		1.3		ns
MAC_{12}	Hold for RXD from RX_CLK		1.8		ns
MAC_{13}	Setup for RX_ER from RX_CLK		3.4		ns
MAC_{14}	Hold for RX_ER from RX_CLK		0		ns
MAC_{15}	Setup for RX_DV from RX_CLK		0.7		ns
MAC_{16}	Hold for RX_DV from RX_CLK		1.3n		ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Figure 40-20. Ethernet MAC MII Mode

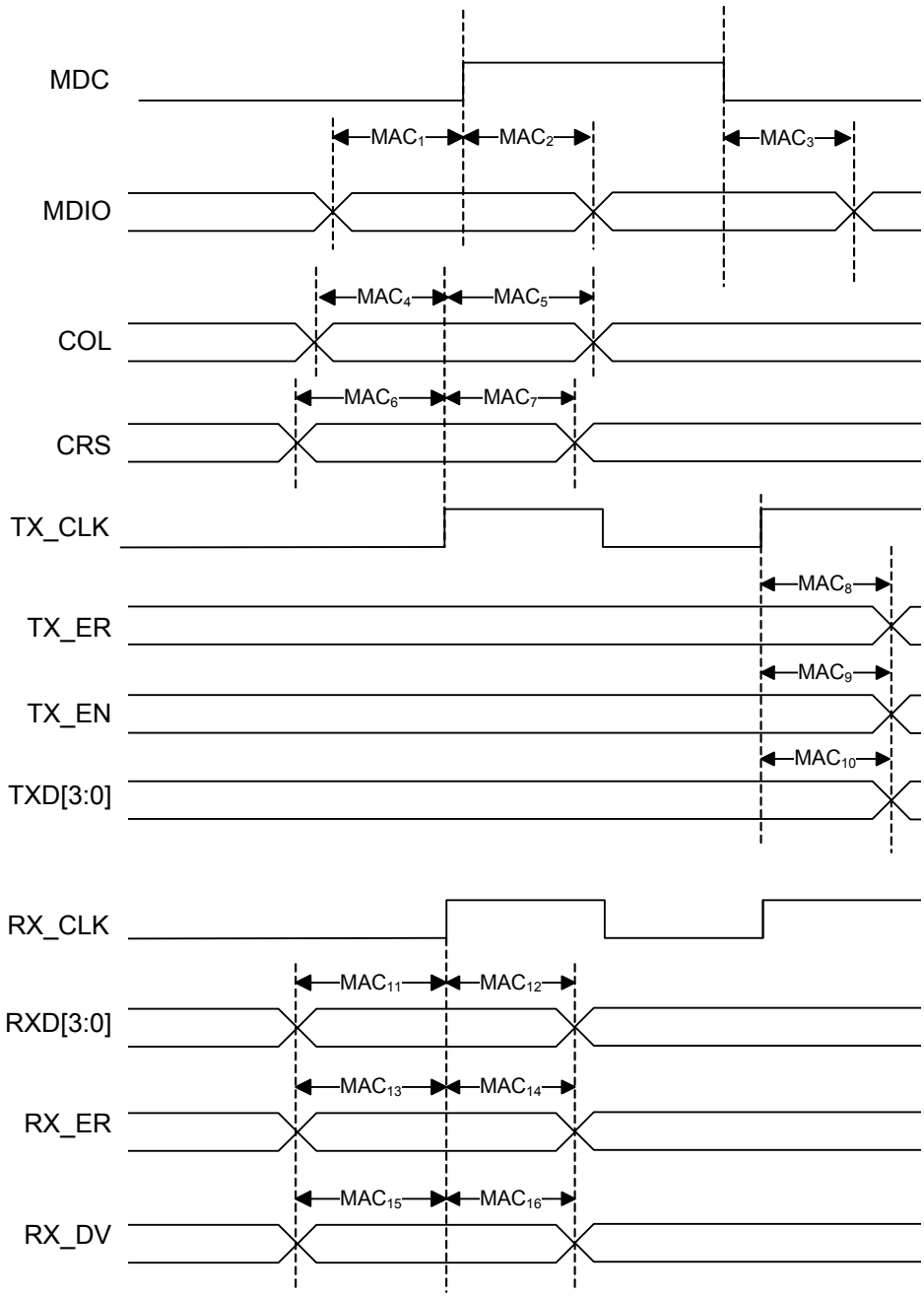
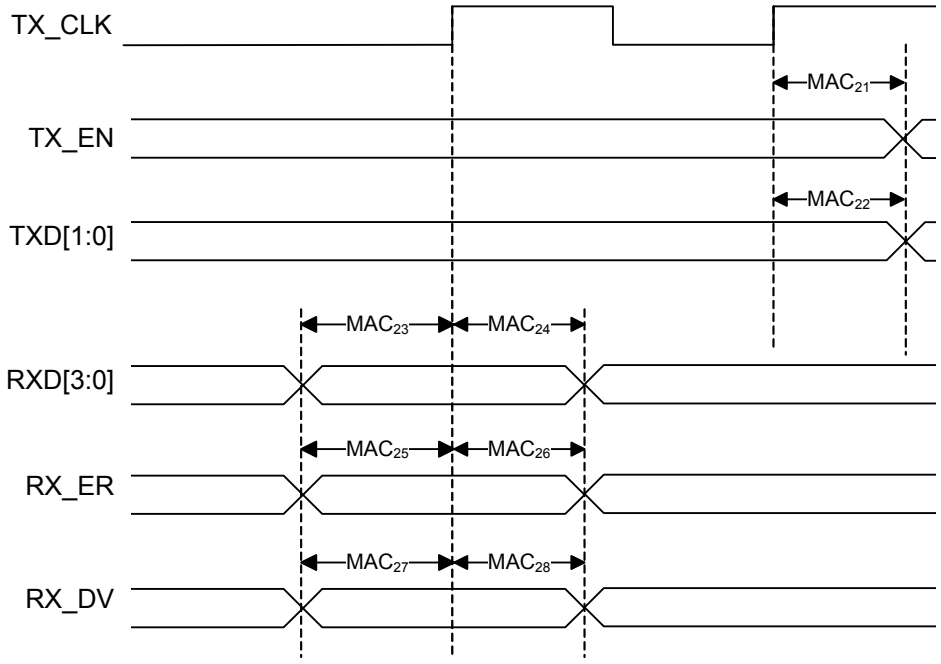


Table 40-59. Ethernet MAC RMII Specific Signals[1]

Symbol	Parameter	Conditions	Min.	Max.	Unit
MAC_{21}	TX_EN toggling from TX_CLK rising	V_{VDD} = 3.0V, drive strength of the pads set to the highest, external capacitor = 10pF on MACB pins	11.7	12.5	ns
MAC_{22}	TXD toggling from TX_CLK rising		11.7	12.5	ns
MAC_{23}	Setup for RXD from TX_CLK		4.5		ns
MAC_{24}	Hold for RXD from TX_CLK		0		ns
MAC_{25}	Setup for RX_ER from TX_CLK		3.4		ns
MAC_{26}	Hold for RX_ER from TX_CLK		0		ns
MAC_{27}	Setup for RX_DV from TX_CLK		4.4		ns
MAC_{28}	Hold for RX_DV from TX_CLK		0		ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Figure 40-21. Ethernet MAC RMII Mode



41. Mechanical Characteristics

41.1 Thermal Considerations

41.1.1 Thermal Data

Table 41-1 summarizes the thermal resistance data depending on the package.

Table 41-1. Thermal Resistance Data

Symbol	Parameter	Condition	Package	Typ	Unit
θ_{JA}	Junction-to-ambient thermal resistance	No air flow	QFN64	20.0	°C/W
θ_{JC}	Junction-to-case thermal resistance		QFN64	0.8	
θ_{JA}	Junction-to-ambient thermal resistance	No air flow	TQFP64	40.5	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP64	8.7	
θ_{JA}	Junction-to-ambient thermal resistance	No air flow	TQFP100	39.3	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP100	8.5	
θ_{JA}	Junction-to-ambient thermal resistance	No air flow	LQFP144	38.1	°C/W
θ_{JC}	Junction-to-case thermal resistance		LQFP144	8.4	

41.1.2 Junction Temperature

The average chip-junction temperature, T_J, in °C can be obtained from the following:

1. $T_J = T_A + (P_D \times \theta_{JA})$
2. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

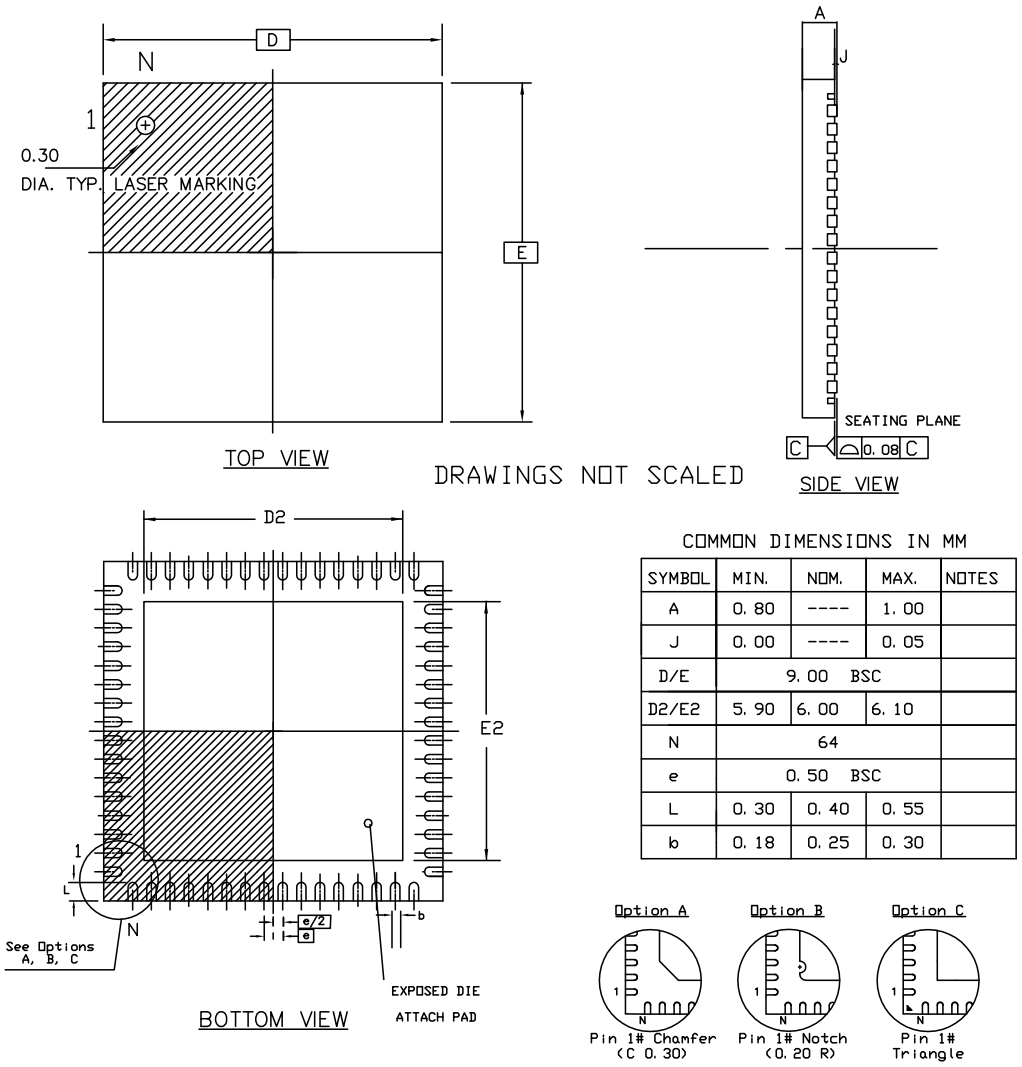
where:

- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 41-1 on page 1289.
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 41-1 on page 1289.
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W) estimated from data provided in the section "Power Consumption" on page 1250.
- T_A = ambient temperature (°C).

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature T_J in °C.

41.2 Package Drawings

Figure 41-1. QFN-64 package drawing



SYMBOL	MIN.	NOM.	MAX.	NOTES
A	0.80	----	1.00	
J	0.00	----	0.05	
D/E		9.00 BSC		
D2/E2	5.90	6.00	6.10	
N		64		
e		0.50 BSC		
L	0.30	0.40	0.55	
b	0.18	0.25	0.30	

Note: The exposed pad is not connected to anything internally, but should be soldered to ground to increase board level reliability.

Table 41-2. Device and Package Maximum Weight

200	mg

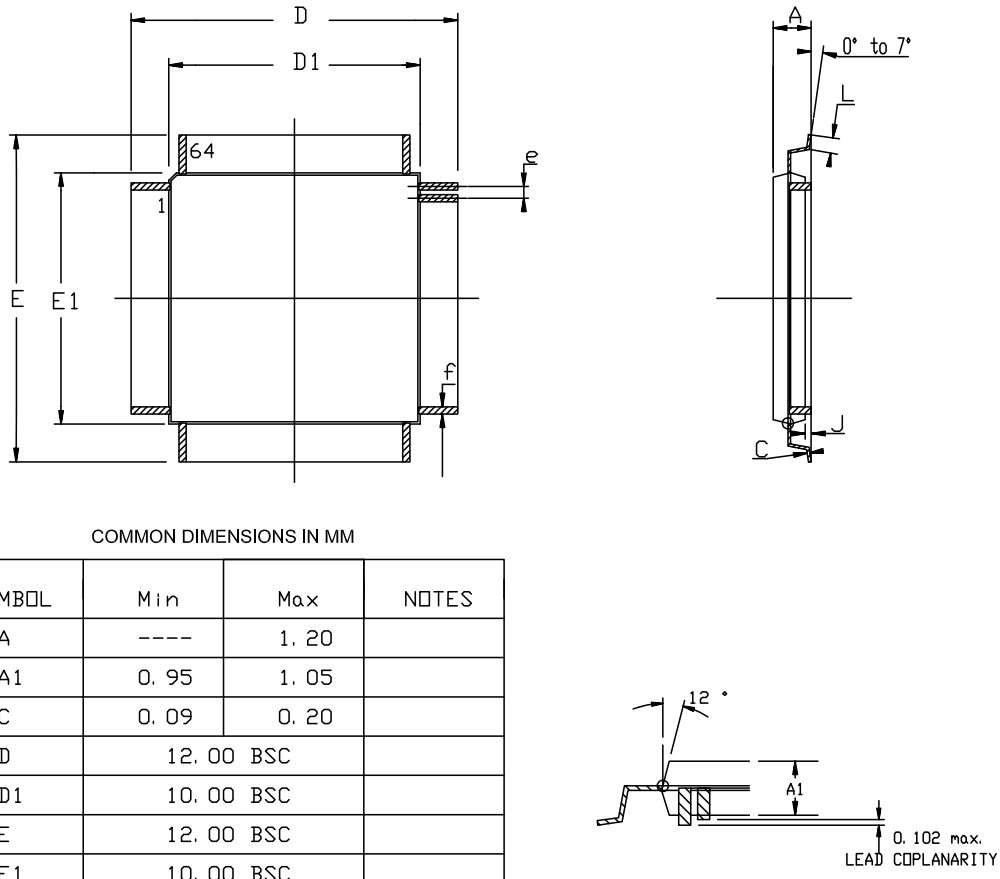
Table 41-3. Package Characteristics

Moisture Sensitivity Level	Jdec J-STD0-20D - MSL 3

Table 41-4. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

Figure 41-2. TQFP-64 package drawing



COMMON DIMENSIONS IN MM

SYMBOL	Min	Max	NOTES
A	----	1.20	
A1	0.95	1.05	
C	0.09	0.20	
D	12.00 BSC		
D1	10.00 BSC		
E	12.00 BSC		
E1	10.00 BSC		
J	0.05	0.15	
L	0.45	0.75	
e	0.50 BSC		
f	0.17	0.27	

Table 41-5. Device and Package Maximum Weight

300	mg

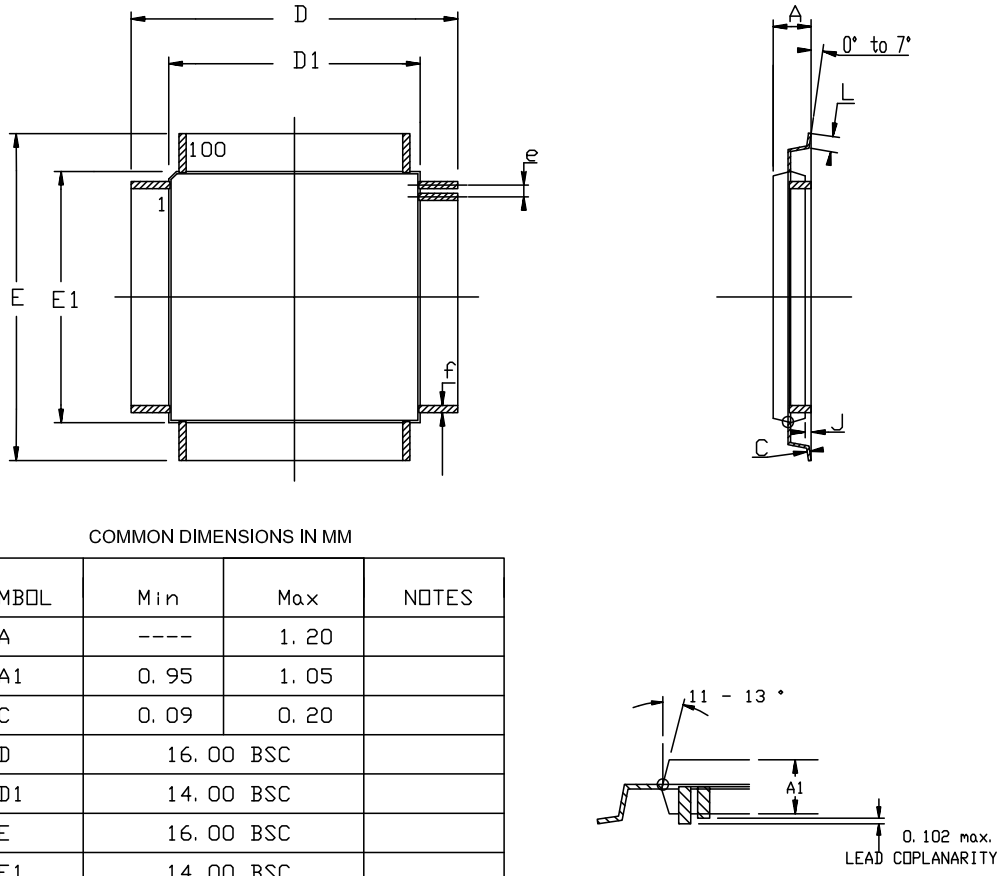
Table 41-6. Package Characteristics

Moisture Sensitivity Level	Jdec J-STD0-20D - MSL 3

Table 41-7. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

Figure 41-3. TQFP-100 package drawing



COMMON DIMENSIONS IN MM

SYMBOL	Min	Max	NOTES
A	----	1.20	
A1	0.95	1.05	
C	0.09	0.20	
D	16.00 BSC		
D1	14.00 BSC		
E	16.00 BSC		
E1	14.00 BSC		
J	0.05	0.15	
L	0.45	0.75	
e	0.50 BSC		
f	0.17	0.27	

Table 41-8. Device and Package Maximum Weight

500	mg

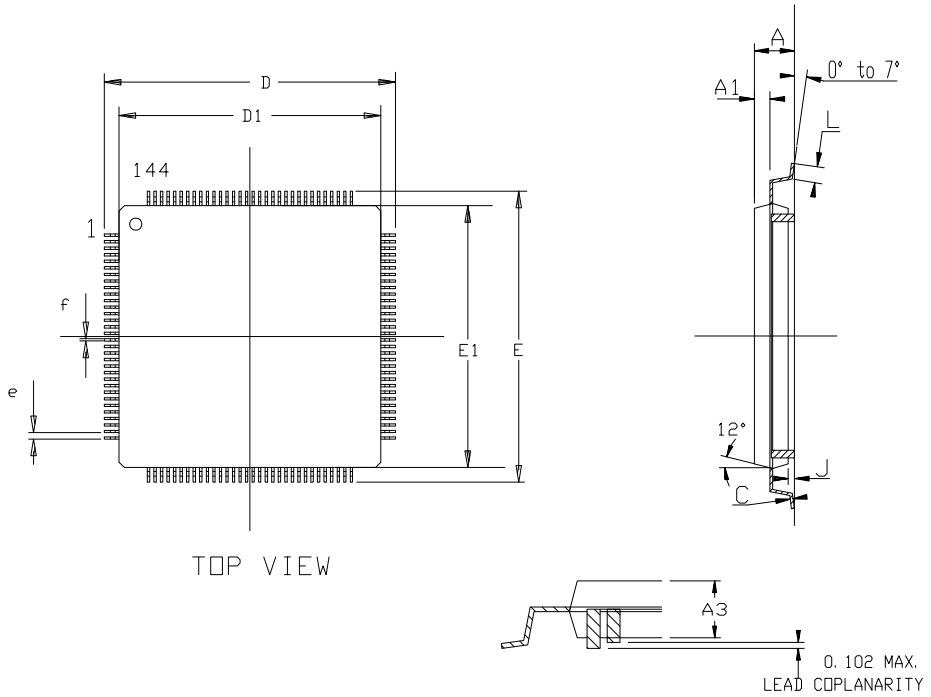
Table 41-9. Package Characteristics

Moisture Sensitivity Level	Jdec J-STD0-20D - MSL 3

Table 41-10. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

Figure 41-4. LQFP-144 package drawing



	MM			INCH		
	Min	Nom	Max	Min	Nom	Max
A	–	–	1.60	–	–	.063
C	0.09	–	0.20	.004	–	.008
A3	1.35	1.40	1.45	.053	.055	.057
D	21.90	22.00	22.10	.862	.866	.870
D1	19.90	20.00	20.10	.783	.787	.791
E	21.90	22.00	22.10	.862	.866	.870
E1	19.90	20.00	20.10	.783	.787	.791
J	0.05	–	0.15	.002	–	.006
L	0.45	0.60	0.75	.018	.024	.030
e	0.50 BSC			.0197 BSC		
f	0.22 BSC			.009 BSC		

Table 41-11. Device and Package Maximum Weight

1300	mg

Table 41-12. Package Characteristics

Moisture Sensitivity Level	Jdec J-STD0-20D - MSL 3

Table 41-13. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

41.3 Soldering Profile

Table 41-14 gives the recommended soldering profile from J-STD-20.

Table 41-14. Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/sec
Preheat Temperature 175°C ±25°C	Min. 150 °C, Max. 200 °C
Temperature Maintained Above 217°C	60-150 sec
Time within 5·C of Actual Peak Temperature	30 sec
Peak Temperature Range	260 °C
Ramp-down Rate	6 °C/sec
Time 25·C to Peak Temperature	Max. 8 minutes

Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

42. Ordering Information

Table 42-1. Ordering Information

Device	Ordering Code	Carrier Type	Package	Temperature Operating Range
AT32UC3C0512C	AT32UC3C0512C-ALUT	Tray	LQFP 144	Industrial (-40°C to 85°C)
	AT32UC3C0512C-ALUR	Tape & Reel		
AT32UC3C0256C	AT32UC3C0256C-ALUT	Tray		
	AT32UC3C0256C-ALUR	Tape & Reel		
AT32UC3C0128C	AT32UC3C0128C-ALUT	Tray		
	AT32UC3C0128C-ALUR	Tape & Reel		
AT32UC3C064C	AT32UC3C064C-ALUT	Tray		
	AT32UC3C064C-ALUR	Tape & Reel		
AT32UC3C1512C	AT32UC3C1512C-AUT	Tray	TQFP 100	
	AT32UC3C1512C-AUR	Tape & Reel		
AT32UC3C1256C	AT32UC3C1256C-AUT	Tray		
	AT32UC3C1256C-AUR	Tape & Reel		
AT32UC3C1128C	AT32UC3C1128C-AUT	Tray		
	AT32UC3C1128C-AUR	Tape & Reel		
AT32UC3C164C	AT32UC3C164C-AUT	Tray		
	AT32UC3C164C-AUR	Tape & Reel		
AT32UC3C2512C	AT32UC3C2512C-A2UT	Tray	TQFP 64	
	AT32UC3C2512C-A2UR	Tape & Reel		
	AT32UC3C2512C-Z2UT	Tray	QFN 64	
	AT32UC3C2512C-Z2UR	Tape & Reel		
AT32UC3C2256C	AT32UC3C2256C-A2UT	Tray	TQFP 64	
	AT32UC3C2256C-A2UR	Tape & Reel		
	AT32UC3C2256C-Z2UT	Tray	QFN 64	
	AT32UC3C2256C-Z2UR	Tape & Reel		
AT32UC3C2128C	AT32UC3C2128C-A2UT	Tray	TQFP 64	
	AT32UC3C2128C-A2UR	Tape & Reel		
	AT32UC3C2128C-Z2UT	Tray	QFN 64	
	AT32UC3C2128C-Z2UR	Tape & Reel		
AT32UC3C264C	AT32UC3C264C-A2UT	Tray	TQFP 64	
	AT32UC3C264C-A2UR	Tape & Reel		
	AT32UC3C264C-Z2UT	Tray	QFN 64	
	AT32UC3C264C-Z2UR	Tape & Reel		

43. Errata

43.1 rev E

43.1.1 AST

1 **AST wake signal is released one ast clock cycle after the busy register is cleared**
After writing to the Status Clear Register (SCR) the wake signal is released one AST clock cycle after the BUSY bit in the Status Register (SR.BUSY) is cleared. If entering sleep mode directly after the BUSY bit is cleared the part will wake up immediately.
Fix/Workaround
Read the Wake Enable Register (WER) and write this value back to the same register. Wait for BUSY to clear before entering sleep mode.

43.1.2 aWire

1 **aWire MEMORY_SPEED_REQUEST command does not return correct CV**
The aWire MEMORY_SPEED_REQUEST command does not return a CV corresponding to the formula in the aWire Debug Interface chapter.
Fix/Workaround
Issue a dummy read to address 0x100000000 before issuing the MEMORY_SPEED_REQUEST command and use this formula instead:

$$f_{sab} = \frac{7 f_{aw}}{CV - 3}$$

43.1.3 Power Manager

1 **PLLCOUNT value larger than zero can cause PLLEN glitch**
Initializing the PLLCOUNT with a value greater than zero creates a glitch on the PLLEN signal during asynchronous wake up.
Fix/Workaround
The lock-masking mechanism for the PLL should not be used.
The PLLCOUNT field of the PLL Control Register should always be written to zero.

43.1.4 SCIF

1 **PLLCOUNT value larger than zero can cause PLLEN glitch**
Initializing the PLLCOUNT with a value greater than zero creates a glitch on the PLLEN signal during asynchronous wake up.
Fix/Workaround
The lock-masking mechanism for the PLL should not be used.
The PLLCOUNT field of the PLL Control Register should always be written to zero.

2 **PLL lock might not clear after disable**
Under certain circumstances, the lock signal from the Phase Locked Loop (PLL) oscillator may not go back to zero after the PLL oscillator has been disabled. This can cause the propagation of clock signals with the wrong frequency to parts of the system that use the PLL clock.
Fix/Workaround
PLL must be turned off before entering STOP, DEEPSTOP or STATIC sleep modes. If PLL has been turned off, a delay of 30us must be observed after the PLL has been enabled

again before the SCIF.PLL0LOCK bit can be used as a valid indication that the PLL is locked.

43.1.5 SPI

1 **SPI data transfer hangs with CSR0.CSAAT==1 and MR.MODFDIS==0**
When CSR0.CSAAT==1 and mode fault detection is enabled (MR.MODFDIS==0), the SPI module will not start a data transfer.
Fix/Workaround
Disable mode fault detection by writing a one to MR.MODFDIS.

2 **Disabling SPI has no effect on flag TDRE flag**
Disabling SPI has no effect on TDRE whereas the write data command is filtered when SPI is disabled. This means that as soon as the SPI is disabled it becomes impossible to reset the TDRE flag by writing in the TDR. So if the SPI is disabled during a PDCA transfer, the PDCA will continue to write data in the TDR (as TDRE stays high) until its buffer is empty, and all data written after the disable command is lost.
Fix/Workaround
Disable the PDCA, insert 2 NOPs, disable SPI. To continue the transfer: Enable SPI, enable PDCA.

3 **SPI disable does not work in SLAVE mode**
SPI disable does not work in SLAVE mode.
Fix/Workaround
Read the last received data, then perform a Software Reset.

4 **SPI Bad Serial Clock Generation on 2nd chip_select when SCBR = 1, CPOL=1 and NCPHA=0**
When multiple CS are in use, if one of the baudrate equals to 1 and one of the others doesn't equal to 1, and CPOL=1 and CPHA=0, then an aditional pulse will be generated on SCK.
Fix/Workaround
When multiple CS are in use, if one of the baudrate equals 1, the other must also equal 1 if CPOL=1 and CPHA=0.

43.1.6 TC

1 **Channel chaining skips first pulse for upper channel**
When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
Fix/Workaround
Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

43.1.7 TWIM

1 **SMBALERT bit may be set after reset**
The SMBus Alert (SMBALERT) bit in the Status Register (SR) might be erroneously set after system reset.
Fix/Workaround
After system reset, clear the SR.SMBALERT bit before commencing any TWI transfer.

43.1.8 TWIS

1 Clearing the NAK bit before the BTF bit is set locks up the TWI bus

When the TWIS is in transmit mode, clearing the NAK Received (NAK) bit of the Status Register (SR) before the end of the Acknowledge/Not Acknowledge cycle will cause the TWIS to attempt to continue transmitting data, thus locking up the bus.

Fix/Workaround

Clear SR.NAK only after the Byte Transfer Finished (BTF) bit of the same register has been set.

43.1.9 USBC

1 UPINRQx.INRQ field is limited to 8-bits

In Host mode, when using the UPINRQx.INRQ feature together with the multi-packet mode to launch a finite number of packet among multi-packet, the multi-packet size (located in the descriptor table) is limited to the UPINRQx.INRQ value multiply by the pipe size.

Fix/Workaround

UPINRQx.INRQ value shall be less than the number of configured multi-packet.

43.2 rev D

43.2.1 AST

1. **AST wake signal is released one ast clock cycle after the busy register is cleared**
   After writing to the Status Clear Register (SCR) the wake signal is released one AST clock cycle after the BUSY bit in the Status Register (SR.BUSY) is cleared. If entering sleep mode directly after the BUSY bit is cleared the part will wake up immediately.
   Fix/Workaround
   Read the Wake Enable Register (WER) and write this value back to the same register. Wait for BUSY to clear before entering sleep mode.

43.2.2 aWire

1. **aWire MEMORY_SPEED_REQUEST command does not return correct CV**
   The aWire MEMORY_SPEED_REQUEST command does not return a CV corresponding to the formula in the aWire Debug Interface chapter.
   Fix/Workaround
   Issue a dummy read to address 0x100000000 before issuing the MEMORY_SPEED_REQUEST command and use this formula instead:

$$f_{sab} = \frac{7 f_{aw}}{CV - 3}$$

43.2.3 GPIO

1. **Clearing Interrupt flags can mask other interrupts**
   When clearing interrupt flags in a GPIO port, interrupts on other pins of that port, happening in the same clock cycle will not be registered.
   Fix/Workaround
   Read the PVR register of the port before and after clearing the interrupt to see if any pin change has happened while clearing the interrupt. If any change occurred in the PVR between the reads, they must be treated as an interrupt.

43.2.4 Power Manager

1. **Clock Failure Detector (CFD) can be issued while turning off the CFD**
   While turning off the CFD, the CFD bit in the Status Register (SR) can be set. This will change the main clock source to RCSYS.
   Fix/Workaround
   Solution 1: Enable CFD interrupt. If CFD interrupt is issues after turning off the CFD, switch back to original main clock source.
   Solution 2: Only turn off the CFD while running the main clock on RCSYS.

2. **Requesting clocks in idle sleep modes will mask all other PB clocks than the requested**
   In idle or frozen sleep mode, all the PB clocks will be frozen if the TWIS or the AST need to wake the cpu up.
   Fix/Workaround
   Disable the TWIS or the AST before entering idle or frozen sleep mode.

3 **PLLCOUNT value larger than zero can cause PLLEN glitch**
Initializing the PLLCOUNT with a value greater than zero creates a glitch on the PLLEN signal during asynchronous wake up.
Fix/Workaround
The lock-masking mechanism for the PLL should not be used.
The PLLCOUNT field of the PLL Control Register should always be written to zero.

43.2.5 SCIF

1 **PLLCOUNT value larger than zero can cause PLLEN glitch**
Initializing the PLLCOUNT with a value greater than zero creates a glitch on the PLLEN signal during asynchronous wake up.
Fix/Workaround
The lock-masking mechanism for the PLL should not be used.
The PLLCOUNT field of the PLL Control Register should always be written to zero.

2 **PLL lock might not clear after disable**
Under certain circumstances, the lock signal from the Phase Locked Loop (PLL) oscillator may not go back to zero after the PLL oscillator has been disabled. This can cause the propagation of clock signals with the wrong frequency to parts of the system that use the PLL clock.
Fix/Workaround
PLL must be turned off before entering STOP, DEEPSTOP or STATIC sleep modes. If PLL has been turned off, a delay of 30us must be observed after the PLL has been enabled again before the SCIF.PLL0LOCK bit can be used as a valid indication that the PLL is locked.

43.2.6 SPI

1 **SPI data transfer hangs with CSR0.CSAAT==1 and MR.MODFDIS==0**
When CSR0.CSAAT==1 and mode fault detection is enabled (MR.MODFDIS==0), the SPI module will not start a data transfer.
Fix/Workaround
Disable mode fault detection by writing a one to MR.MODFDIS.

2 **Disabling SPI has no effect on flag TDRE flag**
Disabling SPI has no effect on TDRE whereas the write data command is filtered when SPI is disabled. This means that as soon as the SPI is disabled it becomes impossible to reset the TDRE flag by writing in the TDR. So if the SPI is disabled during a PDCA transfer, the PDCA will continue to write data in the TDR (as TDRE stays high) until its buffer is empty, and all data written after the disable command is lost.
Fix/Workaround
Disable the PDCA, insert 2 NOPs, disable SPI. To continue the transfer: Enable SPI, enable PDCA.

3 **SPI disable does not work in SLAVE mode**
SPI disable does not work in SLAVE mode.
Fix/Workaround
Read the last received data, then perform a Software Reset.

4 **SPI Bad Serial Clock Generation on 2nd chip_select when SCBR = 1, CPOL=1 and NCPHA=0**
When multiple CS are in use, if one of the baudrate equals to 1 and one of the others doesn't equal to 1, and CPOL=1 and CPHA=0, then an aditional pulse will be generated on SCK.

Fix/Workaround
When multiple CS are in use, if one of the baudrate equals 1, the other must also equal 1 if CPOL=1 and CPHA=0.

43.2.7 TC

1 Channel chaining skips first pulse for upper channel
When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
Fix/Workaround
Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

43.2.8 TWIM

1 SMBALERT bit may be set after reset
The SMBus Alert (SMBALERT) bit in the Status Register (SR) might be erroneously set after system reset.
Fix/Workaround
After system reset, clear the SR.SMBALERT bit before commencing any TWI transfer.

2 TWIM TWALM polarity is wrong
The TWALM signal in the TWIM is active high instead of active low.
Fix/Workaround
use an external inverter to invert the signal going into the TWIM. When using both TWIM and TWIS on the same pins, the TWALM cannot be used.

43.2.9 TWIS

1 Clearing the NAK bit before the BTF bit is set locks up the TWI bus
When the TWIS is in transmit mode, clearing the NAK Received (NAK) bit of the Status Register (SR) before the end of the Acknowledge/Not Acknowledge cycle will cause the TWIS to attempt to continue transmitting data, thus locking up the bus.
Fix/Workaround
Clear SR.NAK only after the Byte Transfer Finished (BTF) bit of the same register has been set.

2 TWIS stretch on Address match error
When the TWIS stretches TWCK due to a slave address match, it also holds TWD low for the same duration if it is to be receiving data. When TWIS releases TWCK, it releases TWD at the same time. This can cause a TWI timing violation.
Fix/Workaround
None.

3 TWALM forced to GND
The TWALM pin is forced to GND when the alternate function is selected and the TWIS module is enabled.
Fix/Workaround
None.

43.2.10 USBC

1 UPINRQx.INRQ field is limited to 8-bits

In Host mode, when using the UPINRQx.INRQ feature together with the multi-packet mode to launch a finite number of packet among multi-packet, the multi-packet size (located in the descriptor table) is limited to the UPINRQx.INRQ value multiply by the pipe size.

Fix/Workaround

UPINRQx.INRQ value shall be less than the number of configured multi-packet.

43.2.11 WDT

1 Clearing the Watchdog Timer (WDT) counter in second half of timeout period will issue a Watchdog reset

If the WDT counter is cleared in the second half of the timeout period, the WDT will immediately issue a Watchdog reset.

Fix/Workaround

Use twice as long timeout period as needed and clear the WDT counter within the first half of the timeout period. If the WDT counter is cleared after the first half of the timeout period, you will get a Watchdog reset immediately. If the WDT counter is not cleared at all, the time before the reset will be twice as long as needed.

44. Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

44.1 Rev. B – 03/11

1	Package and pinout: Added supply column. Updated peripheral functions
2	Supply and Startup Considerations: Updated I/O lines power
3	PM: Added AWEN description
4	SCIF: Added VREGCR register
5	AST: Updated digital tuner formula
6	SDRAMC: cleaned-up SDCS/NCS names. Added VERSION register
7	SAU: Updated SR.IDLE
8	USART: Updated
9	CANIF: Updated address map figure
10	USBC: Updated
11	DACIFB: Updated
12	Programming and Debugging: Added JTAG Data Registers section
13	Electrical Characteristics: Updated
14	Ordering Information: Updated
15	Errata: Updated

44.2 Rev. A – 10/10

1	Initial revision

Table of Content

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

97505