Features

- High Performance, Low Power 32-bit AVR® Microcontroller
  - Compact Single-Cycle RISC Instruction Set Including DSP Instructions
  - Read-Modify-Write Instructions and Atomic Bit Manipulation
  - Performance
    - Up to 64 DMIPS Running at 50MHz from Flash (1 Flash Wait State)
    - Up to 36 DMIPS Running at 25MHz from Flash (0 Flash Wait State)
  - Memory Protection Unit (MPU)
    - Secure Access Unit (SAU) providing user defined peripheral protection
- picoPower® Technology for Ultra-Low Power Consumption
- Multi-Hierarchy Bus System
  - High-Performance Data Transfers on Separate Buses for Increased Performance
  - 12 Peripheral DMA Channels Improve Speed for Peripheral Communication
- Internal High-Speed Flash
  - 64Kbytes, 32Kbytes, and 16Kbytes Versions
  - Single-Cycle Access up to 25MHz
  - 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
  - 16Kbytes (64Kbytes and 32Kbytes Flash), or 8Kbytes (16Kbytes Flash)
- Interrupt Controller (INTC)
  - Autovectored Low Latency Interrupt Service with Programmable Priority
- External Interrupt Controller (EIC)
- Peripheral Event System for Direct Peripheral to Peripheral Communication
- System Functions
  - Power and Clock Manager
  - SleepWalking™ Power Saving Control
  - Internal System RC Oscillator (RCSYS)
  - 32 KHz Oscillator
  - Multipurpose Oscillator and Digital Frequency Locked Loop (DFLL)
- 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
- Six 16-bit Timer/Counter (TC) Channels
  - External Clock Inputs, PWM, Capture and Various Counting Capabilities
- PWM Channels on All I/O Pins (PWMA)
  - 8-bit PWM up to 150MHz Source Clock
- Four Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
  - Independent Baudrate Generator, Support for SPI
  - Support for Hardware Handshaking
- One Master/Slave Serial Peripheral Interfaces (SPI) with Chip Select Signals
  - Up to 15 SPI Slaves can be Addressed
- Two Master and Two Slave Two-Wire Interfaces (TWI), 400kbit/s I^2C-compatible
- One 8-channel Analog-To-Digital Converter (ADC) with up to 12 Bits Resolution
  - Internal Temperature Sensor

32-bit AVR® Microcontroller

AT32UC3L064
AT32UC3L032
AT32UC3L016

Preliminary

32099D–06/2010

- Eight Analog Comparators (AC) with Optional Window Detection
- Capacitive Touch (CAT) Module
  - Hardware Assisted QTouch® and QMatrix® Touch Acquisition
  - Supports QTouch® and QMatrix® Capture from Capacitive Touch Sensors
- 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
- 48-pin TQFP/QFN/TLLGA (36 GPIO Pins)
- Five High-Drive I/O Pins
- Single 1.62-3.6V Power Supply

1. Description

The AT32UC3L is a complete System-On-Chip microcontroller based on the AVR32 UC RISC processor running at frequencies up to 50MHz. AVR32 UC 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. The Secure Access Unit (SAU) is used together with the MPU to provide the required security and integrity.

Higher computation capability is achieved using a rich set of DSP instructions.

The AT32UC3L embeds state-of-the-art picoPower technology for ultra-low power consumption. Combined power control techniques are used to bring active current consumption down to 165µA/MHz, and leakage down to 9nA while still retaining a bank of backup registers. The device allows a wide range of trade-offs between functionality and power consumption, giving the user the ability to reach the lowest possible power consumption with the feature set required for the application.

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

The AT32UC3L 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 costumers, who will be able to program their own code into the device, accessing the secure libraries, but without risk of compromising the proprietary secure code.

The Peripheral Event System allows peripherals to receive, react to, and send peripheral events without CPU intervention. Asynchronous interrupts allow advanced peripheral operation in low power sleep modes.

The Power Manager improves design flexibility and security. The Power Manager supports SleepWalking functionality, by which a module can be selectively activated based on peripheral events, even in sleep modes where the module clock is stopped. Power monitoring is supported by on-chip Power-On Reset (POR), Brown-Out Detector (BOD), and Supply Monitor (SM). The device features several oscillators, such as Digital Frequency Locked Loop (DFLL), Oscillator 0 (OSC0), and system RC oscillator (RCSYS). Either of these oscillators can be used as source for the system clock. The DFLL is a programmable internal oscillator from 40 to 150MHz. It can be tuned to a high accuracy if an accurate oscillator is running, e.g. the 32KHz crystal oscillator.

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 Asynchronous Timer (AST) combined with the 32KHz crystal oscillator supports powerful real-time clock capabilities, with a maximum timeout of up to 136 years. The AST can operate in counter mode or calendar mode.

The Frequency Meter (FREQM) allows accurate measuring of a clock frequency by comparing it to a known reference clock.

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 Pulse Width Modulation controller (PWMA) provides 8-bit PWM channels which can be synchronized and controlled from a common timer. One PWM channel is available for each I/O pin on the device, enabling applications that require multiple PWM outputs, such as LCD backlight control. The PWM channels can operate independently, with duty cycles set independently from each other, or in interlinked mode, with multiple channels changed at the same time.

The AT32UC3L also features many communication interfaces for communication intensive applications like USART, SPI, or TWI.

A general purpose 8-channel ADC is provided, as well as eight analog comparators (AC). The ADC can operate in 10-bit mode at full speed or in enhanced mode at reduced speed, offering up to 12-bit resolution. The ADC also provides an internal temperature sensor input channel. The analog comparators can be paired to detect when the sensing voltage is within or outside the defined reference window.

The Capacitive Touch (CAT) module senses touch on external capacitive touch sensors, using the QTouch technology. Capacitive touch sensors use no external mechanical components, unlike normal push buttons, and therefore demand less maintenance in the user application. The CAT module allows up to 17 touch sensors, or up to 16 by 8 matrix sensors to be interfaced. One touch sensor can be configured to operate autonomously without software interaction, allowing wakeup from sleep modes when activated.

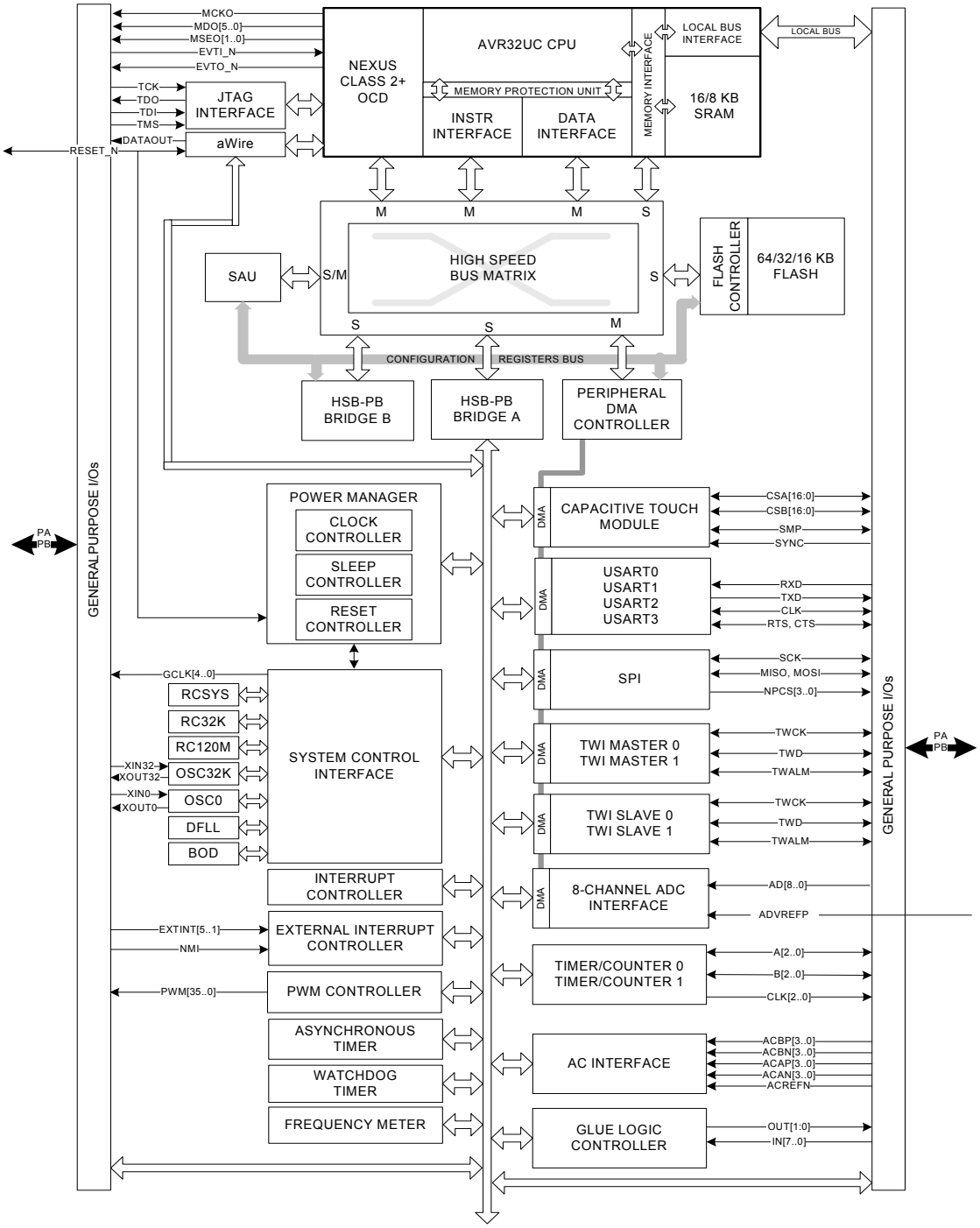
Atmel offers the QTouch library for embedding capacitive touch buttons, sliders, and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and included fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS®) 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.

The AT32UC3L 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	AT32UC3L064	AT32UC3L032	AT32UC3L016
Flash	64KB	32KB	16KB
SRAM	16KB	16KB	8KB
GPIO	36		
High-drive pins	5		
External Interrupts	6		
TWI	2		
USART	4		
Peripheral DMA Channels	12		
Peripheral Event System	1		
SPI	1		
Asynchronous Timers	1		
Timer/Counter Channels	6		
PWM channels	36		
Frequency Meter	1		
Watchdog Timer	1		
Power Manager	1		
Secure Access Unit	1		
Glue Logic Controller	1		
Oscillators	Digital Frequency Locked Loop 40-150 MHz (DFLL) Crystal Oscillator 3-16 MHz (OSC0) Crystal Oscillator 32 KHz (OSC32K) RC Oscillator 120MHz (RC120M) RC Oscillator 115 kHz (RCSYS) RC Oscillator 32 kHz (RC32K)		
ADC	8-channel 12-bit		
Temperature Sensor	1		
Analog Comparators	8		
Capacitive Touch Module	1		
JTAG	1		
aWire	1		
Max Frequency	50 MHz		
Package	TQFP48/QFN48/TLLGA48		

3. Package and Pinout

3.1 Package

The device pins are multiplexed with peripheral functions as described in Section 3.2.

Figure 3-1. TQFP48/QFN48 Pinout

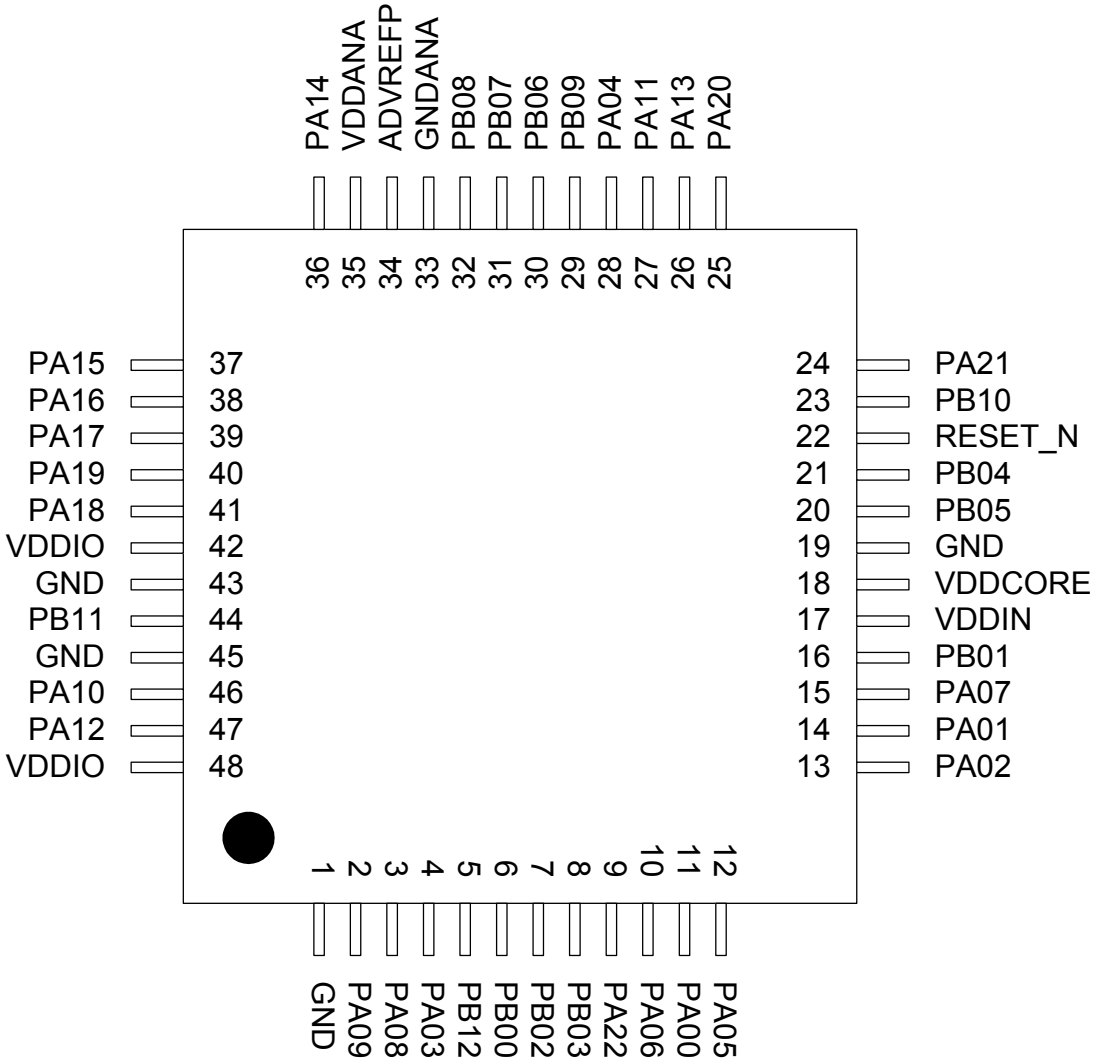
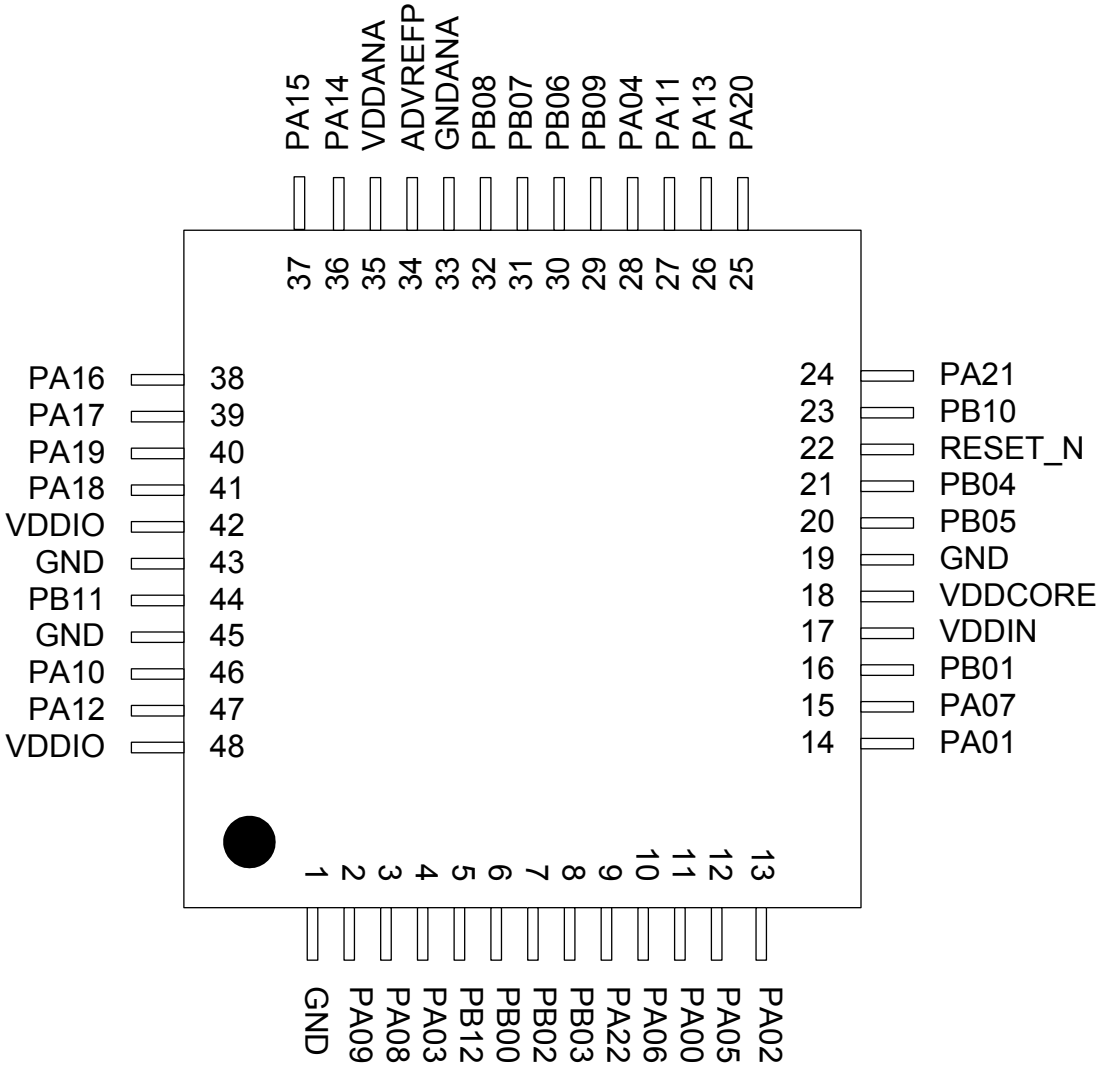


Figure 3-2. TLLGA48 Pinout



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

QFP 48	PIN	GPIO	Supply	Pad Type	GPIO Function							
					A	B	C	D	E	F	G	H
11	PA00	0	VDDIO	Normal I/O	USART0-TXD	USART1-RTS	SPI-NPCS[2]		PWMA-PWMA[0]		SCIF-GCLK[0]	CAT-CSA[2]
14	PA01	1	VDDIO	Normal I/O	USART0-RXD	USART1-CTS	SPI-NPCS[3]	USART1-CLK	PWMA-PWMA[1]	ACIFB-ACAP[0]	TWIMS0-TWALM	CAT-CSA[1]
13	PA02	2	VDDIO	High-drive I/O	USART0-RTS	ADCIFB-TRIGGER	USART2-TXD	TC0-A0	PWMA-PWMA[2]	ACIFB-ACBP[0]	USART0-CLK	CAT-CSA[3]
4	PA03	3	VDDIO	Normal I/O	USART0-CTS	SPI-NPCS[1]	USART2-TXD	TC0-B0	PWMA-PWMA[3]	ACIFB-ACBN[3]	USART0-CLK	CAT-CSB[3]
28	PA04	4	VDDIO	Normal I/O	SPI-MISO	TWIMS0-TWCK	USART1-RXD	TC0-B1	PWMA-PWMA[4]	ACIFB-ACBP[1]		CAT-CSA[7]
12	PA05	5	VDDIO	TWI, Normal I/O	SPI-MOSI	TWIMS1-TWCK	USART1-TXD	TC0-A1	PWMA-PWMA[5]	ACIFB-ACBN[0]	TWIMS0-TWD	CAT-CSB[7]
10	PA06	6	VDDIO	High-drive I/O, 5V tolerant	SPI-SCK	USART2-TXD	USART1-CLK	TC0-B0	PWMA-PWMA[6]		SCIF-GCLK[1]	CAT-CSB[1]
15	PA07	7	VDDIO	TWI, Normal I/O	SPI-NPCS[0]	USART2-RXD	TWIMS1-TWALM	TWIMS0-TWCK	PWMA-PWMA[7]	ACIFB-ACAN[0]	NMI	CAT-CSB[2]
3	PA08	8	VDDIO	High-drive I/O	USART1-TXD	SPI-NPCS[2]	TC0-A2	ADCIFB-ADP[0]	PWMA-PWMA[8]			CAT-CSA[4]
2	PA09	9	VDDIO	High-drive I/O	USART1-RXD	SPI-NPCS[3]	TC0-B2	ADCIFB-ADP[1]	PWMA-PWMA[9]	SCIF-GCLK[2]	EIC-EXTINT[1]	CAT-CSB[4]
46	PA10	10	VDDIO	Normal I/O	TWIMS0-TWD		TC0-A0		PWMA-PWMA[10]	ACIFB-ACAP[1]	SCIF-GCLK[2]	CAT-CSA[5]
27	PA11	11	VDDIN	Normal I/O					PWMA-PWMA[11]			
47	PA12	12	VDDIO	Normal I/O	ADCIFB-PRND	USART2-CLK	TC0-CLK1	CAT-SMP	PWMA-PWMA[12]	ACIFB-ACAN[1]	SCIF-GCLK[3]	CAT-CSB[5]
26	PA13	13	VDDIN	Normal I/O	GLOC-OUT[0]	GLOC-IN[7]	TC0-A0	SCIF-GCLK[2]	PWMA-PWMA[13]	CAT-SMP	EIC-EXTINT[2]	CAT-CSA[0]
36	PA14	14	VDDIO	Normal I/O	ADCIFB-AD[0]	TC0-CLK2	USART2-RTS	CAT-SMP	PWMA-PWMA[14]		SCIF-GCLK[4]	CAT-CSA[6]
37	PA15	15	VDDIO	Normal I/O	ADCIFB-AD[1]	TC0-CLK1		GLOC-IN[6]	PWMA-PWMA[15]	CAT-SYNC	EIC-EXTINT[3]	CAT-CSB[6]
38	PA16	16	VDDIO	Normal I/O	ADCIFB-AD[2]	TC0-CLK0		GLOC-IN[5]	PWMA-PWMA[16]	ACIFB-ACREFN	EIC-EXTINT[4]	CAT-CSA[8]
39	PA17	17	VDDIO	TWI, Normal I/O		TC0-A1	USART2-CTS	TWIMS1-TWD	PWMA-PWMA[17]	CAT-SMP	CAT-DIS	CAT-CSB[8]

Table 3-1. GPIO Controller Function Multiplexing

41	PA18	18	VDDIO	Normal I/O	ADCIFB-AD[4]	TC0-B1		GLOC-IN[4]	PWMA-PWMA[18]	CAT-SYNC	EIC-EXTINT[5]	CAT-CSB[0]
40	PA19	19	VDDIO	Normal I/O	ADCIFB-AD[5]		TC0-A2	TWIMS1-TWALM	PWMA-PWMA[19]		CAT-SYNC	CAT-CSA[10]
25	PA20	20	VDDIN	Normal I/O	USART2-TXD		TC0-A1	GLOC-IN[3]	PWMA-PWMA[20]	SCIF-RC32OUT		CAT-CSA[12]
24	PA21	21	VDDIN	TWI, 5V tolerant, SMBus, Normal I/O	USART2-RXD	TWIMS0-TWD	TC0-B1	ADCIFB-TRIGGER	PWMA-PWMA[21]	PWMA-PWMAOD[21]	SCIF-GCLK[0]	CAT-SMP
9	PA22	22	VDDIO	Normal I/O	USART0-CTS	USART2-CLK	TC0-B2	CAT-SMP	PWMA-PWMA[22]	ACIFB-ACBN[2]		CAT-CSB[10]
6	PB00	32	VDDIO	Normal I/O	USART3-TXD	ADCIFB-ADP[0]	SPI-NPCS[0]	TC0-A1	PWMA-PWMA[23]	ACIFB-ACAP[2]	TC1-A0	CAT-CSA[9]
16	PB01	33	VDDIO	High-drive I/O	USART3-RXD	ADCIFB-ADP[1]	SPI-SCK	TC0-B1	PWMA-PWMA[24]		TC1-A1	CAT-CSB[9]
7	PB02	34	VDDIO	Normal I/O	USART3-RTS	USART3-CLK	SPI-MISO	TC0-A2	PWMA-PWMA[25]	ACIFB-ACAN[2]	SCIF-GCLK[1]	CAT-CSB[11]
8	PB03	35	VDDIO	Normal I/O	USART3-CTS	USART3-CLK	SPI-MOSI	TC0-B2	PWMA-PWMA[26]	ACIFB-ACBP[2]	TC1-A2	CAT-CSA[11]
21	PB04	36	VDDIN	TWI, 5V tolerant, SMBus, Normal I/O	TC1-A0	USART1-RTS	USART1-CLK	TWIMS0-TWALM	PWMA-PWMA[27]	PWMA-PWMAOD[27]	TWIMS1-TWCK	CAT-CSA[14]
20	PB05	37	VDDIN	TWI, 5V tolerant, SMBus, Normal I/O	TC1-B0	USART1-CTS	USART1-CLK	TWIMS0-TWCK	PWMA-PWMA[28]	PWMA-PWMAOD[28]	SCIF-GCLK[3]	CAT-CSB[14]
30	PB06	38	VDDIO	Normal I/O	TC1-A1	USART3-TXD	ADCIFB-AD[6]	GLOC-IN[2]	PWMA-PWMA[29]	ACIFB-ACAN[3]	NMI	CAT-CSB[13]
31	PB07	39	VDDIO	Normal I/O	TC1-B1	USART3-RXD	ADCIFB-AD[7]	GLOC-IN[1]	PWMA-PWMA[30]	ACIFB-ACAP[3]	EIC-EXTINT[1]	CAT-CSA[13]
32	PB08	40	VDDIO	Normal I/O	TC1-A2	USART3-RTS	ADCIFB-AD[8]	GLOC-IN[0]	PWMA-PWMA[31]	CAT-SYNC	EIC-EXTINT[2]	CAT-CSB[12]
29	PB09	41	VDDIO	Normal I/O	TC1-B2	USART3-CTS	USART3-CLK		PWMA-PWMA[32]	ACIFB-ACBN[1]	EIC-EXTINT[3]	CAT-CSB[15]
23	PB10	42	VDDIN	Normal I/O	TC1-CLK0	USART1-TXD	USART3-CLK	GLOC-OUT[1]	PWMA-PWMA[33]		EIC-EXTINT[4]	CAT-CSB[16]
44	PB11	43	VDDIO	Normal I/O	TC1-CLK1	USART1-RXD		ADCIFB-TRIGGER	PWMA-PWMA[34]	CAT-VDIVEN	EIC-EXTINT[5]	CAT-CSA[16]
5	PB12	44	VDDIO	Normal I/O	TC1-CLK2		TWIMS1-TWALM	CAT-SYNC	PWMA-PWMA[35]	ACIFB-ACBP[3]	SCIF-GCLK[4]	CAT-CSA[15]

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

Signals are prioritized according to the function priority listed in Table 3-2 on page 11 if multiple functions are enabled simultaneously.

Refer to "Electrical Characteristics" on page 776 for a description of the electrical properties of the pad types used.

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.

Table 3-2. Peripheral Functions

Function	Description
A	GPIO peripheral selection A
B	GPIO peripheral selection B
C	GPIO peripheral selection C
D	GPIO peripheral selection D
E	GPIO peripheral selection E
F	GPIO peripheral selection F
G	GPIO peripheral selection G
H	GPIO peripheral selection H

3.2.3 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 3-3. JTAG Pinout

48TQFP/QFN/TLLGA	Pin	JTAG Function
11	PA00	TCK
14	PA01	TMS
13	PA02	TDO
4	PA03	TDI

3.2.4 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 I/O Controller configuration. Two different OCD trace pin mappings are possible, depending on the configuration of the OCD AXS register. For details, see the AVR32 UC Technical Reference Manual.

Table 3-4. Nexus OCD AUX Port Connections

Pin	AXS=1	AXS=0
EVTI_N	PA05	PB08
MDO[5]	PA10	PB00
MDO[4]	PA18	PB04
MDO[3]	PA17	PB05

Table 3-4. Nexus OCD AUX Port Connections

Pin	AXS=1	AXS=0
MDO[2]	PA16	PB03
MDO[1]	PA15	PB02
MDO[0]	PA14	PB09
EVTO_N	PA04	PA04
MCKO	PA06	PB01
MSEO[1]	PA07	PB11
MSEO[0]	PA11	PB12

3.2.5 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 3-5. Oscillator Pinout

48TQFP/QFN/TLLGA	Pin	Oscillator Function
3	PA08	XIN0
46	PA10	XIN32
26	PA13	XIN32_2
2	PA09	XOUT0
47	PA12	XOUT32
25	PA20	XOUT32_2

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 actice after the aWire is enabled and the 2_PIN_MODE command has been sent. The WAKE_N pin is always enabled. Please refer to Section 6.1.4 on page 40 for constraints on the WAKE_N pin.

Table 3-6. Other Functions

48TQFP/TQFN/TLLGA	Pin	Function
27	PA11	WAKE_N
22	RESET_N	aWire DATA
11	PA00	aWire DATAOUT

3.3 Signal Descriptions

The following table gives details on signal name classified by peripheral.

Table 3-7. Signal Descriptions List

Signal Name	Function	Type	Active Level	Comments
Analog Comparator Interface - ACIFB				
ACAN3 - ACAN0	Negative inputs for comparators "A"	Analog		
ACAP3 - ACAP0	Positive inputs for comparators "A"	Analog		
ACBN3 - ACBN0	Negative inputs for comparators "B"	Analog		
ACBP3 - ACBP0	Positive inputs for comparators "B"	Analog		
ACREFN	Common negative reference	Analog		
ADC Interface - ADCIFB				
AD8 - AD0	Analog Signal	Analog		
ADP1 - ADP0	Drive Pin for resistive touch screen	Output		
PRND	Pseudorandom output signal	Output		
TRIGGER	External trigger	Input		
aWire - AW				
DATA	aWire data	I/O		
DATAOUT	aWire data output for 2-pin mode	I/O		
Capacitive Touch Module - CAT				
CSA16 - CSA0	Capacitive Sense A	I/O		
CSB16 - CSB0	Capacitive Sense B	I/O		
SMP	SMP signal	Output		
SYNC	Synchronize signal	Input		
VDIVEN	Voltage divider enable	Output		
External Interrupt Controller - EIC				
NMI	Non-Maskable Interrupt	Input		
EXTINT5 - EXTINT1	External interrupt	Input		
Glue Logic Controller - GLOC				
IN7 - IN0	Inputs to lookup tables	Input		
OUT1 - OUT0	Outputs from lookup tables	Output		
JTAG module - JTAG				
TCK	Test Clock	Input		
TDI	Test Data In	Input		
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		

Table 3-7. Signal Descriptions List

Power Manager - PM				
RESET_N	Reset	Input	Low	
Pulse Width Modulation Controller - PWMA				
PWMA35 - PWMA0	PWMA channel waveforms	Output		
PWMAOD35 - PWMAOD0	PWMA channel waveforms, open drain mode	Output		Not all channels support open drain mode
System Control Interface - SCIF				
GCLK4 - GCLK0	Generic Clock Output	Output		
RC32OUT	RC32K output at startup	Output		
XIN0	Crystal 0 Input	Analog/ Digital		
XIN32	Crystal 32 Input (primary location)	Analog/ Digital		
XIN32_2	Crystal 32 Input (secondary location)	Analog/ Digital		
XOUT0	Crystal 0 Output	Analog		
XOUT32	Crystal 32 Output (primary location)	Analog		
XOUT32_2	Crystal 32 Output (secondary location)	Analog		
Serial Peripheral Interface - SPI				
MISO	Master In Slave Out	I/O		
MOSI	Master Out Slave In	I/O		
NPCS3 - NPCS0	SPI Peripheral Chip Select	I/O	Low	
SCK	Clock	I/O		
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				
TWALM	SMBus SMBALERT	I/O	Low	
TWCK	Two-wire Serial Clock	I/O		
TWD	Two-wire Serial Data	I/O		
Universal Synchronous/Asynchronous Receiver/Transmitter - USART0, USART1, USART2, USART3				

Table 3-7. Signal Descriptions List

CLK	Clock	I/O		
CTS	Clear To Send	Input	Low	
RTS	Request To Send	Output	Low	
RXD	Receive Data	Input		
TXD	Transmit Data	Output		

Note: 1. ADCIFB: AD3 does not exist.

Table 3-8. Signal Description List, continued

Signal Name	Function	Type	Active Level	Comments
Power				
VDDCORE	Core Power Supply / Voltage Regulator Output	Power Input/Output		1.62V to 1.98V
VDDIO	I/O Power Supply	Power Input		1.62V to 3.6V. VDDIO should always be equal to or lower than VDDIN.
VDDANA	Analog Power Supply	Power Input		1.62V to 1.98V
ADVREFP	Analog Reference Voltage	Power Input		TBD to 1.98V
VDDIN	Voltage Regulator Input	Power Input		1.62V to 3.6V (1)
GNDANA	Analog Ground	Ground		
GND	Ground	Ground		
Auxiliary Port - AUX				
MCKO	Trace Data Output Clock	Output		
MDO5 - MDO0	Trace Data Output	Output		
MSEO1 - MSEO0	Trace Frame Control	Output		
EVTI_N	Event In	Input	Low	
EVTO_N	Event Out	Output	Low	
General Purpose I/O pin				
PA22 - PA00	Parallel I/O Controller I/O Port 0	I/O		
PB12 - PB00	Parallel I/O Controller I/O Port 1	I/O		

1. See Section 6.1 on page 36

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 VDDIO, and has no pull-up resistor. The JTAG pins can be used as GPIO pins and multiplexed with peripherals when the JTAG is disabled. Please refer to Section 3.2.3 on page 11 for the JTAG port connections.

3.4.2 PA00

Note that PA00 is multiplexed with TCK. PA00 GPIO function must only be used as output in the application.

3.4.3 RESET_N Pin

The RESET_N pin is a schmitt input and integrates a permanent pull-up resistor to VDDIN. As the product integrates a power-on reset detector, 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.

3.4.4 TWI0 Pins

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with spike filtering. When used as GPIO pins or used for other peripherals, the pins have the characteristics indicated in the Electrical Characteristics section. Selected pins are also SMBus compliant (refer to Section 3.2 on page 9). As required by the SMBus specification, these pins provide no leakage path to ground when the AT32UC3L is powered down. This allows other devices on the SMBus to continue communicating even though the AT32UC3L is not powered. This feature is only available when pins PA21/PB04/PB05 are used for TWI0.

3.4.5 TWI1 Pins

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

3.4.6 GPIO Pins

All the I/O lines integrate a pull-up resistor. Programming of this pull-up resistor is performed independently for each I/O line through the GPIO Controllers. After reset, I/O lines default as inputs with pull-up resistors disabled, except PA00. PA20 selects SCIF-RC32OUT (GPIO Function F) as default enabled after reset.

3.4.7 High-Drive Pins

The five pins PA02, PA06, PA08, PA09, and PB01 have high-drive output capabilities. Refer to Section 32. on page 776 for electrical characteristics.

3.4.8 RC32OUT Pin

3.4.8.1 Clock output at startup

After power-up, the clock generated by the 32kHz RC oscillator (RC32K) will be output on PA20, even when the device is still reset by the Power-On Reset Circuitry. This clock can be used by

the system to start other devices or to clock a switching regulator to rise the power supply voltage up to an acceptable value.

The clock will be available on PA20 until one of the following conditions are true:

- PA20 is configured to use a GPIO function other than F (SCIF-RC32OUT)
- PA20 is configured as a General Purpose Input/Output (GPIO)
- The bit FRC32 in the Power Manager PPCR register is written to zero (refer to the Power Manager chapter)

The maximum amplitude of the clock signal will be defined by VDDIN.

3.4.8.2 XOUT32_2 function

PA20 selects RC32OUT as default enabled after reset. This function is not automatically disabled when the user enables the XOUT32_2 function on PA20. This disturbes the oscillator and may result in the wrong frequency. To avoid this, RC32OUT must be disabled when XOUT32_2 is enabled.

3.4.9 ADC Input Pins

These pins are regular I/O pins powered from the VDDIO. However, when these pins are used for ADC inputs, the voltage applied to the pin must not exceed 1.98V. Internal circuitry ensures that the pin cannot be used as an analog input pin when the I/O drives to VDD. When the pins are not used for ADC inputs, the pins may be driven to the full I/O voltage range.

4. Processor and Architecture

Rev: 2.1.0.0

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**
- **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

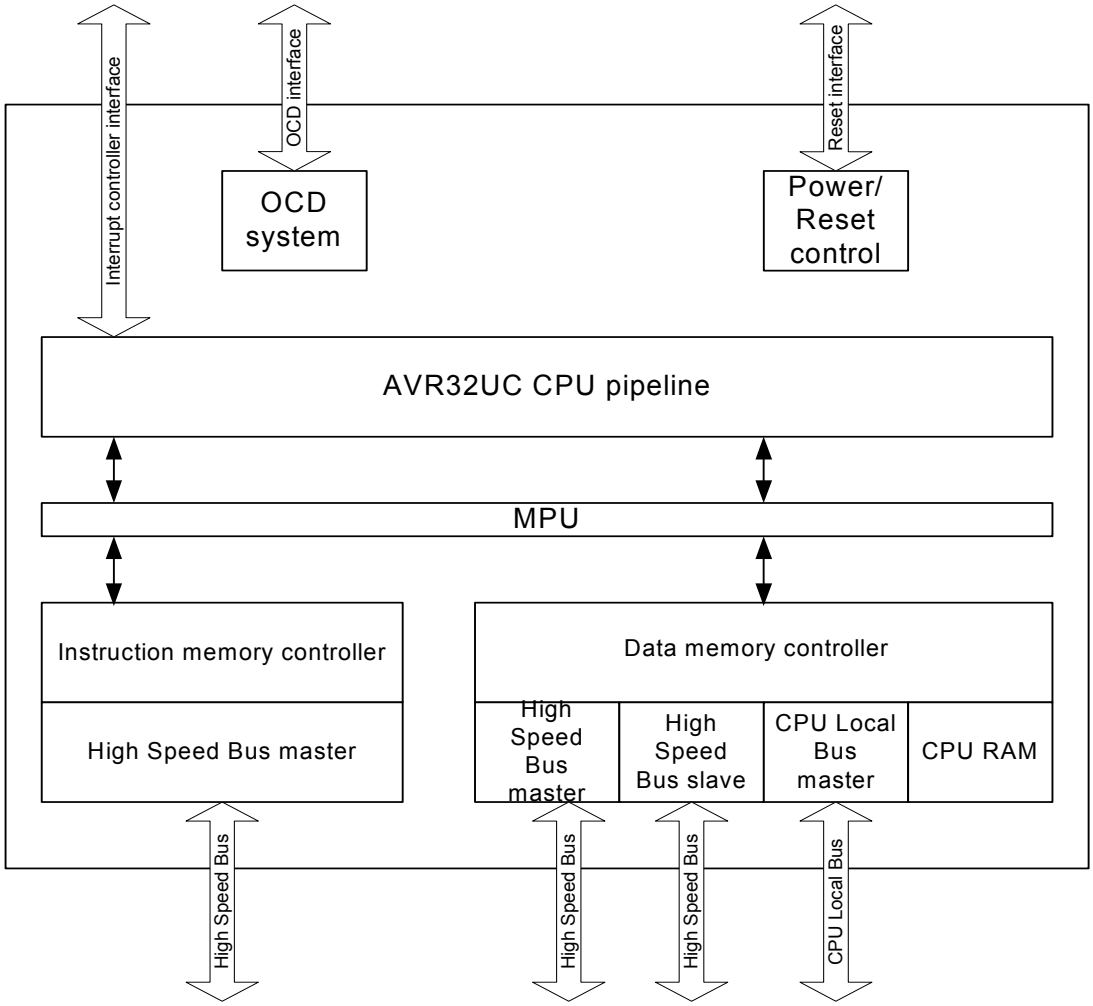
The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced On-Chip Debug (OCD) system, no caches, and a Memory Protection Unit (MPU). Java acceleration hardware is not implemented.

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 20 displays the contents of AVR32UC.

Figure 4-1. Overview of the AVR32UC CPU



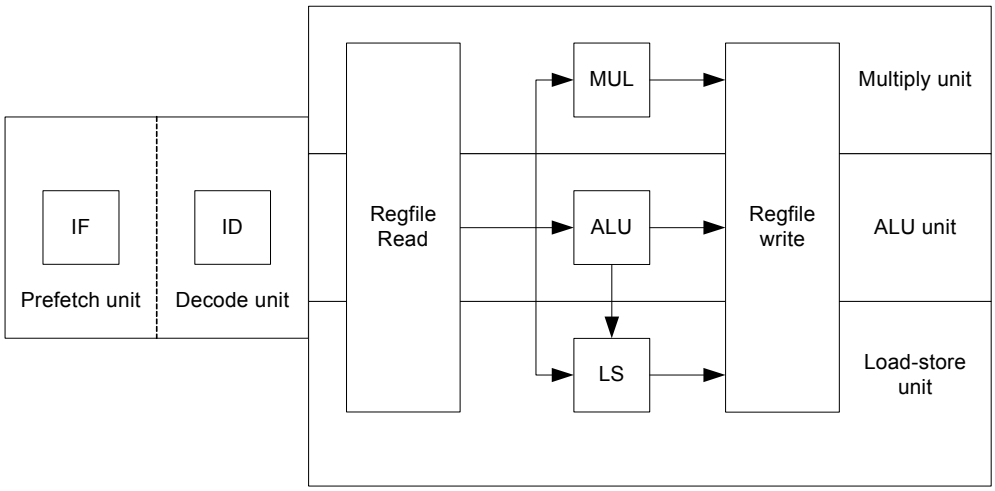
4.3.1 Pipeline Overview

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 21 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 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.

4.3.2.4 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

4.3.2.5 *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

4.3.2.6 *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

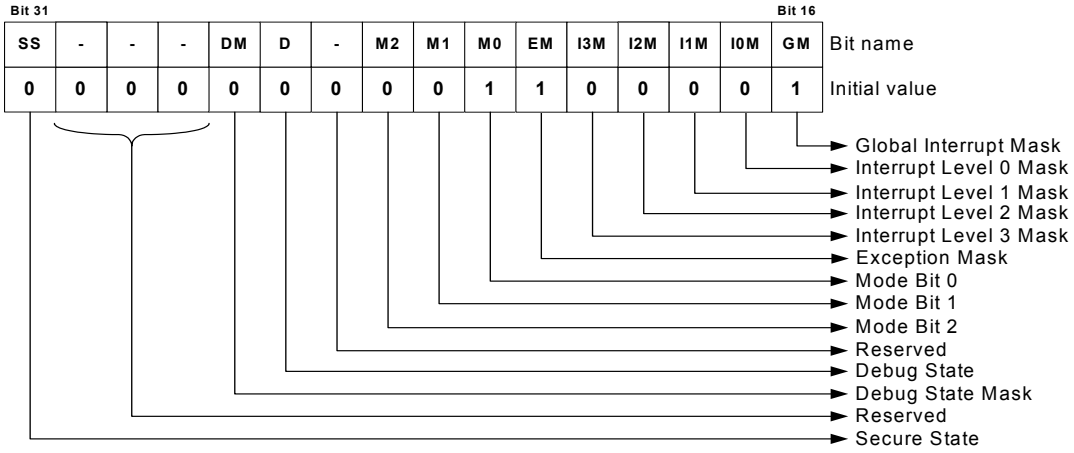
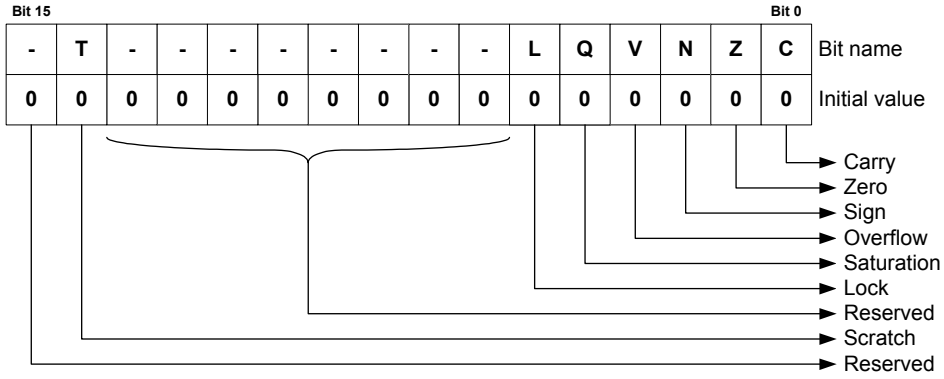


Figure 4-5. The Status Register Low Halfword



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 31. 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 31, 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 31. 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 31. 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**
  - **64 Kbytes (AT32UC3L064)**
  - **32 Kbytes (AT32UC3L032)**
  - **16 Kbytes (AT32UC3L016)**
    - **0 Wait State Access at up to 25 MHz in Worst Case Conditions**
    - **1 Wait State Access at up to 50 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 8% 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**
  - **16 Kbytes (AT32UC3L064, AT32UC3L032)**
  - **8 Kbytes (AT32UC3L016)**

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 AVR32 UC CPU uses unsegmented translation, as described in the AVR32 Architecture Manual. The 32-bit physical address space is mapped as follows:

Table 5-1. AT32UC3L Physical Memory Map

Device	Start Address	Size		
		AT32UC3L064	AT32UC3L032	AT32UC3L016
Embedded SRAM	0x00000000	16 Kbytes	16 Kbytes	8 Kbytes
Embedded Flash	0x80000000	64 Kbytes	32 Kbytes	16 Kbytes
HSB-PB Bridge B	0xFFFE0000	64 Kbytes	64 Kbytes	64 Kbytes
HSB-PB Bridge A	0xFFFF0000	64 Kbytes	64 Kbytes	64 Kbytes

Table 5-2. Flash Memory Parameters

Part Number	Flash Size (*FLASH_PW*)	Number of pages (*FLASH_P*)	Page size (*FLASH_W*)
AT32UC3L064	64 Kbytes	256	256 bytes
AT32UC3L032	32 Kbytes	128	256 bytes
AT32UC3L016	16 Kbytes	64	256 bytes

5.3 Peripheral Address Map

Table 5-3. Peripheral Address Mapping

Address		Peripheral Name	Bus
0xFFFE0000	FLASHCDW	Flash Controller - FLASHCDW	
0xFFFE0400	HMATRIX	HSB Matrix - HMATRIX	
0xFFFE0800	SAU	Secure Access Unit - SAU	
0xFFFF0000	PDCA	Peripheral DMA Controller - PDCA	
0xFFFF1000	INTC	Interrupt controller - INTC	
0xFFFF1400	PM	Power Manager - PM	
0xFFFF1800	SCIF	System Control Interface - SCIF	
0xFFFF1C00	AST	Asynchronous Timer - AST	
0xFFFF2000	WDT	Watchdog Timer - WDT	
0xFFFF2400	EIC	External Interrupt Controller - EIC	
0xFFFF2800	FREQM	Frequency Meter - FREQM	
0xFFFF2C00	GPIO	General Purpose Input/Output Controller - GPIO	
0xFFFF3000	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter - USART0	
0xFFFF3400	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter - USART1	
0xFFFF3800	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter - USART2	
0xFFFF3C00	USART3	Universal Synchronous/Asynchronous Receiver/Transmitter - USART3	
0xFFFF4000	SPI	Serial Peripheral Interface - SPI	
0xFFFF4400	TWIM0	Two-wire Master Interface - TWIM0	

Table 5-3. Peripheral Address Mapping

0xFFFF4800	TWIM1	Two-wire Master Interface - TWIM1
0xFFFF4C00	TWIS0	Two-wire Slave Interface - TWIS0
0xFFFF5000	TWIS1	Two-wire Slave Interface - TWIS1
0xFFFF5400	PWMA	Pulse Width Modulation Controller - PWMA
0xFFFF5800	TC0	Timer/Counter - TC0
0xFFFF5C00	TC1	Timer/Counter - TC1
0xFFFF6000	ADCIFB	ADC Interface - ADCIFB
0xFFFF6400	ACIFB	Analog Comparator Interface - ACIFB
0xFFFF6800	CAT	Capacitive Touch Module - CAT
0xFFFF6C00	GLOC	Glue Logic Controller - GLOC
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
0	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
1	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

6. Supply and Startup Considerations

6.1 Supply Considerations

6.1.1 Power Supplies

The AT32UC3L has several types of power supply pins:

- VDDIO: Powers I/O lines. Voltage is 1.8 to 3.3V nominal.
- VDDIN: Powers I/O lines and the internal regulator. Voltage is 1.8 to 3.3V nominal.
- VDDANA: Powers the ADC. Voltage is 1.8V nominal.
- VDDCORE: Powers the core, memories, and peripherals. Voltage is 1.8V nominal.

The ground pins GND are common to VDDCORE and VDDIO. The ground pin for VDDANA is GNDANA.

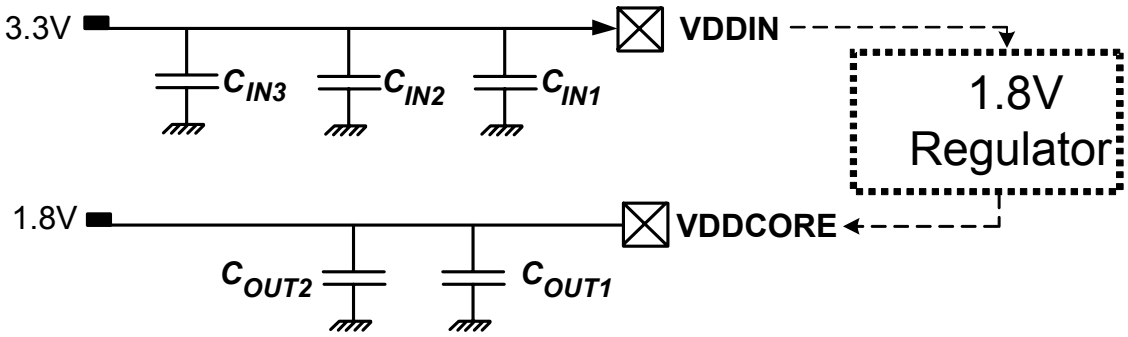
Refer to Section 32. on page 776 for power consumption on the various supply pins.

6.1.2 Voltage Regulator

The AT32UC3L embeds a voltage regulator that converts from 3.3V nominal to 1.8V with a load of up to 60 mA. The regulator supplies the output voltage on VDDCORE. The regulator may only be used to drive internal circuitry in the device. VDDCORE should be externally connected to the 1.8V domains. See Section 6.1.3 for regulator connection figures.

Adequate output supply decoupling is mandatory for VDDCORE to reduce ripple and avoid oscillations. The best way to achieve this is to use two capacitors in parallell between VDDCORE and GND as close to the chip as possible. Please refer to Section 32.9.1 on page 785 for decoupling capacitors values and regulator characteristics.

Figure 6-1. Supply Decoupling



6.1.3 Regulator Connection

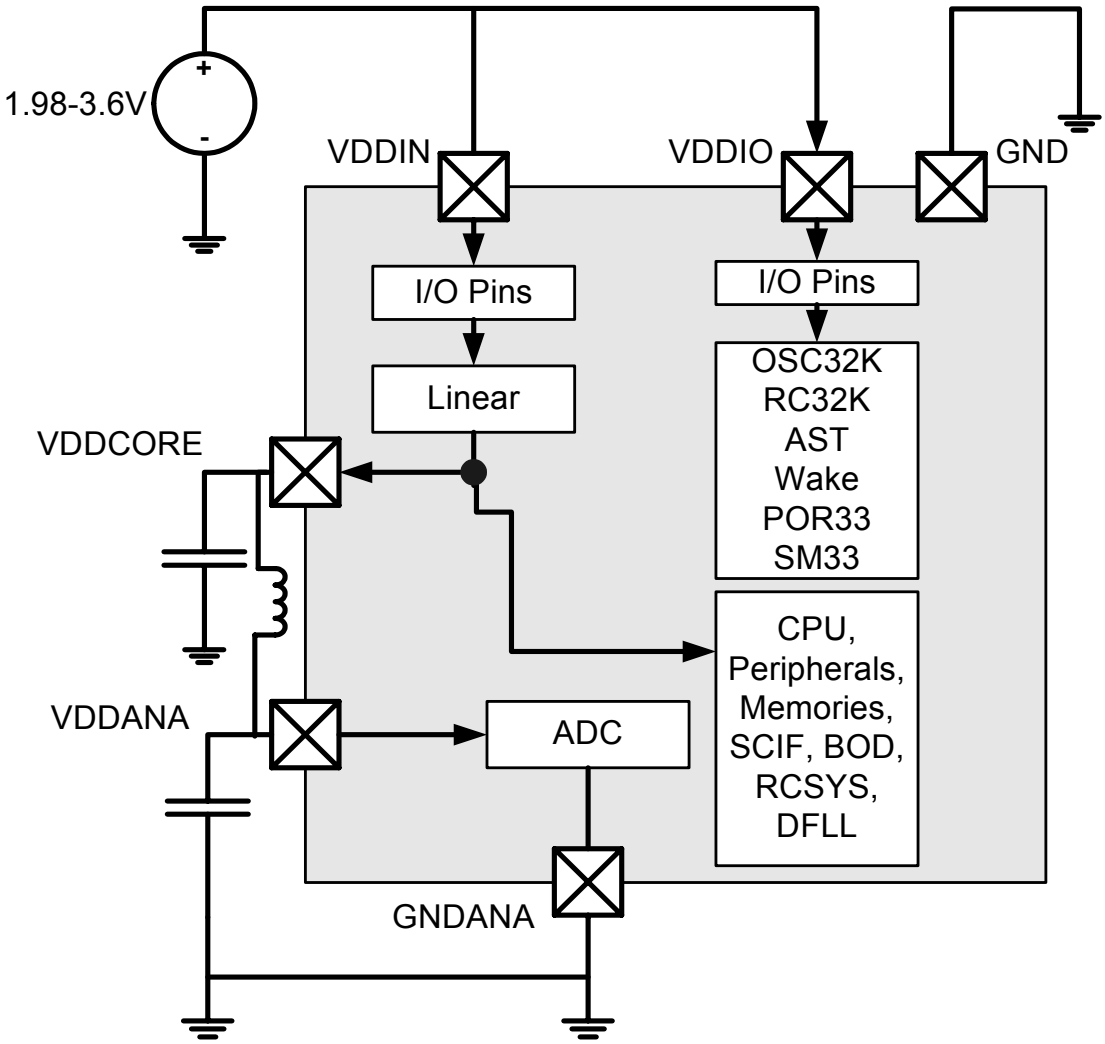
The AT32UC3L supports three power supply configurations:

- 3.3V single supply mode
- 1.8V single supply mode
- 3.3V supply mode, with 1.8V regulated I/O lines

6.1.3.1 3.3V Single Supply Mode

In 3.3V single supply mode the internal regulator is connected to the 3.3V source (VDDIN pin) and its output feeds VDDCORE. Figure 6-2 shows the power schematics to be used for 3.3V single supply mode. All I/O lines will be powered by the same power (VDDIN=VDDIO).

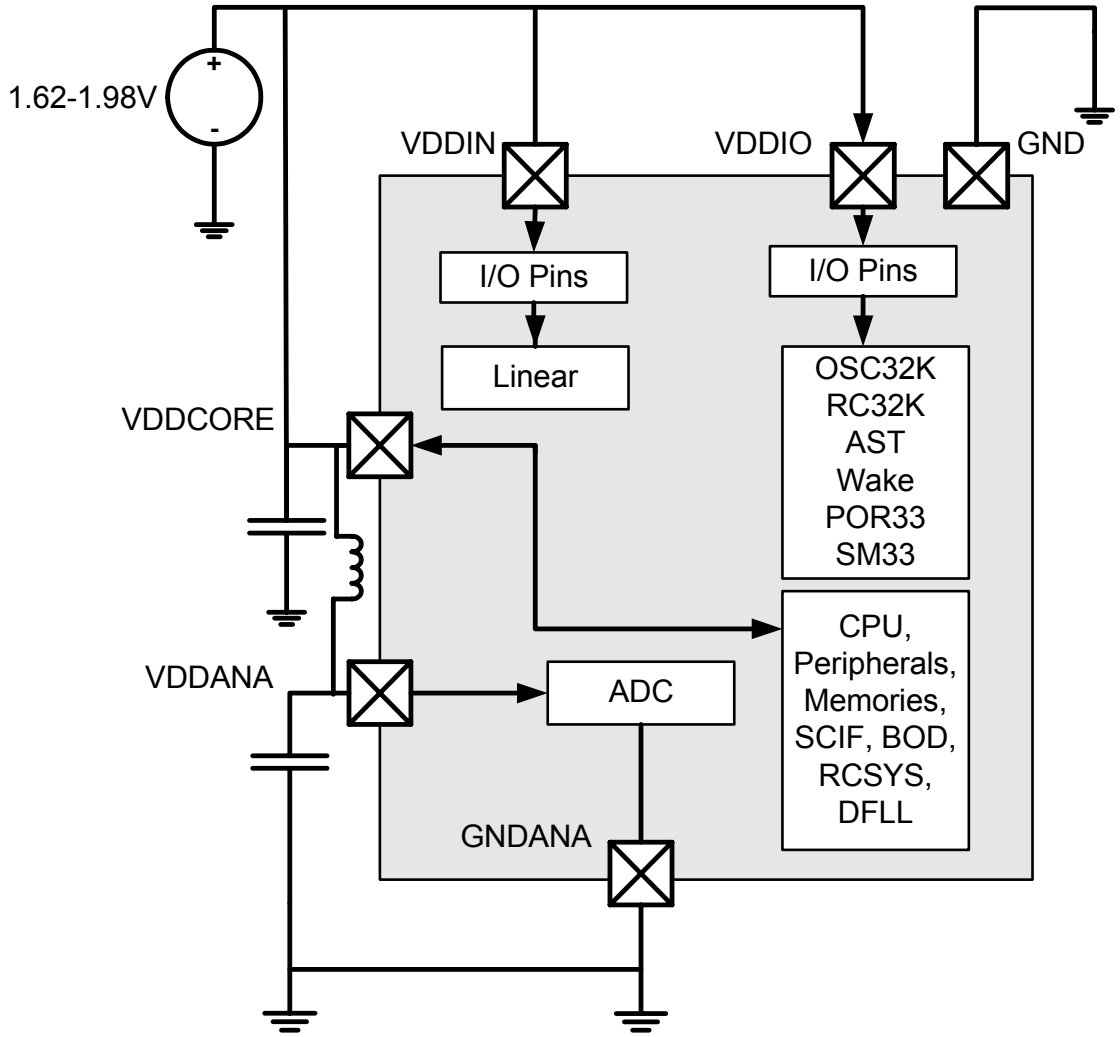
Figure 6-2. 3.3V Single Power Supply mode



6.1.3.2 *1.8V Single Supply Mode*

In 1.8V single supply mode the internal regulator is not used, and VDDIO and VDDCORE are powered by a single 1.8V supply as shown in Figure 6-3. All I/O lines will be powered by the same power (VDDIN = VDDIO = VDDCORE).

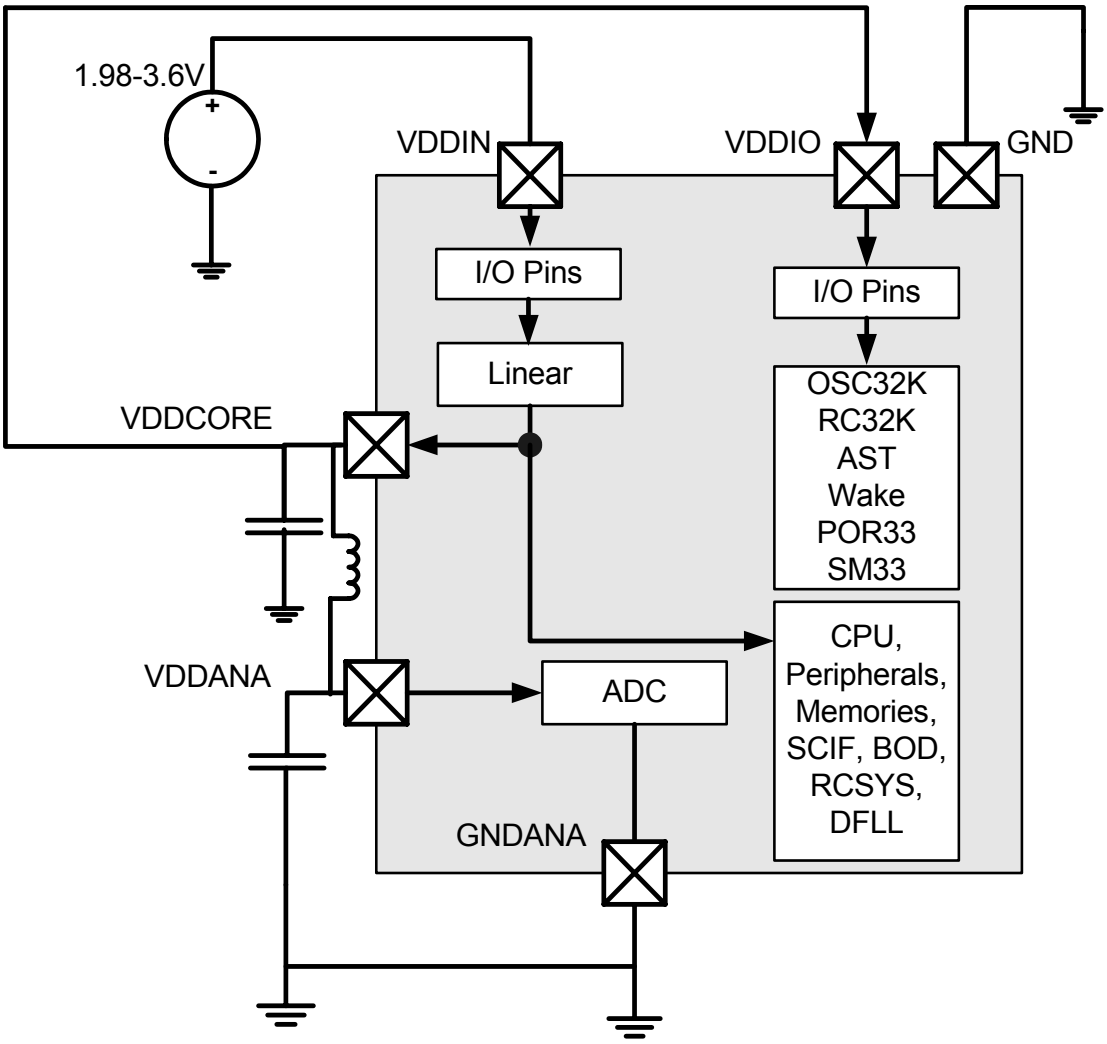
Figure 6-3. 1.8V Single Power Supply Mode.



6.1.3.3 *3.3V Supply Mode with 1.8V Regulated I/O Lines*

In this mode, the internal regulator is connected to the 3.3V source and its output is connected to both VDDCORE and VDDIO as shown in Figure 6-4. This configuration is required in order to use Shutdown mode.

Figure 6-4. 3.3V Power with 1.8V Regulated I/O Lines



In this mode, some I/O lines are powered by VDDIN while others I/O lines are powered by VDDIO. Refer to Section 3.2 on page 9 for description of power supply for each I/O line.

Refer to the Power Manager chapter for a description of what parts of the system are powered in Shutdown mode.

Important note: As the regulator has a maximum output current of 60mA, this mode can only be used in applications where the maximum I/O current is known and compatible with the core and peripheral power consumption. Typically, great care must be used to ensure that only a few I/O lines are toggling at the same time and drive very small loads.

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 32-3 on page 777.

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

6.1.4.2 Minimum Rise Rate

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

See Table 32-3 on page 777 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, one of the following configuration can be used:

- A logic “0” value is applied during power-up on pin PA11 until VDDIN rises above 1.2V.
- A logic “0” value is applied during power-up on pin RESET_N until VDDIN rises above 1.2V.

6.2 Startup Considerations

This chapter summarizes the boot sequence of the AT32UC3L. 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

After power-up, the device will be held in a reset state by the Power-On Reset circuitry for a short time to allow the power to stabilize throughout the device. After reset, the device will use the System RC Oscillator (RCSYS) as clock source. Please refer to Table 32-17 on page 784 for the frequency for this oscillator.

On system start-up, the DFLL is 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 System RC Oscillator.

When powering up the device, there may be a delay before the voltage has stabilized, depending on the rise time of the supply used. The CPU can start executing code as soon as the supply is above the POR threshold, and before the supply is stable. Before switching to a high-speed clock source, the user should use the BOD to make sure the VDDCORE is above the minimum level (1.62V).

6.2.2 Fetching of Initial Instructions

After reset has been released, the AVR32 UC CPU starts fetching instructions from the reset address, which is 0x80000000. This address points to the first address in the internal Flash.

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

7. Flash Controller (FLASHCDW)

Rev: 1.0.2.0

7.1 Features

- **Controls on-chip flash memory**
- **Supports 0 and 1 wait state bus access**
- **Buffers reducing penalty of wait state in sequential code or loops**
- **Allows interleaved burst reads for systems with one wait state, outputting one 32-bit word per clock cycle for sequential reads**
- **Supports AVR32 Secure State**
- **32-bit HSB interface for reads from flash and writes to page buffer**
- **32-bit PB interface for issuing commands to and configuration of the controller**
- **Flash memory is divided into 16 regions can be individually protected or unprotected**
- **Additional protection of the Boot Loader pages**
- **Supports reads and writes of general-purpose Non Volatile Memory (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**

7.2 Overview

The Flash Controller (FLASHCDW) interfaces the on-chip flash memory with the 32-bit internal HSB bus. The controller manages the reading, writing, erasing, locking, and unlocking sequences.

7.3 Product Dependencies

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

7.3.1 Power Management

If the CPU enters a sleep mode that disables clocks used by the FLASHCDW, the FLASHCDW will stop functioning and resume operation after the system wakes up from sleep mode.

7.3.2 Clocks

The FLASHCDW has two bus clocks connected: One High Speed Bus clock (CLK_FLASHCDW_HSB) and one Peripheral Bus clock (CLK_FLASHCDW_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_FLASHCDW_HSB is not turned off before reading the flash or writing the pagebuffer and that CLK_FLASHCDW_PB is not turned off before accessing the FLASHCDW configuration and control registers. Failing to do so may deadlock the bus.

7.3.3 Interrupts

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

7.3.4 Debug Operation

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

7.4 Functional Description

7.4.1 Bus Interfaces

The FLASHCDW 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.

7.4.2 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 User page

7.4.3 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 7-1 on page 44.

7.4.4 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 FLASHCDW 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 FLASHCDW 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 FLASHCDW 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 internal read word width of the flash memory is 64 bits. When a 32-bit word is to be addressed, the word itself and

also the other word in the same 64-bit location is read. The first word is output on the bus, and the other word is put into an internal buffer. If a read to a sequential address is to be performed in the next cycle, the buffered word is output on the bus, while the next 64-bit location is read from the flash memory. Thus, latency in 1 wait state mode is hidden for sequential fetches.

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. In 1 ws mode, if the first access in a burst access is to an address that is not 64-bit aligned, an additional stall cycle is also encountered when reading the second word in the burst. All subsequent words in the burst are accessed without any stall cycles.

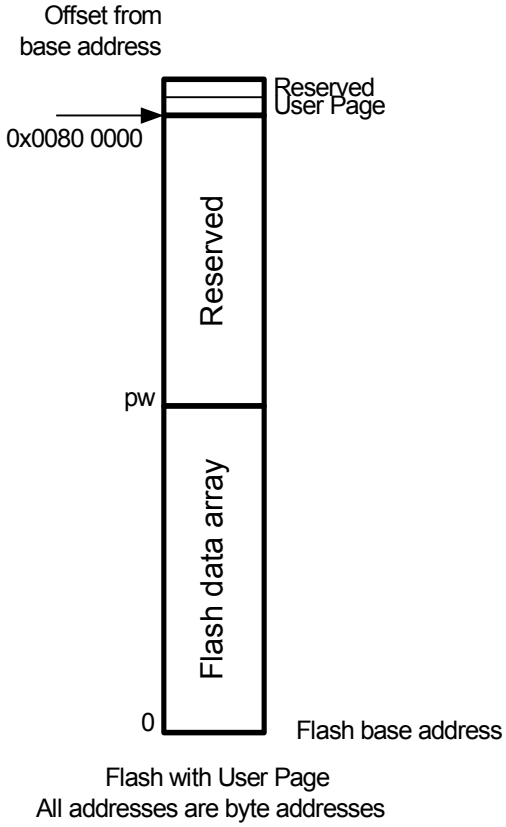
The Flash Controller provides two sets of buffers that can be enabled in order to speed up instruction fetching. These buffers can be enabled by writing a one to the FCR.SEQBUF and FCR.BRBUF bits. The SEQBUF bit enables buffering hardware optimizing sequential instruction fetches. The BRBUF bit enables buffering hardware optimizing tight inner loops. These buffers are never used when the flash is in 0 wait state mode. Usually, both these buffers should be enabled when operating in 1 wait state mode. Some users requiring absolute cycle determinism may want to keep the buffers disabled.

The Flash Controller address space is displayed in Figure 7-1. 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 0x00800000 from the start address of the flash memory.

Table 7-1. User Page Addresses

Memory type	Start address, byte sized	Size
Main array	0	*pw* words = 4*pw* bytes
User	0x00800000	64 words = 256 bytes

Figure 7-1. Memory Map for the Flash Memories

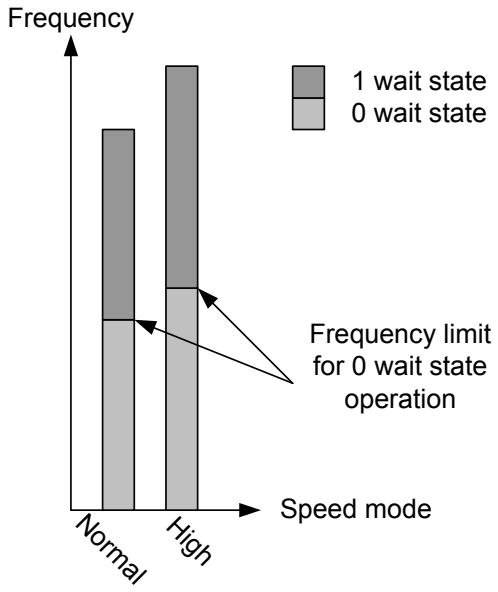


7.4.5 High Speed Read Mode

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) bit in the 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 7-2. High Speed Mode



7.4.6 Quick Page Read

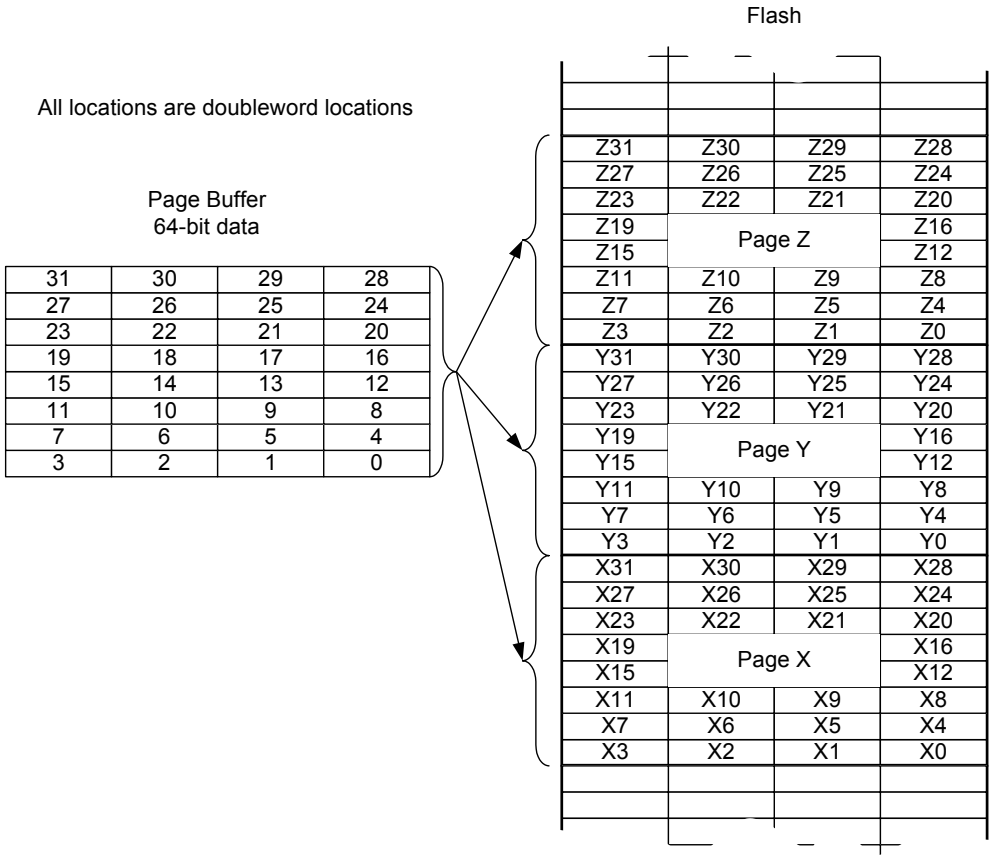
A dedicated command, Quick Page Read (QPR), is provided to read all words in an addressed page. All bits in all words in this page are AND'ed together, returning a 1-bit result. This result is placed in the Quick Page Read Result (QPRR) bit in Flash Status Register (FSR). The QPR command is useful to 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.

7.4.7 Page Buffer Operations

The flash memory has a write and erase granularity of one page; data is 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, Refer to Section 7.5.1 on page 48.

In order to program data into flash page Y, write the desired data to locations Y0 to Y31 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%32 in the page buffer. The PAGEN field in the Flash Command (FCMD) register will at the same time be updated with the value A/32.

Figure 7-3. Mapping from Page Buffer to Flash



Internally, the flash memory stores data in 64-bit doublewords. Therefore, the native data size of the Page Buffer is also a 64-bit doubleword. All locations shown in Figure 7-3 are therefore doubleword locations. Since the HSB bus only has a 32-bit data width, two 32-bit HSB transfers must be performed to write a 64-bit doubleword into the Page Buffer. The FLASHCDW has logic to combine two 32-bit HSB transfers into a 64-bit data before writing this 64-bit data into the Page Buffer. This logic requires the word with the low address to be written to the HSB bus before the word with the high address. To exemplify, to write a 64-bit value to doubleword X0 residing in page X, first write a 32-bit word to the byte address pointing to address X0, thereafter write a word to the byte address pointing to address (X0+4).

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 FLASHCDW 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.

7.4.8 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. Note that “one word” for the FLASCHDW is actually 64 bits. Only words that are in a completely erased state (0xFFFFFFFFFFFFFFFF) 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.

7.5 Flash Commands

The FLASHCDW offers a command set to manage programming of the flash memory, locking and unlocking of regions, and full flash erasing. See Section 7.8.2 for a complete list of commands.

To run a command, the CMD field in the Flash Command Register (FCMD) has to be written with the command number. As soon as the FCMD register is written, the FRDY bit in the Flash Status Register (FSR) is automatically cleared. Once the current command is complete, the FSR.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 FCR.FRDY is one.

Any HSB bus transfers attempting to read flash memory when the FLASHCDW 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 FLASHCDW is IDLE. The user must make sure that the access pattern to the FLASHCDW 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.

7.5.1 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 is stored in an internal buffer called the 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 7.4.7 on page 45.
- 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 FRDY bit in the Flash Status Register (FSR) is set. If an interrupt was enabled by writing FCR.FRDY to one, an interrupt request is generated.

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.

7.5.2 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 LOCKE bit in FCR is one, an interrupt request is set generated.

When the command is complete, the FRDY bit in the Flash Status Register (FSR) is set. If an interrupt has been enabled by writing FCR.FRDY to one, an interrupt request is generated. 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.

7.5.3 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 7.6. 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 7.6.

7.6 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 7-2. 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 or aWire 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 or aWire chip erase. BOOTPROT can only be changed when the security bit is cleared. If the security bit is set, only an external JTAG or aWire 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. Decoded into SSE and SSDE signals as shown in Table 7-4. Refer to the AVR32 Architecture Manual and the AVR32UC Technical Reference Manual for more details on SSE and SSDE.
22	UPROT	If programmed (i.e. “0”), 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 7-3. Boot Loader Area Specified by BOOTPROT

BOOTPROT	Pages protected by BOOTPROT	Size of protected memory
7	None	0
6	0-1	512 byte
5	0-3	1 Kbyte
4	0-7	2 Kbyte
3	0-15	4 Kbyte
2	0-31	8 Kbyte
1	0-63	16 Kbyte
0	0-127	32 Kbyte

The SECURE fuses have the following functionality:

Table 7-4. Secure State Configuration

SECURE	Functionality	SSE	SSDE
00	Secure state disabled	0	0
01	Secure enabled, secure state debug enabled	1	1
10	Secure enabled, secure state debug disabled	1	0
11	Secure state disabled	0	0

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 7.5.3.

7.7 Security Bit

The security bit allows the entire chip to be locked from external JTAG, aWire, 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 or aWire 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.

7.8 User Interface

Table 7-5. FLASHCDW Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Flash Control Register	FCR	Read/Write	0x00000000
0x04	Flash Command Register	FCMD	Read/Write	0x00000000
0x08	Flash Status Register	FSR	Read/Write	-(1)
0x0C	Flash Parameter Register	FPR	Read-only	-(3)
0x10	Flash Version Register	FVR	Read-only	-(3)
0x14	Flash General Purpose Fuse Register Hi	FGPFRHI	Read-only	-(2)
0x18	Flash General Purpose Fuse Register Lo	FGPFRLO	Read-only	-(2)

Note:
1. The value of the Lock bits depend on their programmed state. All other bits in FSR are 0.
2. All bits in FGPRHI/LO are dependent on the programmed state of the fuses they map to. Any bits in these registers not mapped to a fuse read as 0.
3. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

7.8.1 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
-	-	-	-	-	BRBUF	SEQBUF	-

7	6	5	4	3	2	1	0
-	FWS	-	-	PROGE	LOCKE	-	FRDY

- **BRBUF: Branch Target Instruction Buffer Enable**
  0: The Branch Target Instruction Buffer is disabled.
  1: The Branch Target Instruction Buffer is enabled.
- **SEQBUF: Sequential Instruction Fetch Buffer Enable**
  0: The Sequential Instruction Fetch Buffer is disabled.
  1: The Sequential Instruction Fetch Buffer is enabled.
- **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 request.
  1: Programming Error generates an interrupt request.
- **LOCKE: Lock Error Interrupt Enable**
  0: Lock Error does not generate an interrupt request.
  1: Lock Error generates an interrupt request.
- **FRDY: Flash Ready Interrupt Enable**
  0: Flash Ready does not generate an interrupt request.
  1: Flash Ready generates an interrupt request.

7.8.2 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 in FSR 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 7-6. 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 7-6. 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 in FSR to be set, and the corresponding interrupt to be requested if the PROGE bit in FCR is one.

Table 7-7. 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
RESERVED	16-31	

7.8.3 Flash Status Register

Name: FSR

Access Type: Read/Write

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.

7.8.4 Flash Parameter Register

Name: FPR

Access Type: Read-only

Offset: 0x0C

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
-	-	-	-	-	PSZ		

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

- **PSZ: Page Size**
  The size of each flash page.

Table 7-8. Flash Page Size

PSZ	Page Size
0	32 Byte
1	64 Byte
2	128 Byte
3	256 Byte
4	512 Byte
5	1024 Byte
6	2048 Byte
7	4096 Byte

- **FSZ: Flash Size**
  The size of the flash. Not all device families will provide all flash sizes indicated in the table.

Table 7-9. 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

7.8.5 Flash Version Register

Name: FVR

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.

7.8.6 Flash General Purpose Fuse Register High

Name: FGPFRHI

Access Type: Read-only

Offset: 0x14

Reset Value: -

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.

7.8.7 Flash General Purpose Fuse Register Low

Name: FGPFRLO

Access Type: Read-only

Offset: 0x18

Reset Value: -

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.

7.9 Fuse Settings

The flash block contains 32 general purpose fuses. These 32 fuses can be found in the Flash General Purpose Fuse Register Low (FGPFRLO). The Flash General Purpose Fuse Register High (FGPFRHI) is not used. 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.

The general purpose fuses are erased by a JTAG or aWire chip erase.

7.9.1 Flash General Purpose Fuse Register Low (FGPFRLO)

31	30	29	28	27	26	25	24
BODEN		BODHYST	BODLEVEL[5:1]				

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]							

BODEN: Brown Out Detector Enable

BODEN	Description
00	BOD disabled
01	BOD enabled, BOD reset enabled
10	BOD enabled, BOD reset disabled
11	Reserved

BODHYST: Brown Out Detector Hysteresis

0: The Brown out detector hysteresis is disabled

1: The Brown out detector hysteresis is enabled

BODLEVEL: Brown Out Detector Trigger Level

This controls the voltage trigger level for the Brown out detector. Refer to ”Electrical Characteristics” on page 776.

UPROT, SECURE, BOOTPROT, EPFL, LOCK

These are Flash Controller fuses and are described in the FLASHCDW section.

7.9.1.1 *Default Fuse Value*

The devices are shipped with the FGPFRLO register value: 0xE075FFFF:

- BODEN fuses set to 11. BOD is disabled.
- BODHYST fuse set to 1. The BOD hysteresis is enabled.
- BODLEVEL fuses set to 000000. This is the minimum voltage trigger level for BOD.
- UPROT fuse set to 1.
- SECURE fuse set to 11.
- BOOTPROT fuses set to 010. The bootloader protected size is 8KBytes.
- EPFL fuse set to 1. External privileged fetch is not locked.
- LOCK fuses set to 1111111111111111. No region locked.

After the JTAG or aWire chip erase command, the FGPFR register value is 0xFFFFFFFF.

7.9.2 First Word of the User Page (Address 0x80800000)

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

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

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

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

WDTAUTO: WatchDog Timer Auto Enable at Startup

0: The WDT is automatically enabled at startup.

1: The WDT is not automatically enabled at startup.

Please refer to the WDT chapter for detail about timeout settings when the WDT is automatically enabled.

7.9.2.1 *Default user page first word value*

The devices are shipped with the User page erased (all bits 1):

- WDTAUTO set to 1, WDT disabled.

7.9.3 Second Word of the User Page (Address 0x80800004)

31	30	29	28	27	26	25	24
SSADRR[15:8]							

23	22	21	20	19	18	17	16
SSADRR[7:0]							

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

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

SSADRR: Secure State End Address for the RAM

SSADRF: Secure State End Address for the Flash

7.9.3.1 Default user page second word value

The devices are shipped with the User page erased (all bits 1).

7.10 Module Configuration

The specific configuration for each FLASHCDW 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 7-10. Module Configuration

Feature	AT32UC3L064	AT32UC3L032	AT32UC3L016
Flash size	64Kbytes	32Kbytes	16Kbytes
Number of pages	256	128	64
Page size	256 bytes	256 bytes	256 bytes

Table 7-11. Module Clock Name

Module Name	Clock Name	Clock Name
FLASHCDW	CLK_FLASHCDW_HSB	CLK_FLASHCDW_PB

Table 7-12. Register Reset Values

Register	Reset Value
FVR	0x00000102
FPR	0x00000305

8. HSB Bus Matrix (HMATRIX)

Rev: 1.3.0.3

8.1 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**
- **Three different memory mappings for each master (internal and external boot, remap)**
- **One remap function 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)**

8.2 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.

8.3 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.

8.3.1 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.

8.4 Functional Description

8.4.1 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.

8.4.1.1 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.

8.4.1.2 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.

8.4.1.3 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).

8.4.2 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” .

8.4.2.1 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.

8.4.2.2 *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.

8.4.2.3 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).

8.4.3 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.

8.5 User Interface

Table 8-1. 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 8-1. 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 8-1. 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	–

8.5.1 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 8-2. 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.

8.5.2 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.

8.5.3 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.

8.5.4 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.

8.5.5 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.

8.6 Module Configuration

8.6.1 Bus Matrix Connections

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, HMATRIX MCFG0 register is associated with the CPU Data master interface.

Table 8-3. High Speed Bus Masters

Master 0	CPU Data
Master 1	CPU Instruction
Master 2	CPU SAB
Master 3	SAU
Master 4	PDCA

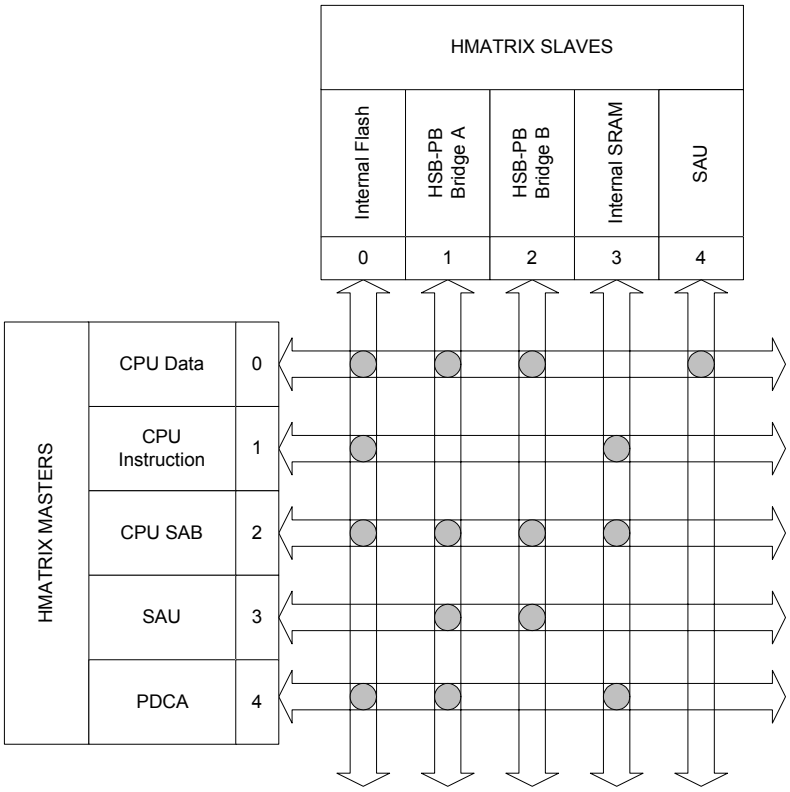
Each slave has its own arbiter, thus allowing a different arbitration per slave. The slave number in the table below can be used to index the HMATRIX control registers. For example, SCFG3 is associated with the Internal SRAM Slave Interface.

Accesses to unused areas returns an error result to the master requesting such an access.

Table 8-4. High Speed Bus Slaves

Slave 0	Internal Flash
Slave 1	HSB-PB Bridge A
Slave 2	HSB-PB Bridge B
Slave 3	Internal SRAM
Slave 4	SAU

Figure 8-1. HMatrix Master / Slave Connections



9. Secure Access Unit (SAU)

Rev 1.1.0.1

9.1 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**

9.2 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.

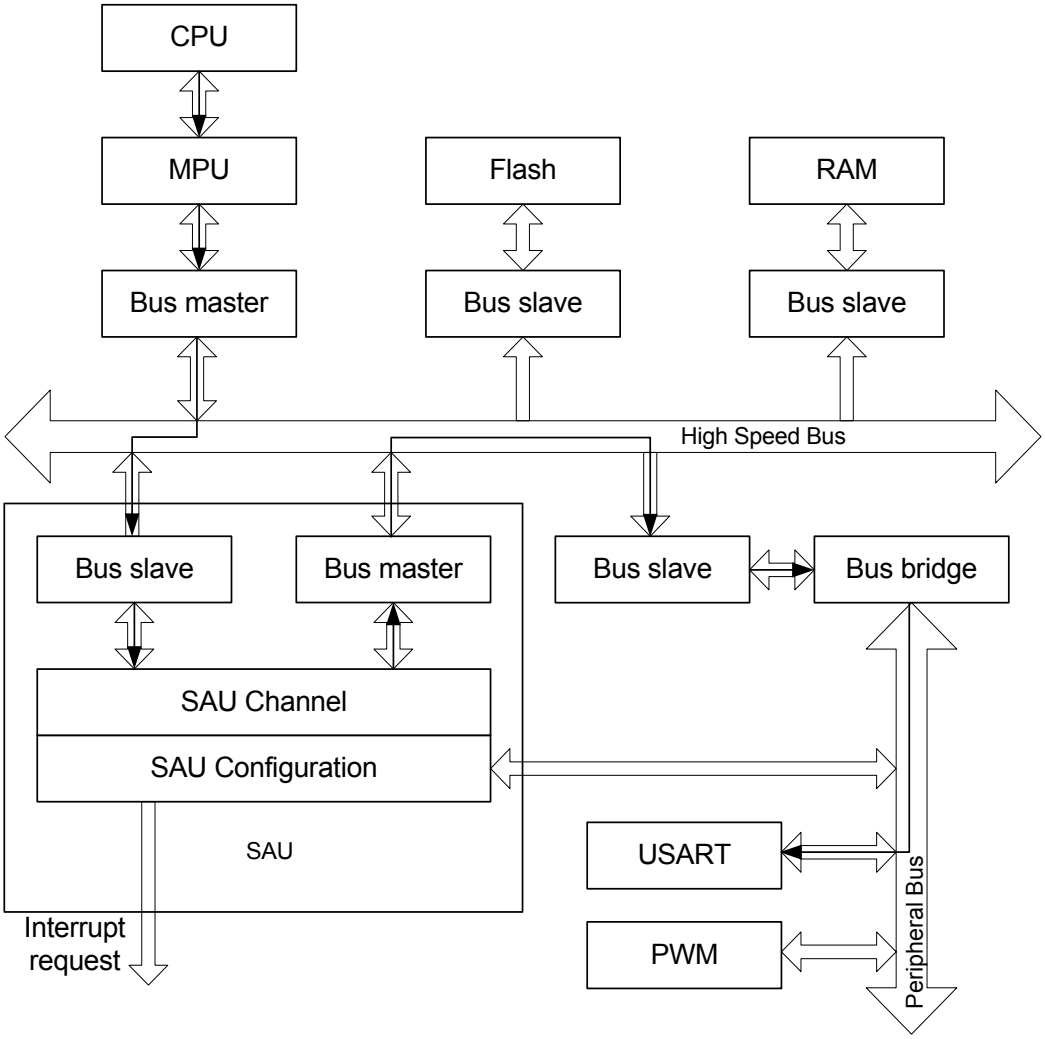
Usually, the Secure Access Unit (SAU) is used together with the MPU to provide the required security and integrity. The MPU is set up to protect regions of memory, while the SAU is set up to provide a secure channel into specific memory locations that are protected by the MPU. These specific locations can be thought of as fine-grained overrides of the general coarse-grained protection provided by the MPU.

9.3 Block Diagram

Figure 9-1 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 9-1 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 9-1.

Figure 9-1. SAU 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

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.

9.4.2 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.

9.4.3 Interrupt

The SAU interrupt request line is connected to the interrupt controller. Using the SAU interrupt requires the interrupt controller to be programmed first.

9.4.4 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.

9.5 Functional Description

9.5.1 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).

9.5.2 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.

9.5.2.1 *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.

9.5.3 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 key value must be written to UR.KEY, and the channel number to UR.CHANNEL. 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.

9.5.4 Normal Operation

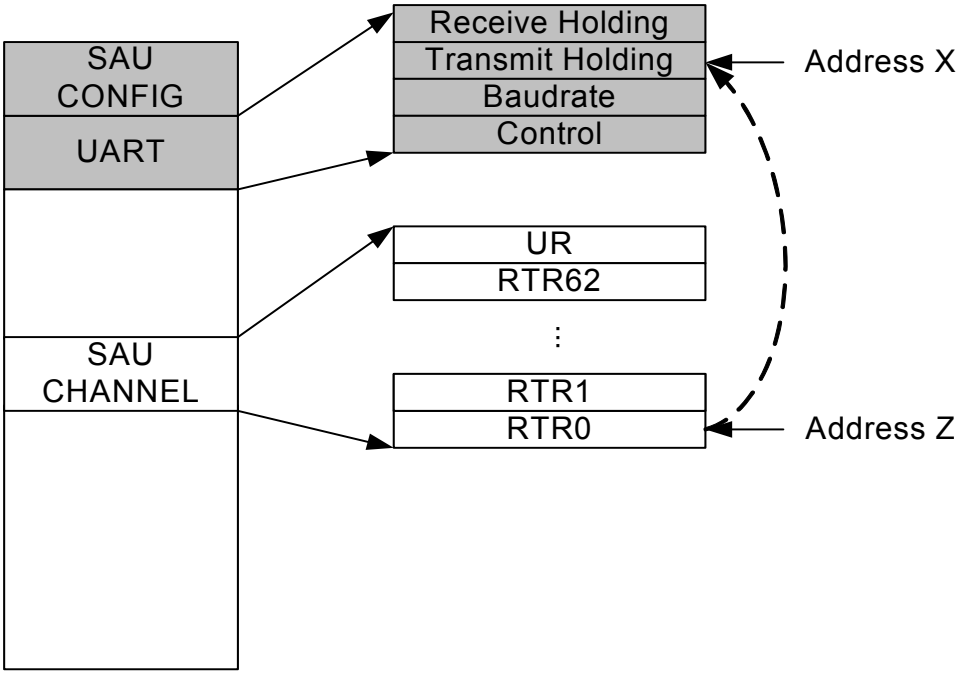
The following sequence must be used in order to access a SAU channel in normal operation (CR.SEN=0):

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.

9.5.4.1 *Operation example*

Figure 9-2 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 9-2. Example Memory Map for a System with SAU



9.5.5 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 9.5.7 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 9.5.6). 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.

9.5.6 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

9.5.7 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

9.5.8 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 STATUS.IDLE bit is set.

The SAU may then be disabled by writing a one to the Disable (DIS) bit in CR.

9.6 User Interface

The following addresses are used by SAU channel configuration registers. All offsets are relative to the SAU's PB base address.

Table 9-1. 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	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 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 9-2. 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

9.6.1 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.

9.6.2 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
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
UCYC							

7	6	5	4	3	2	1	0
UKEY							

- **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.

9.6.3 Channel Enable Register High

Name: CERH

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

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

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.

9.6.4 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.

9.6.5 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
-	-	-	-	-	-	-	-

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 the operation is completed and no SAU bus operations are pending.
  This bit is set when a read or write operation to the SAU channel is started.
- **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.

9.6.6 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.

9.6.7 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.

9.6.8 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.

9.6.9 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.

9.6.10 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**

9.6.11 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.

9.6.12 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.

9.6.13 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**
  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**
  Number of the channel to unlock.

9.7 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. Please refer to the Power Manager chapter for details.

Table 9-3. Module configuration

Feature	SAU
SAU Channels	16

Table 9-4. Module clock name

Module name	Clock name
SAU	**CLK_SAU_HSB**
SAU	**CLK_SAU_PB**

Table 9-5. Register Reset Values

Register	Reset Value
VERSION	0x00000110
PARAMETER	0x00000010

10. Peripheral DMA Controller (PDCA)

Rev: 1.2.2.1

10.1 Features

- **Multiple channels**
- **Generates transfers to/from 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**

10.2 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:

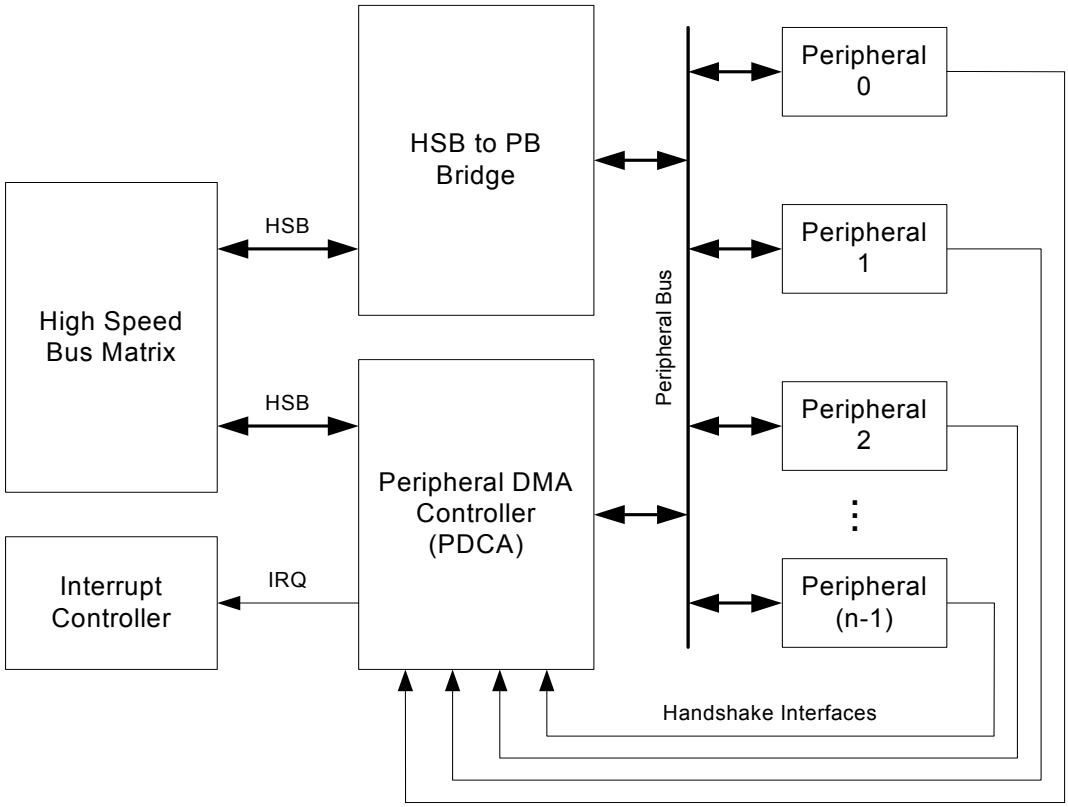
- A Peripheral Select Register
- A 32-bit memory pointer
- A 16-bit transfer counter
- A 32-bit memory pointer reload value
- A 16-bit transfer counter reload value

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.

10.3 Block Diagram

Figure 10-1. PDCA 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

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.

10.4.2 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.

10.4.3 Interrupts

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

10.4.4 Peripheral Events

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

10.5 Functional Description

10.5.1 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 10.5.6.

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 10.5.7.

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 10.5.2.

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 10.5.3.

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).

10.5.2 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.

10.5.3 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.

10.5.4 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.

10.5.5 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).

10.5.6 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 PST 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.

10.5.7 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 differen peripherals and then to the peripheral specific chapter for information about the size option available for the different registers.

10.5.8 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.

10.5.9 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.

10.5.10 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.

10.5.11 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.

10.5.12 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.

10.6 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.

10.6.1 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).

10.7 User Interface

10.7.1 Memory Map Overview

Table 10-1. 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 10-1. Each channel has a set of configuration registers, shown in Table 10-2, where *n* is the channel number.

10.7.2 Channel Memory Map

Table 10-2. 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.

10.7.3 Performance Monitor Memory Map

Table 10-3. 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.

10.7.4 Version Register Memory Map

Table 10-4. 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.

10.7.5 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.

10.7.6 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.

10.7.7 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.

10.7.8 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.

10.7.9 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.

10.7.10 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.

10.7.11 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 10-5. Size of Transfer

SIZE	Size of Transfer
0	Byte
1	Halfword
2	Word
3	Reserved

10.7.12 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.

10.7.13 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.

10.7.14 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.

10.7.15 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.

10.7.16 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.

10.7.17 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.

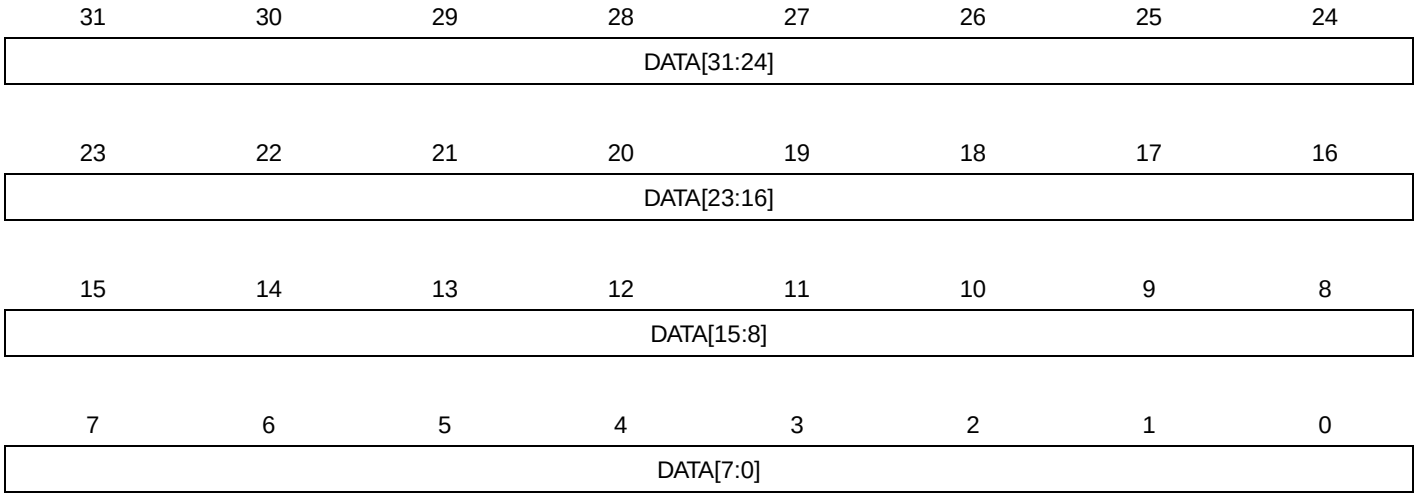
10.7.18 Performance Channel 0 Read Data Cycles

Name: PRDATA0

Access Type: Read-only

Offset: 0x804

Reset Value: 0x00000000



- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

10.7.19 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

10.7.20 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.

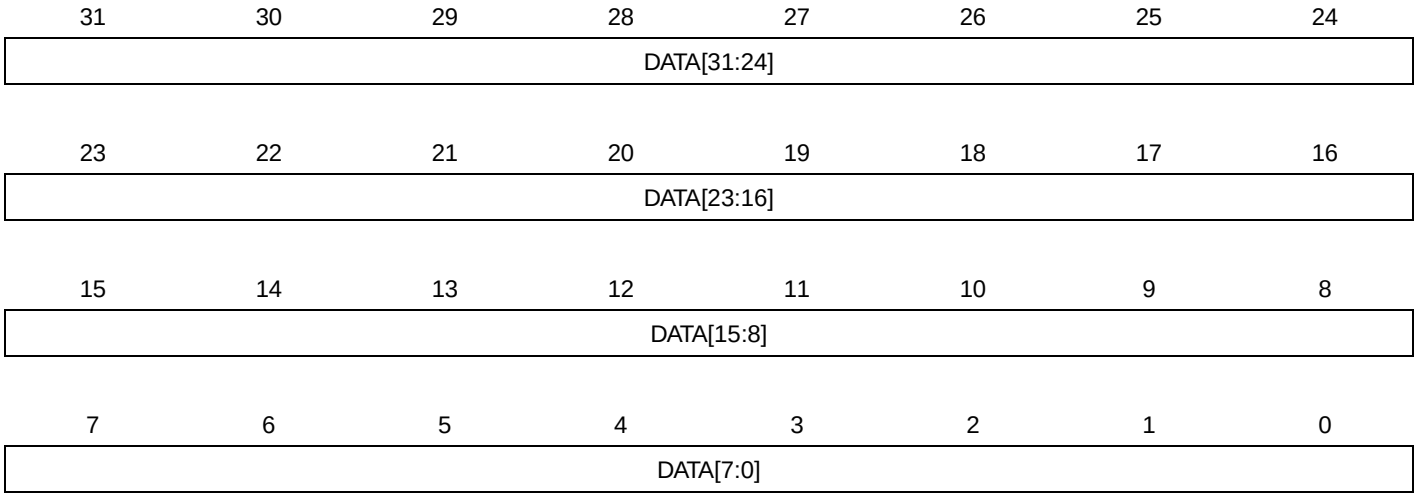
10.7.21 Performance Channel 0 Write Data Cycles

Name: PWDATA0

Access Type: Read-only

Offset: 0x810

Reset Value: 0x00000000



- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

10.7.22 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

10.7.23 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.

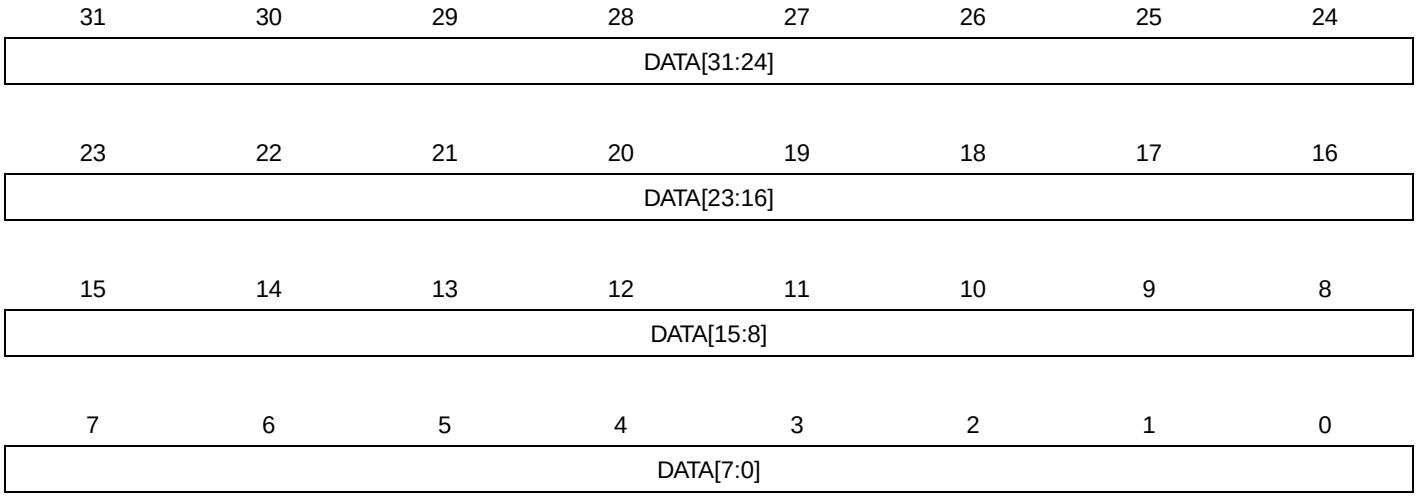
10.7.24 Performance Channel 1 Read Data Cycles

Name: PRDATA1

Access Type: Read-only

Offset: 0x81C

Reset Value: 0x00000000



- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

10.7.25 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

10.7.26 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.

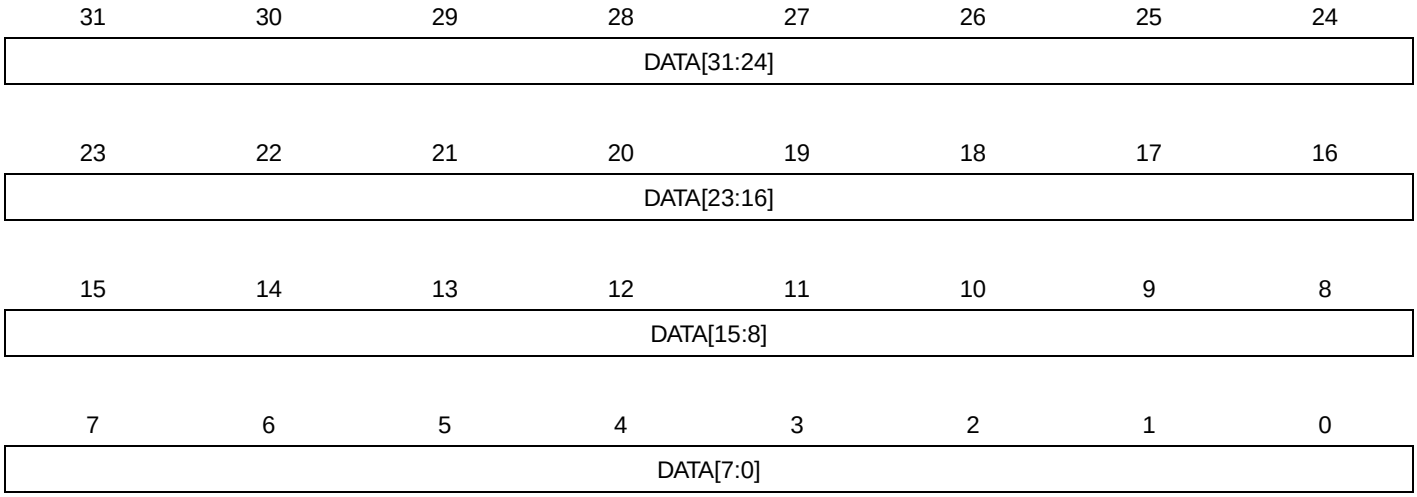
10.7.27 Performance Channel 1 Write Data Cycles

Name: PWDATA1

Access Type: Read-only

Offset: 0x828

Reset Value: 0x00000000



- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

10.7.28 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

10.7.29 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.

10.7.30 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.

10.8 Module Configuration

The specific configuration for each PDCA 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 10-6. PDCA Configuration

Feature	PDCA
Number of channels	12
Number of performance monitors	1

Table 10-7. Module Clock Name

Module name	PB Clock Name	HSB Clock Name
PDCA	CLK_PDCA_PB	CLK_PDCA_HSB

Table 10-8. Register Reset Values

Register	Reset Value
PSR CH 0	0
PSR CH 1	1
PSR CH 2	2
PSR CH 3	3
PSR CH 4	4
PSR CH 5	5
PSR CH 6	6
PSR CH 7	7
PSR CH 8	8
PSR CH 9	9
PSR CH 10	10
PSR CH 11	11
VERSION	122

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 10-9. Peripheral Identity Values

PID	Direction	Peripheral Instance	Peripheral Register
0	RX	USART0	RHR
1	RX	USART1	RHR
2	RX	USART2	RHR
3	RX	USART3	RHR
4	RX	SPI	RDR

Table 10-9. Peripheral Identity Values

PID	Direction	Peripheral Instance	Peripheral Register
5	RX	TWIM0	RHR
6	RX	TWIM1	RHR
7	RX	TWIS0	RHR
8	RX	TWIS1	RHR
9	RX	ADCIFB	LCDR
10	RX	AW	RHR
11	RX	CAT	ACOUNT
12	TX	USART0	THR
13	TX	USART1	THR
14	TX	USART2	THR
15	TX	USART3	THR
16	TX	SPI	TDR
17	TX	TWIM0	THR
18	TX	TWIM1	THR
19	TX	TWIS0	THR
20	TX	TWIS1	THR
21	TX	AW	THR
22	TX	CAT	MBLEN

11. Peripheral Event System

Rev: 1.0.0.1

11.1 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**

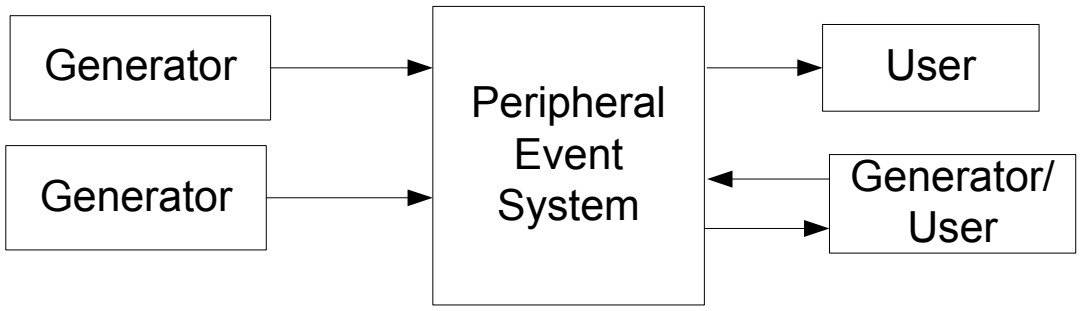
11.2 Overview

Several 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 peripheral events are called peripheral event users and peripherals that emit peripheral events are called peripheral event generators. A single module can be both a peripheral event generator and user.

The peripheral event generators and users are interconnected by a network known as the Peripheral Event System. This allows low latency peripheral-to-peripheral signaling 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.

11.3 Peripheral Event System Block Diagram

Figure 11-1. Peripheral Event System Block Diagram



11.4 Functional Description

11.4.1 Configuration

The Peripheral Event System in the AT32UC3L has a fixed mapping of peripheral events between generators and users, as described in Table 11-1 to Table 11-4. Thus, the user does not need to configure the interconnection between the modules, although each peripheral event can be enabled or disabled at the generator or user side as described in the peripheral chapter for each module.

Table 11-1. Peripheral Event Mapping from ACIFB to PWMA

Generator	Generated Event	User	Effect	Asynchronous
ACIFB channel 0	AC0 V_{INP} > AC0 V_{INN}	PWMA channel 0	PWMA duty cycle value increased by one	No
	AC0 V_{INN} > AC0 V_{INP}		PWMA duty cycle value decreased by one	
ACIFB channel 1	AC1 V_{INP} > AC1 V_{INN}	PWMA channel 6	PWMA duty cycle value increased by one	
	AC1 V_{INN} > AC1 V_{INP}		PWMA duty cycle value decreased by one	
ACIFB channel 2	AC2 V_{INP} > AC2 V_{INN}	PWMA channel 8	PWMA duty cycle value increased by one	
	AC2 V_{INN} > AC2 V_{INP}		PWMA duty cycle value decreased by one	
ACIFB channel 3	AC3 V_{INP} > AC3 V_{INN}	PWMA channel 9	PWMA duty cycle value increased by one	
	AC3 V_{INN} > AC3 V_{INP}		PWMA duty cycle value decreased by one	
ACIFB channel 4	AC4 V_{INP} > AC4 V_{INN}	PWMA channel 11	PWMA duty cycle value increased by one	
	AC4 V_{INN} > AC4 V_{INP}		PWMA duty cycle value decreased by one	
ACIFB channel 5	AC5 V_{INP} > AC5 V_{INN}	PWMA channel 14	PWMA duty cycle value increased by one	
	AC5 V_{INN} > AC5 V_{INP}		PWMA duty cycle value decreased by one	
ACIFB channel 6	AC6 V_{INP} > AC6 V_{INN}	PWMA channel 19	PWMA duty cycle value increased by one	
	AC6 V_{INN} > AC6 V_{INP}		PWMA duty cycle value decreased by one	
ACIFB channel 7	AC7 V_{INP} > AC7 V_{INN}	PWMA channel 20	PWMA duty cycle value increased by one	
	AC7 V_{INN} > AC7 V_{INP}		PWMA duty cycle value decreased by one	
ACIFB channel n	ACn V_{INN} > ACn V_{INP}	CAT	Automatically used by the CAT when performing QMatrix acquisition.	No

Table 11-2. Peripheral Event Mapping from GPIO to TC

Generator	Generated Event	User	Effect	Asynchronous
GPIO	Pin change on PA00-PA07	TC0	A0 capture	No
	Pin change on PA08-PA15		A1 capture	
	Pin change on PA16-PA23		A2 capture	
	Pin change on PB00-PB07	TC1	A1 capture	
	Pin change on PB08-PB15		A2 capture	

Table 11-3. Peripheral Event Mapping from AST

Generator	Generated Event	User	Effect	Asynchronous
AST	Overflow event	ACIFB	Comparison is triggered if the ACIFB.CONFn register is written to 11 (Event Triggered Single Measurement Mode) and the EVENTEN bit in the ACIFB.CTRL register is written to 1.	Yes
	Periodic event			
	Alarm event			
	Overflow event	ADCIFB	Conversion is triggered if the TRGMOD bit in the ADCIFB.TRGR register is written to 111 (Peripheral Event Trigger).	
	Periodic event			
	Alarm event			
	Overflow event	CAT	Trigger one iteration of autonomous touch detection.	
	Periodic event			
	Alarm event			

Table 11-4. Peripheral Event Mapping from PWMA

Generator	Generated Event	User	Effect	Asynchronous
PWMA channel 0	Timebase counter reaches the duty cycle value.	ACIFB	Comparison is triggered if the ACIFB.CONFn register is written to 11 (Event Triggered Single Measurement Mode) and the EVENTEN bit in the ACIFB.CTRL register is written to 1.	No
		ADCIFB	Conversion is triggered if the TRGMOD bit in the ADCIFB.TRGR register is written to 111 (Peripheral Event Trigger).	

11.4.2 Peripheral Event Connections

Each generated peripheral event is connected to one or more users. If a peripheral event is connected to multiple users, the peripheral event can trigger actions in multiple modules.

A peripheral event user can likewise be connected to one or more peripheral event generators. If a peripheral event user is connected to multiple generators, the peripheral events are OR'ed together to a single peripheral event. This means that peripheral events from either one of the generators will result in a peripheral event to the user.

To configure a peripheral event, the peripheral event must be enabled at both the generator and user side. Even if a generator is connected to multiple users, only the users with the peripheral event enabled will trigger on the peripheral event.

11.4.3 Low Power Operation

As the peripheral events do not require CPU intervention, they are available in Idle mode. They are also available in deeper sleep modes if both the generator and user remain clocked in that mode.

Certain events are known as asynchronous peripheral events, as identified in Table 11-1 to Table 11-4. These can be issued even when the system clock is stopped, and revive unclocked user peripherals. The clock will be restarted for this module only, without waking the system from sleep mode. The clock remains active only as long as required by the triggered function, before being switched off again, and the system remains in the original sleep mode. The CPU and sys-

tem will only be woken up if the user peripheral generates an interrupt as a result of the operation. This concept is known as SleepWalking™ and is described in further detail in the Power Manager chapter. Note that asynchronous peripheral events may be associated with a delay due to the need to restart the system clock source if this has been stopped in the sleep mode.

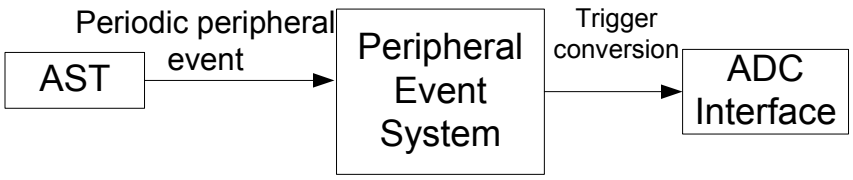
11.5 Application Example

This application example shows how the Peripheral Event System can be used to program the ADC Interface to perform ADC conversions at selected intervals.

Conversions of the active analog channels are started with a software or a hardware trigger. One of the possible hardware triggers is a peripheral event trigger, allowing the Peripheral Event System to synchronize conversion with some configured peripheral event source. From Table 11-3 and Table 11-4, it can be read that this peripheral event source can be either an AST peripheral event, or an event from the PWM Controller. The AST can generate periodic peripheral events at selected intervals, among other types of peripheral events. The Peripheral Event System can then be used to set up the ADC Interface to sample an analog signal at regular intervals.

The user must enable peripheral events in the AST and in the ADC Interface to accomplish this. The periodic peripheral event in the AST is enabled by writing a one to the corresponding bit in the AST Event Enable Register (EVE). To select the peripheral event trigger for the ADC Interface, the user must write the value 0x7 to the Trigger Mode (TRGMOD) field in the ADC Interface Trigger Register (TRGR). When the peripheral events are enabled, the AST will generate peripheral events at the selected intervals, and the Peripheral Event System will route the peripheral events to the ADC Interface, which will perform ADC conversions at the selected intervals.

Figure 11-2. Application Example



Since the AST peripheral event is asynchronous, the description above will also work in sleep modes where the ADC clock is stopped. In this case, the ADC clock (and clock source, if needed) will be restarted during the ADC conversion. After the conversion, the ADC clock and clock source will return to the sleep state, unless the ADC generates an interrupt, which in turn will wake up the system. Using asynchronous interrupts thus allows ADC operation in much lower power states than would otherwise be possible.

12. Interrupt Controller (INTC)

Rev: 1.0.2.5

12.1 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**

12.2 Overview

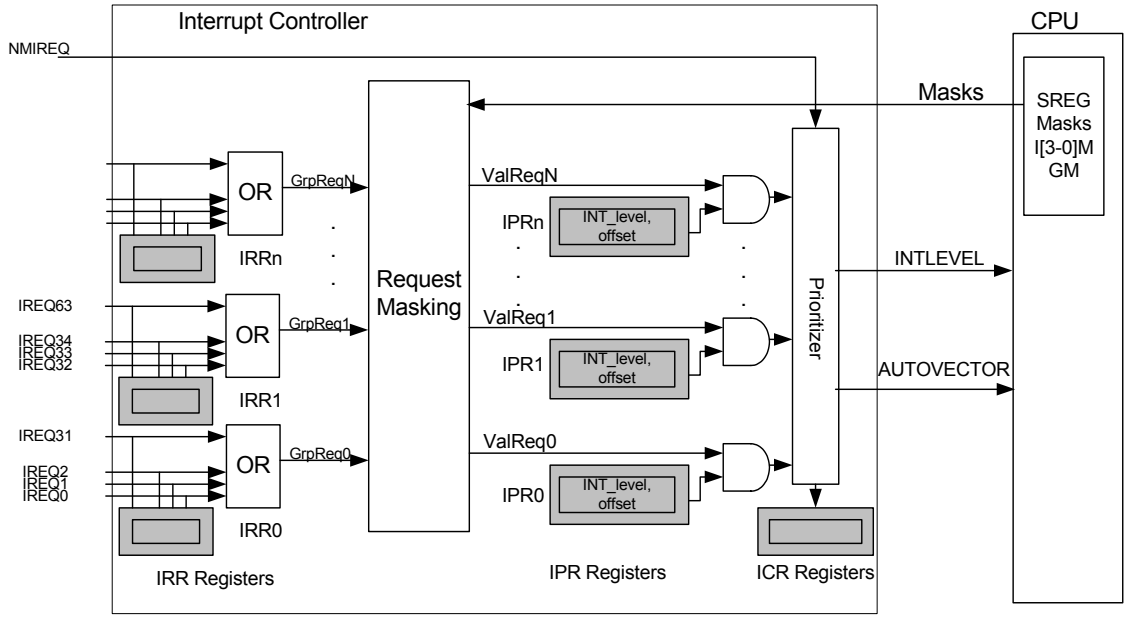
The INTC collects interrupt requests from the peripherals, prioritizes them, and delivers an interrupt request and an autovector to the CPU. The AVR32 architecture supports 4 priority levels for regular, maskable interrupts, and a Non-Maskable Interrupt (NMI).

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.

12.3 Block Diagram

Figure 12-1 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 12-1. INTC Block Diagram



12.4 Product Dependencies

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

12.4.1 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.

12.4.2 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.

12.4.3 Debug Operation

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

12.5 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.

12.5.1 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.

12.5.2 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.

12.5.3 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.

12.6 User Interface

Table 12-1. 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

12.6.1 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.

12.6.2 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.

12.6.3 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.

12.7 Module Configuration

The specific configuration for each INTC 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 12-2. INTC Clock Name

Module Name	Clock Name
INTC	CLK_INTC

12.7.1 Interrupt Request Signal Map

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.

The table below shows how the interrupt request signals are connected to the INTC.

Table 12-3. Interrupt Request Signal Map

Group	Line	Module	Signal
0	0	AVR32 UC3 Core	SYSREG COMPARE
1	0	AVR32 UC3 Core	OCD DCEMU_DIRTY
	1	AVR32 UC3 Core	OCD DCCPU_READ
2	0	Flash Controller	FLASHCDW
3	0	Secure Access Unit	SAU
4	0	Peripheral DMA Controller	PDCA 0
	1	Peripheral DMA Controller	PDCA 1
	2	Peripheral DMA Controller	PDCA 2
	3	Peripheral DMA Controller	PDCA 3
5	0	Peripheral DMA Controller	PDCA 4
	1	Peripheral DMA Controller	PDCA 5
	2	Peripheral DMA Controller	PDCA 6
	3	Peripheral DMA Controller	PDCA 7
6	0	Peripheral DMA Controller	PDCA 8
	1	Peripheral DMA Controller	PDCA 9
	2	Peripheral DMA Controller	PDCA 10
	3	Peripheral DMA Controller	PDCA 11
7	0	Power Manager	PM
8	0	System Control Interface	SCIF
9	0	Asynchronous Timer	AST ALARM

Table 12-3. Interrupt Request Signal Map

10	0	Asynchronous Timer	AST PER
	1	Asynchronous Timer	AST OVF
	2	Asynchronous Timer	AST READY
	3	Asynchronous Timer	AST CLKREADY
11	0	External Interrupt Controller	EIC 1
	1	External Interrupt Controller	EIC 2
	2	External Interrupt Controller	EIC 3
	3	External Interrupt Controller	EIC 4
12	0	External Interrupt Controller	EIC 5
13	0	Frequency Meter	FREQM
14	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
15	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART0
16	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART1
17	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART2
18	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART3
19	0	Serial Peripheral Interface	SPI
20	0	Two-wire Master Interface	TWIM0
21	0	Two-wire Master Interface	TWIM1
22	0	Two-wire Slave Interface	TWIS0
23	0	Two-wire Slave Interface	TWIS1
24	0	Pulse Width Modulation Controller	PWMA
25	0	Timer/Counter	TC00
	1	Timer/Counter	TC01
	2	Timer/Counter	TC02
26	0	Timer/Counter	TC10
	1	Timer/Counter	TC11
	2	Timer/Counter	TC12
27	0	ADC Interface	ADCIFB

Table 12-3. Interrupt Request Signal Map

28	0	Analog Comparator Interface	ACIFB
29	0	Capacitive Touch Module	CAT
30	0	aWire	AW

13. Power Manager (PM)

Rev: 4.1.1.1

13.1 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**
- **Support advanced Shutdown sleep mode**

13.2 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.

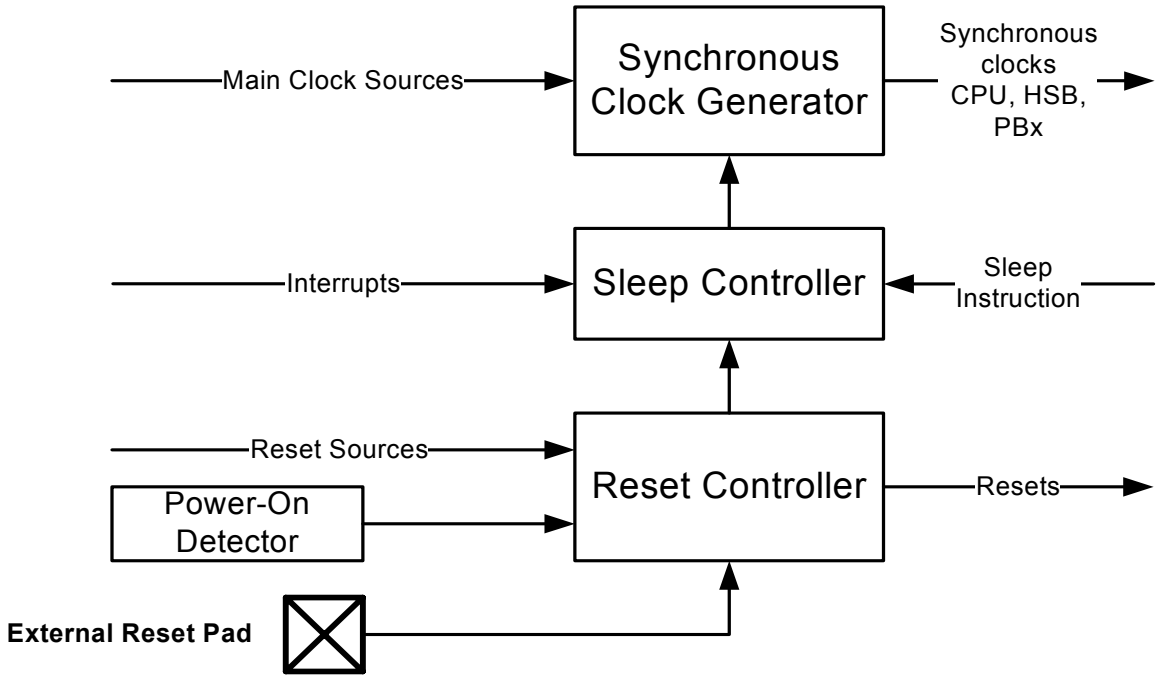
Each module also has a separate clock, enabling the user to switch off the clock for inactive modules, to save further power. Additionally, clocks and oscillators can be automatically switched off during idle periods by using the sleep instruction on the CPU. The system will return to normal operation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

To get maximum power savings, a special sleep mode, called Shutdown is available, where power on all internal logic (CPU, peripherals) and most of the I/O lines is removed, reducing leakage. Only a small amount of logic, including 32KHz crystal oscillator and AST is left powered.

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.

13.3 Block Diagram

Figure 13-1. PM Block Diagram



13.4 I/O Lines Description

Table 13-1. I/O Lines Description

Name	Description	Type	Active Level
RESET_N	Reset	Input	Low

13.5 Product Dependencies

13.5.1 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.

13.5.2 Clock Implementation

In AT32UC3L, 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.

13.5.3 Power Considerations

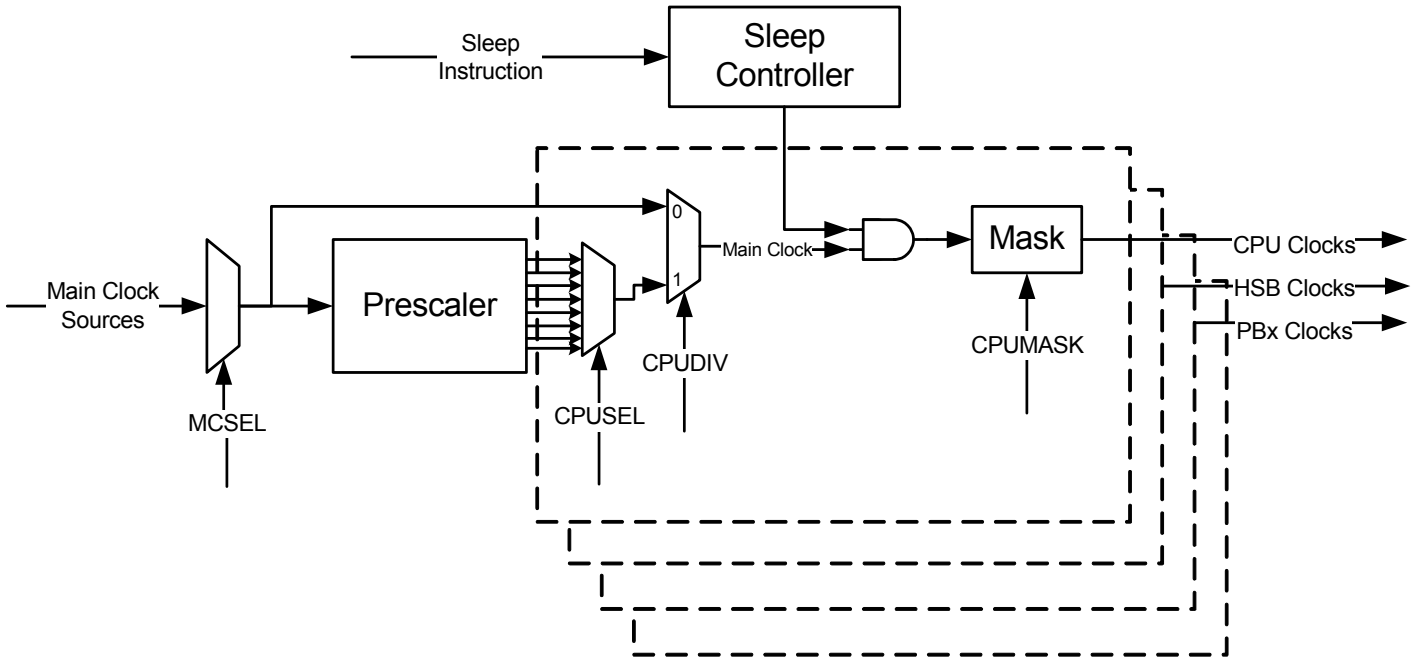
The Shutdown mode is only available for the "**3.3V supply mode, with 1.8V regulated I/O lines**" power configuration.

13.6 Functional Description

13.6.1 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 four 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 13.6.3. Additionally, the clocks for each module in the four domains can be individually masked, to avoid power consumption in inactive modules.

Figure 13-2. Synchronous Clock Generation



13.6.1.1 *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, 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.

13.6.1.2 *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.

13.6.1.3 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.

13.6.2 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.

13.6.2.1 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.

13.6.3 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 13-2 on page 159 as argument.

13.6.3.1 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.

13.6.3.2 Supported sleep modes

The following sleep modes are supported. These are detailed in Table 13-2 on page 159.

- Idle: The CPU is stopped, the rest of the chip is operating. Wake-up sources are any interrupts.
- Frozen: The CPU and HSB modules are stopped, peripherals are operating. Wake-up sources are any interrupts from PB modules.
- Standby: All synchronous clocks are stopped, but the clock sources are running, allowing quick wake-up to normal mode. Wake-up sources are AST, WDT, external interrupts, external reset or any asynchronous interrupts from PB modules.
- Stop: As Standby, but oscillators, and other clock sources are stopped. 32KHz (if enabled), RC oscillators, AST and WDT will still operate. Wake-up sources are the same as for Standby mode.
- 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. Wake-up sources are the same as for Standby mode.
- Static: All clock sources, including RC oscillator are stopped. 32KHz oscillator can run if enabled. Bandgap voltage reference and BOD detector are turned off. Wake-up sources are AST, WDT (if clocked from the 32KHz oscillator), external interrupts, external reset or any asynchronous interrupts from PB modules.
- Shutdown: All clock sources, including RC oscillator are stopped. 32 KHz oscillator can run if enabled. Bandgap voltage reference BOD detector is turned off. Voltage regulator is turned off. Wake-up sources are external reset or external wake-up pin. This mode can only be used in the "**3.3V supply mode, with 1.8V regulated I/O lines**" configuration (described in Power Considerations chapter). See Section 13.6.4 for more details.

Table 13-2. Sleep Modes

Index[1]	Sleep Mode	CPU	HSB	PBA,B 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
6	**Shutdown**	Stop	Stop	Stop	Stop	Run	Stop	Off	Off

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.

13.6.3.3 *SleepWalking™*

In all sleep modes where the PBx clocks are stopped, except for Shutdown mode, the chip can wake partially up if a PBx module asynchronously discovers that it needs its clock. Only the requested clocks and clock sources needed will be started, and all other clocks will be masked to zero. E.g. if the main clock source is OSC0, only OSC0 will be started even if other clock sources were enabled in normal mode. Also generic clocks can be started in a similar way. The state where only requested clocks are running is referred to as SleepWalking.

The time spent to start the requested clock is mostly limited by the startup time of the given clock source. This allows PBx modules to handle incoming requests, while still keeping the power consumption at a minimum.

When the chip is SleepWalking any asynchronous interrupt can wake up the chip at any time without stopping the requested PBx clock.

All requests to start clocks can be masked by writing to the Peripheral Power Control Register (PPCR), all requests are enabled at reset.

During SleepWalking the interrupt controller clock will be running. If an interrupt is pending when entering SleepWalking, this will wake up the whole chip.

13.6.3.4 *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.

The Shutdown sleep mode requires extra care. Please refer to Section 13.6.4.

13.6.4 Shutdown Sleep Mode

13.6.4.1 *Description*

The Shutdown sleep mode is available only when the chip is used in the "**3.3V supply mode, with 1.8V regulated I/O lines**" configuration (see Power Considerations chapter). In this configuration, the voltage regulator supplies both VDDCORE and VDDIO power supplies.

When the device enters Shutdown mode, the regulator is turned off and only the following logic is kept powered by VDDIN:

– 2nd 32KHz crystal oscillator (available on PA13/PA20)

– AST core logic (internal counter and alarm detection logic)
– Backup Registers
– I/O lines PA11, PA13, PA20, PA21, PB04, PB05, PB10
– RESET_N line

The table gives the possible usage of the I/O lines that stay powered during the Shutdown sleep mode. If no special function are used, then the I/O lines will keep its settings before entering the sleep mode

Table 13-3. I/O Lines Usage During Shutdown Mode

Pin	Possible Usage During Shutdown Sleep Mode
PA11	WAKE_N signal (active low wake-up)
PA13	XIN32_2 (2nd 32KHz crystal oscillator)
PA20	XOUT32_2
PA21	
PB04	
PB05	
PB10	
RESET_N	Reset pin

13.6.4.2 Entering Shutdown sleep mode

Before entering the Shutdown sleep mode, a few actions are required:

– All modules should normally be disabled before entering Shutdown sleep mode (see Section 13.6.3.4)
– The POR33 (see System Control Interface “SCIF” chapter) must be masked to avoid any spurious reset when the power is back. This is done by writing a one to the POR33MASK bit of the SCIF.VREGCR register. Because of internal synchronisation, this bit nust be read as a one before the sleep instruction is executed by the CPU.

.

As soon as the Shutdown sleep mode is entered, all CPU and peripherals are reset to ensure a consistent state.

POR33 and RC32 oscillator are automatically disabled when entering the Shutdown sleep mode to save extra power

13.6.4.3 *Leaving Shutdown sleep mode*

Exiting the Shutdown sleep mode can be done using the events described in Table 13-4 on page 162.

Table 13-4. Events That Can Wake-Up The Device From Shutdown Mode

Source	How
PA11 (WAKE_N)	Pulling-down PA11 will wake-up the device
RESET_N	Pulling-down RESET_N pin will wake-up the device The device is kept under reset until RESET_N is tied high again
AST	32KHz Crystal oscillator must be set-up to use alternate pinout (XIN32_2 and XOUT32_2) See SCIF Chapter AST must be configured to use the clock from the 32KHz crystal oscillator AST must be configured to allow alarm,periodic or overflow wake-up

When a wake-up event occurs, the regulator is turned-on again and the device will wait for VDDCORE power to be valid again before starting again.

The bit SLEEP is then set in the RCAUSE register. This allows software running on the device to distinguish between the first power-up and a wake-up from Shutdown mode.

13.6.4.4 *Special consideration regarding waking-up from Shutdown sleep mode using the WAKE_N pin*

By default, WAKE_N pin can normally be used to wake-up the device from Shutdown mode only after Shutdown mode has been entered. If the WAKE_N is pulled low before the Shutdown mode is entered, the device will not wake-up from the Shutdown sleep mode.

To allow WAKE_N pin to wake-up the device even if the Shutdown sleep mode is still not entered, the bit WAKEN in the Asynchronous Wake Up Enable register (WAKEN.AWEN) must be write to one. If this bit is set, CPU execution will continue after the sleep instruction if the WAKE_N pin was driven low before the Shutdown sleep mode is entered. The RCAUSE register content will not be changed.

13.6.5 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.

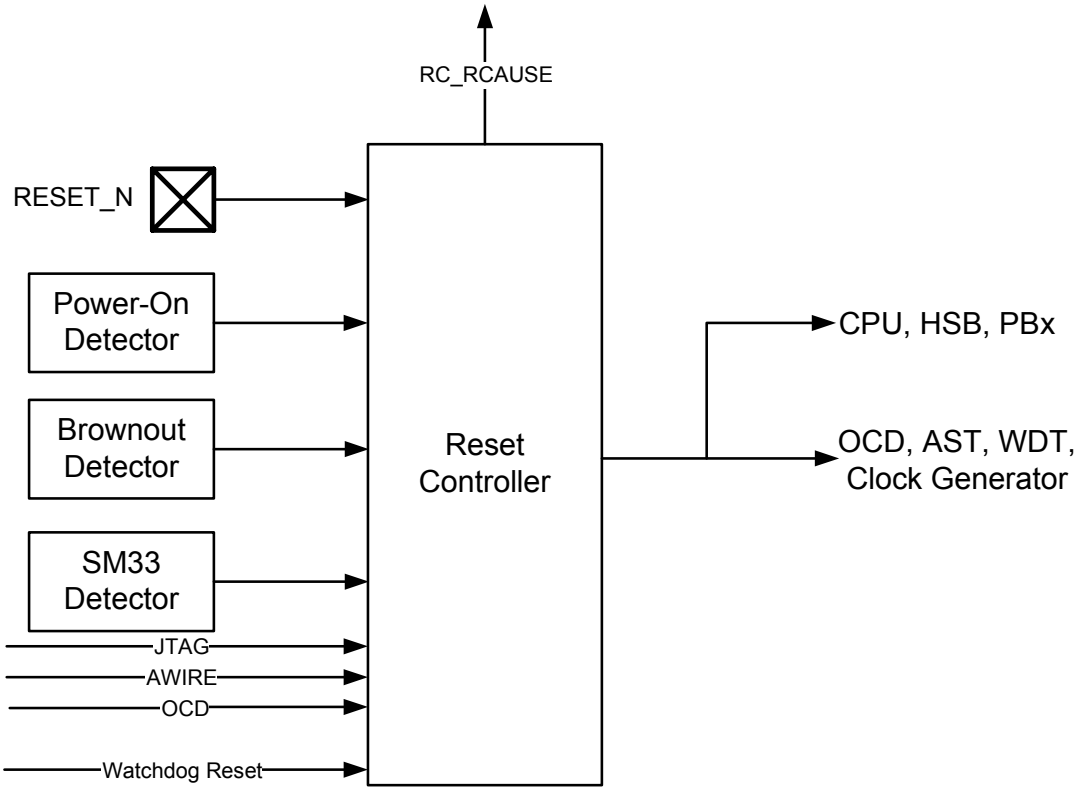
13.6.6 Reset Controller

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 13-5 on page 163 lists these and other reset sources supported by the Reset Controller.

Figure 13-3. 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 13-5. 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
SM33 Reset	Supply voltage on VDDCORE below the brownout reset detector threshold voltage
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.

13.6.6.1 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.

13.6.6.2 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.

13.6.7 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 cyrstal 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.

13.6.8 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.

13.7 User Interface

Table 13-6. PM Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Main Clock Control	MCCTRL	Read/Write	0x00000000
0x004	CPU Clock Select	CPUSEL	Read/Write	0x00000000
0x008	HSB Clock Select	HSBSEL	Read-only	0x00000000
0x00C	PBA Clock Select	PBASEL	Read/Write	0x00000000
0x010	PBB Clock Select	PBBSEL	Read/Write	0x00000000
0x014 - 0x01C	Reserved			
0x020	CPU Mask	CPUMASK	Read/Write	0x00000003
0x024	HSB Mask	HSBMASK	Read/Write	0x0000007F
0x028	PBA Mask	PBAMASK	Read/Write	0x03FFFFFF
0x02C	PBB Mask	PBBMASK	Read/Write	0x00000007
0x030- 0x03C	Reserved			
0x040	PBA Divided Mask	PBADIVMASK	Read/Write	0x0000007F
0x044 - 0x050	Reserved			
0x054	Clock Failure Detector Control	CFDCTRL	Read/Write	0x00000000
0x058	Unlock Register	UNLOCK	Write-only	0x00000000
0x05C - 0x0BC	Reserved			
0x0C0	PM Interrupt Enable Register	IER	Write-only	0x00000000
0x0C4	PM Interrupt Disable Register	IDR	Write-only	0x00000000
0x0C8	PM Interrupt Mask Register	IMR	Read-only	0x00000000
0x0CC	PM Interrupt Status Register	ISR	Read-only	0x00000000
0x0D0	PM Interrupt Clear Register	ICR	Write-only	0x00000000
0x0D4	Status Register	SR	Read-only	0x00000000
0x0D8 - 0x15C	Reserved			
0x160	Peripheral Power Control Register	PPCR	Read/Write	0x00000002
0x164 - 0x17C	Reserved			
0x180	Reset Cause Register	RCAUSE	Read-only	-(2)
0x184	Wake Cause Register	WCAUSE	Read-only	-(3)
0x188	Asyncronous Wake Enable	AWEN	Read/Write	0x00000000
0x18C - 0x3F4	Reserved			
0x3F8	Configuration Register	CONFIG	Read-only	0x00000043
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.
2. Latest Reset Source.
3. Latest Wake Source.

13.7.1 Main Clock Control

Name: MCCTRL

Access Type: Read/Write

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
-	-	-	-	-	MCSEL		

- **MCSEL: Main Clock Select**

Table 13-7. Main clocks in AT32UC3L.

MCSEL[2:0]	Main clock source
0	System RC oscillator (RCSYS)
1	Oscillator0 (OSC0)
2	DFLL
3	120 MHz RC oscillator (RC120M)(1)

Note: 1. If the 120 MHz RC oscillator is selected as main clock source, it must be divided by at least 4 before being used as clock source for the CPU. This division is selected by writing to the CPUSEL and CPUDIV bits in the CPUSEL register, before switching to RC120M as main clock source.

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

13.7.2 CPU Clock Select

Name: CPUSEL

Access Type: Read/Write

Offset: 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
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 refere to the UNLOCK register description for details.

13.7.3 HSB Clock Select

Name: HSBSEL

Access Type: Read

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
HSBDIV	-	-	-	-	HSBSEL		

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

13.7.4 PBx Clock Select

Name: PBxSEL

Access Type: Read/Write

Offset: 0x008-0x00C

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
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 SR.CKRDY. The register must not be re-written until SR.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 refere to the UNLOCK register description for details.

13.7.5 Clock Mask

Name: CPUMASK/HSBMASK/PBAMASK/PBBMASK

Access Type: Read/Write

Offset: 0x020-0x02C

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 13-8.

Table 13-8. Maskable Module Clocks in AT32UC3L.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK
0	OCD	PDCA	PDCA	FLASHCDW
1	-	FLASHCDW	INTC	HMATRIX
2	-	SAU	PM	SAU
3	-	PBB bridge	SCIF	-
4	-	PBA bridge	AST	-
5	-	Peripheral Event System	WDT	-
6	-	-	EIC	-
7	-	-	FREQM	-
8	-	-	GPIO	-
9	-	-	USART0	-
10	-	-	USART1	-
11	-	-	USART2	-

Table 13-8. Maskable Module Clocks in AT32UC3L.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK
12	-	-	USART3	-
13	-	-	SPI	-
14	-	-	TWIM0	-
15	-	-	TWIM1	-
16	-	-	TWIS0	-
17	-	-	TWIS1	-
18	-	-	PWMA	-
19	-	-	TC0	-
20	-	-	TC1	-
21	-	-	ADCIFB	-
22	-	-	ACIFB	-
23	-	-	CAT	-
24	-	-	GLOC	-
25	-	-	AW	-
31:26	-	-	-	-

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

13.7.6 Divided Clock Mask

Name: PBADIVMASK

Access Type: Read/Write

Offset: 0x040

Reset Value: 0x0000007F

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

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 13-9 shows what clocks are affected by the different MASK bits.

Table 13-9. Divided Clock Mask

Bit	USART0	USART1	USART2	USART3	TC0	TC1
0	-	-	-	-	TIMER_CLOCK2	TIMER_CLOCK2
1	-	-	-	-	-	-
2	CLK_USART/ DIV	CLK_USART/ DIV	CLK_USART/ DIV	CLK_USART/ DIV	TIMER_CLOCK3	TIMER_CLOCK3
3	-	-	-	-	-	-
4	-	-	-	-	TIMER_CLOCK4	TIMER_CLOCK4
5	-	-	-	-	-	-
6	-	-	-	-	TIMER_CLOCK5	TIMER_CLOCK5

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

13.7.7 Clock Failure Detector Control Register

Name: CFDCTRL

Access Type: Read/Write

Offset: 0x054

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 refere to the UNLOCK register description for details.

13.7.8 PM Unlock Register

Name: UNLOCK

Access Type: Write-Only

Offset: 0x058

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.

13.7.9 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	-	-	-	-	-

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.

13.7.10 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	-	-	-	-	-

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.

13.7.11 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	-	-	-	-	-

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.

13.7.12 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	-	-	-	-	-

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.

13.7.13 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	-	-	-	-	-

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.

13.7.14 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 corretly.
  1: Failure on main clock detected. Main clock is now running on RC osc.

13.7.15 Peripheral Power Control Register

Name: PPCR

Access Type: Read/Write

Offset: 0x004

Reset Value: 0x00000000

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

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

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

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

Table 13-10. Peripheral Power Control

Bit	Name
0	RSTPUN
1	FRC32
2	RSTTM
3	CATRCMASK
4	ACIFBCRCMASK
5	ADCIFBRCMASK
6	ASTRCMASK
7	TWIS0RCMASK
8	TWIS1RCMASK
31:9	-

- **RSTTM : Reset test mode**
  0: External reset not in test mode
  1: External reset in test mode
- **FRC32 : Force RC32 out**
  0: RC32 signal is not forced as output
  1: RC32 signal is forced as output
- **RSTPUN: Reset Pullup, active low**
  0: Pull-up for external reset on
  1: Pull-up for external reset off

- **CATRCMASK : CAT Request Clock Mask**
  0: CAT Request Clock is disabled
  1: CAT Request Clock is enabled
- **ACIFBRCMASK : ACIFB Request Clock Mask**
  0: ACIFB Request Clock is disabled
  1: ACIFB Request Clock is enabled
- **ADCIFBRCMASK : ADCIFB Request Clock Mask**
  0: ADCIFB Request Clock is disabled
  1: ADCIFB Request Clock is enabled
- **ASTRCMASK : AST Request Clock Mask**
  0: AST Request Clock is disabled
  1: AST Request Clock is enabled
- **TWIS0RCMASK : TWIS0 Request Clock Mask**
  0: TWIS0 Request Clock is disabled
  1: TWIS0 Request Clock is enabled
- **TWIS1RCMASK : TWIS1 Request Clock Mask**
  0: TWIS1 Request Clock is disabled
  1: TWIS1 Request Clock 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 refere to the UNLOCK register description for details.

13.7.16 Reset Cause

Name: RCAUSE

Access Type: Read-only

Offset: 0x180

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
-	-	-	AWIRE	-		JTAG	OCDRST

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

- **AWIRE: AWIRE Reset**
  The CPU was reset by the AWIRE
- **JTAG: JTAG Reset**
  The chip was reset by the JTAG system reset.
- **OCDRST: OCD Reset**
  The CPU was reset because the RES strobe in the OCD Development Control register has been written to one.
- **SLEEP: SLEEP reset**
  The CPU was reset because it woke up from Shutdown sleep mode.
- **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.

13.7.17 Wake Cause Register

Name: WCAUSE

Access Type: Read-only

Offset: 0x184

Reset Value: Lateset Wake Source

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

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

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 13-11 on page 184.

Table 13-11. Wake Cause

Bit	Wake Cause
0	CAT
1	ACIFB
2	ADCIFB
3	TWI Slave 0
4	TWI Slave 1
5	-
6	-
7	-
8	-
9	-
10	-
11	-
12	-
13	-
14	-
15	-

Table 13-11. Wake Cause

Bit	Wake Cause
16	EIC
17	AST
31:18	-

13.7.18 Asynchronous Wake Up Enable Register

Name: AWEN

Access Type: Read/Write

Offset: 0x188

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 13-12 on page 186.
0: The correcponding wake up is disabled.
1: The corresponding wake up is enabled

Table 13-12. Asynchronous Wake Up

Bit	Asynchronous Wake Up
0	CATWEN
1	ACIFBWEN
2	ADCIFBWEN
3	TWIS0WEN
4	TWIS1WEN
31:5	-

13.7.19 Configuration Register

Name: CONFIG

Access Type: Read-Only

Offset: 0x3F8

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
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.

13.7.20 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.

13.8 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 “Synchronous Clocks”, “Peripheral Clock Masking” and “Sleep Modes” sections for details.

Table 13-13. Power Manager Clocks

Clock Name	Description
CLK_PM	Clock for the PM bus interface

Table 13-14. Register Reset Values

Register	Reset Value
VERSION	0x00000411

Table 13-15. Effect of the Different Reset Events

	Power-On Reset	External Reset	Watchdog Reset	BOD Reset	SM33 Reset	CPU Error Reset	OCD Reset	JTAG Reset
CPU/HSB/PBx (excluding Power Manager)	Y	Y	Y	Y	Y	Y	Y	Y
32KHz oscillator	Y	N	N	N	N	N	N	N
AST control register	Y	N	N	N	N	N	N	N
Watchdog control register	Y	Y	N	Y	Y	Y	Y	Y
Voltage Calibration register	Y	N	N	N	N	N	N	N
RC Oscillator Calibration register	Y	N	N	N	N	N	N	N
SM33 control register	Y	Y	Y	Y	Y	Y	Y	Y
BOD control register	Y	Y	Y	N	Y	Y	Y	Y
Bandgap control register	Y	Y	Y	N	Y	Y	Y	Y
Clock control registers	Y	Y	Y	Y	Y	Y	Y	Y
OSC control registers	Y	Y	Y	Y	Y	Y	Y	Y
DFLL control registers	Y	Y	Y	Y	Y	Y	Y	Y
OCD system and OCD registers	Y	Y	N	Y	Y	Y	N	Y

14. System Control Interface (SCIF)

Rev: 1.0.2.2

14.1 Features

- **Controls integrated oscillators and Digital Frequency Locked Loop (DFLL)**
- **Supports crystal oscillator 3-16 MHz (OSC0)**
- **Supports Digital Frequency Locked Loop 40-150 MHz (DFLL)**
- **Supports 32 KHz ultra-low power oscillator (OSC32K)**
- **Supports 32 kHz RC oscillator (RC32K)**
- **Integrated low-power RC oscillator (RCSYS)**
- **Generic clocks (GCLK) with wide frequency range provided**
- **Controls bandgap voltage reference through control and calibration registers**
- **Controls Brown-out detectors (BOD) and supply monitors**
- **Controls Voltage Regulator (VREG) behavior and calibration**
- **Controls Temperature Sensor**
- **Controls Supply Monitor 33 (SM33) operating modes and calibration**
- **Controls 120 MHz integrated RC Oscillator (RC120M)**
- **Four 32 bit general purpose backup registers**

14.2 Overview

The System Control Interface (SCIF) controls the Oscillators, DFLL, Generic Clocks, BODs, Temperature Sensor, VREG, and backup registers.

14.3 I/O Lines Description

Table 14-1. I/O Lines Description

Pin Name	Pin Description	Type
RC32OUT	RC32 output at startup	Output
XIN0	Crystal 0 Input	Analog/Digital
XIN32	Crystal 32 Input (primary location)	Analog/Digital
XIN32_2	Crystal 32 Input (secondary location)	Analog/Digital
XOUT0	Crystal 0 Output	Analog
XOUT32	Crystal 32 Output (primary location)	Analog
XOUT32_2	Crystal 32 Output (secondary location)	Analog
GCLK4-GCLK0	Generic Clock Output	Output

14.4 Product Dependencies

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

14.4.1 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. Oscillator pins are also multiplexed

with GPIO. When oscillators are used, the related pins are controlled directly by the SCIF, overriding GPIO settings.

14.4.2 Interrupt

The SCIF interrupt request line is connected to the interrupt controller. Using the SCIF interrupt requires the interrupt controller to be programmed first.

14.4.3 Debug Operation

The SCIF module does not interact with debug operations.

14.4.4 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.)

14.5 Functional Description

14.5.1 Oscillator (OSC0) operation

The main oscillator (OSC0) is designed to be used with an external 3 to 16 MHz crystal and two biasing capacitors, as shown in Figure 14-1. Capacitor values can be found in the Electrical Characteristics chapter. 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 Section 14.5.4 on page 197.

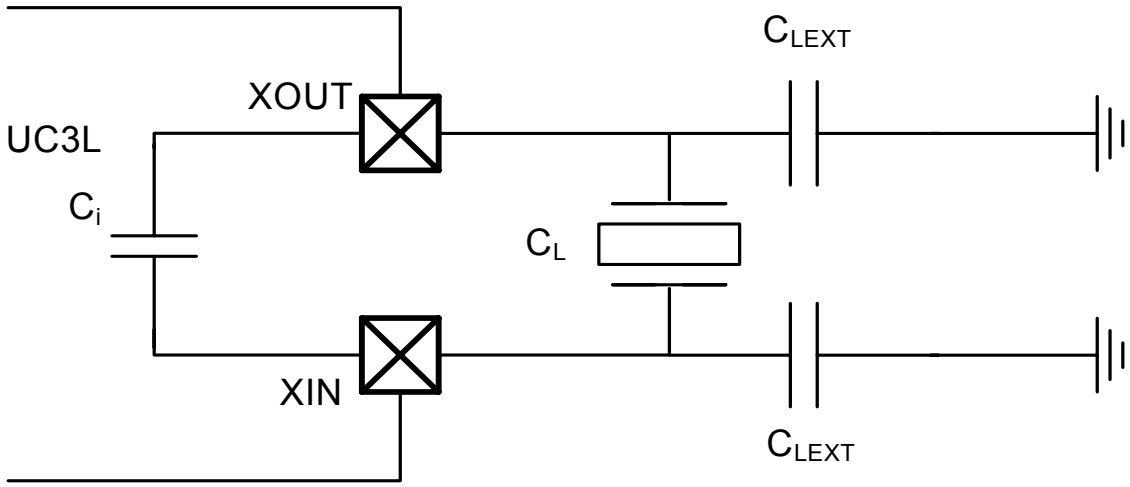
The oscillator is disabled by default after reset. When the oscillator is disabled, the XIN and XOUT pins can be used as general purpose I/Os. When the oscillator is configured to use an external clock, the clock must be applied to the XIN pin while the XOUT pin can be used as a general purpose I/O.

The oscillator can be enabled by writing to the OSCEN bits in OSCCTRL0. Operation mode (external clock or crystal) is chosen by writing to the MODE field in OSCCTRL0. 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 OSCCTRL0 register.

The SCIF masks the oscillator outputs during the start-up time, to ensure that no unstable clocks propagate to the digital logic. The OSC0RDY 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 Section 14.6.1.

Figure 14-1. Oscillator Connection



14.5.2 32KHz Oscillator (OSC32K) Operation

The 32KHz oscillator operates as described for the oscillator above. Figure 14-1 also applies to this oscillator. 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 197.

The oscillator is disabled by default, but can be enabled by writing OSC32EN 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 also has a 1KHz output. This is enabled by writing to EN1K bit in OSCCTRL32 register. If the 32KHz output clock is not needed when 1K is enabled, this can be disabled by writing zero to EN32K in OSCCTRL32 register. EN32K is set to one after reset.

The 32KHz oscillator has two possible set of pins. To select between them write to the PINSEL bit in OSCCTRL32 register.

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

14.5.3 DFLL Operation

The device contains one Digital Frequency Locked Loop (DFLL). This is disabled by default, but can be enabled to provide a high frequency source clock for synchronous and generic clocks.

Features:

- Internal oscillator with no external components
- 40-150MHz output frequency

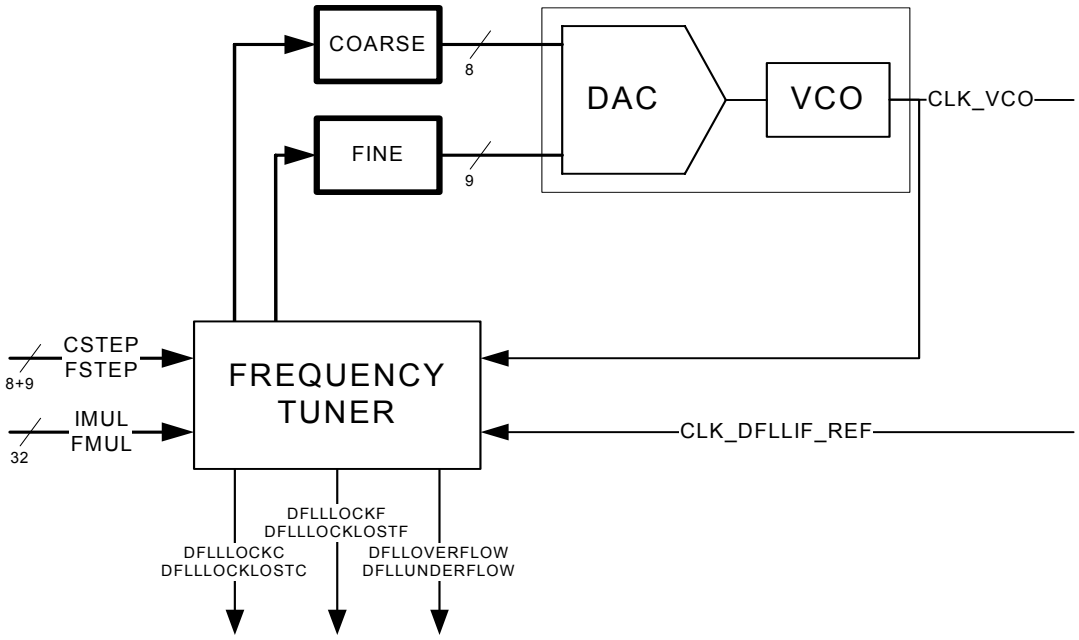
- Can operate standalone as high frequency programmable oscillator in open loop mode
- Can operate as accurate frequency multiplier against a known frequency in closed loop mode
- Optional spread-spectrum clock generation
- Very high frequency multiplication supported - can generate all frequencies from 32kHz

The DFLL provides a high frequency clock. The DFLL can operate in both open loop mode and closed loop mode. In closed loop mode a low frequency clock with high accuracy can be used as reference clock to get high accuracy on the output frequency.

The output frequency is adjusted by changing the values in the COARSE and FINE fields in the DFLL0 Configuration Register (DFLL0CONF.COARSE and DFLL0CONF.FINE). In open loop mode, the user must find the values of COARSE and FINE to get the correct output frequency. the Frequency Meter can be used to measure the output frequency until the desired output frequency is reached. In closed loop mode, the COARSE and FINE values are controlled by the DFLL Interface to meet a user specified frequency. Open loop operation and closed loop operation are described below.

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

Figure 14-2. DFLLIF Block Diagram



14.5.3.1 *Open Loop Operation*

When operating in open loop mode the output frequency of the DFLL depends on the values written to the COARSE and the FINE fields in the DFLL0CONF register. Take care when setting the value of the COARSE and the FINE fields, to make sure the output frequency does not exceed the maximum frequency of the device after the division in the clock generator. The open loop operation is selected by writing a one to the EN bit in the DFLL0CONF register, and then writing a zero to the MODE bit in the DFLL0CONF register. It is possible to change the value of

COARSE and FINE fields, and thereby the output frequency of the DFLL, while the DFLL is enabled and in use.

The DFLL clock is ready to be used when PCLKSR.DFLL0RDY is cleared after enabling the DFLL.

14.5.3.2 Closed Loop Operation

To select closed loop operation, the user must write a one to the DFLL0CONF.MODE bit and the DFLL0CONF.EN bits. The output frequency of the DFLL is given by:

$$f_{vco} = \left(IMUL + \frac{FMUL}{2^{16}}\right) f_{ref}$$

The COARSE and FINE fields in DFLL0CONF register are read-only in closed loop mode, and are controlled by the DFLLIF to meet user specified frequency. The values in the COARSE register when the closed loop mode is enabled is used by the frequency tuner as a starting point for the COARSE value. Setting the COARSE to a value believed to be the correct will reduce the time needed to get a lock on the coarse value. To set up the DFLLIF first enable the DFLL by writing one to EN bit in DFLL0CONF register. Then enable and select a reference clock (CLK_DFLLIF_REF). CLK_DFLLIF_REF is a generic clock, please refer to Generic Clocks chapter for details. Then set the maximum step size allowed in finding the COARSE and FINE values by setting the CSTEP and FSTEP bits in DFLL0STEP register. A small step size will ensure low overshoot on the output frequency, but will typically be slower. A high value might give a big overshoot, but will typically give faster locking. DFLL0STEP.CSTEP and DFLL0STEP.FSTEP should be set lower than 50% of the maximum value of DFLL0CONF.COARSE and DFLL0CONF.FINE respectively. Then set the value of IMUL and FMUL fields in the DFLL0MUL register, care must be taken when choosing IMUL and FMUL so the output frequency does not exceed the maximum frequency of the device.

The locking of the frequency in closed loop mode is divided into three stages. In the COARSE stage the control logic quickly finds the correct value for the COARSE field in DFLL0CONF register and thereby setting the output frequency to a value close to the correct frequency. The DFLL0LOCKC interrupt is issued when this is done. In the FINE stage the control logic tunes the value in the FINE field in the DFLL0CONF register so the output frequency very close to the desired frequency. The DFLL0LOCKF interrupt is issued when this is done. In the ACCURATE stage the DFLL frequency tuning mechanism uses dithering on the FINE bits to obtain an accurate output frequency. When the accurate frequency is obtained the DFLL0LOCKA interrupt is issued. The ACCURATE stage will only be executed if DITHER bit in DFLL0CONF register has been written to one. If DITHER is written to zero DFLL0LOCKA will never occur. If dithering is enabled, the frequency of the dithering is decided by a generic clock (CLK_DFLLIF_DITHER). This clock has to be set up correctly before enabling dithering. Please refer to the Generic Clocks chapter for details. The flow for finding the correct settings is shown in Figure 14-3 on page 195.

When dithering is enable the accuracy of the average output frequency of the DFLL will be higher. However, the frequency will be alternating between two frequencies. If a fixed frequency is required, the dithering should not be enabled.

The DFLL clock is ready to be used when PCLKSR.DFLL0RDY is cleared after enabling the DFLL. However, the accuracy of the outputed frequency depends on which locks that are set.

The frequency tuner will automatically compensate for drift in the output frequency of the VCO without losing either of the locks. If the FINE register overflows or underflows, which should nor-

mally not happen, but could occur due to large drift in temperature and voltage, all locks will be lost, and the COARSE and FINE values will be recalibrated as described earlier. When spread spectrum is enabled and the AMPLITUDE is high, an overflow/underflow is more likely to occur.

If the reference clock stops or the reference clock is too slow, the DFLL0RCS interrupt will be asserted. Note that the detection of the clock stop will take long time. The DFLL will enter open loop mode if it detects that the reference clock has stopped.

The ratio between the reference clock and the VCO clock is measured automatically by the DFLLIF. The difference between this ratio and DFLL0MUL is stored in the Multiplication Ratio Difference field (RATIODIFF) in the DFLL0RATIO register. To get the result on the same form as DFLL0MUL, the error must be calculated as follows:

$$error = \frac{RATIODIFF \cdot f_{ref}}{2^{NUMREF} \cdot f_{vco}}$$

where 2^{NUMREF} is the number of reference clock cycles the DFLLIF is using for calculating the ratio. The DFLL0RATIO register is only updated every time the Synchronization (SYNC) bit in DFLL0SYNC is written to one; refer to DFLL0SYNC register description for details.

Figure 14-3. DFLL Closed Loop State Diagram

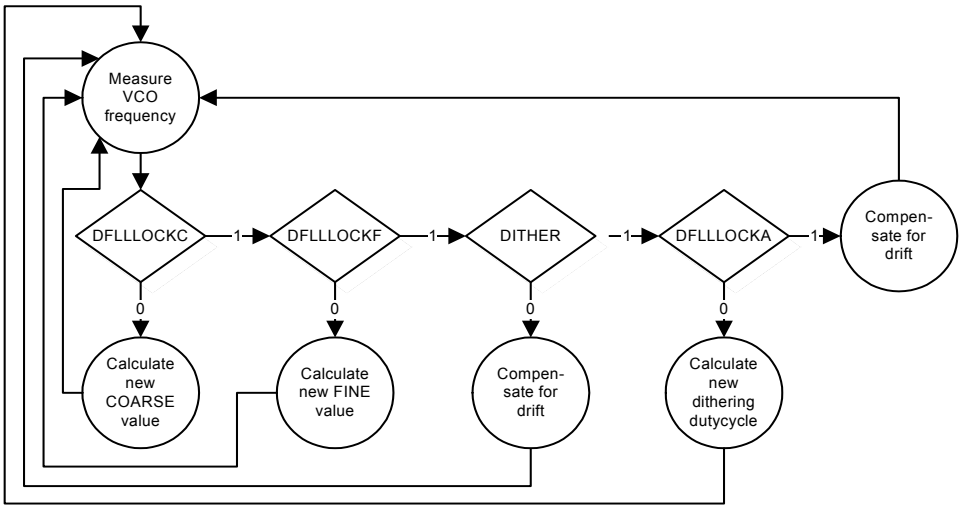
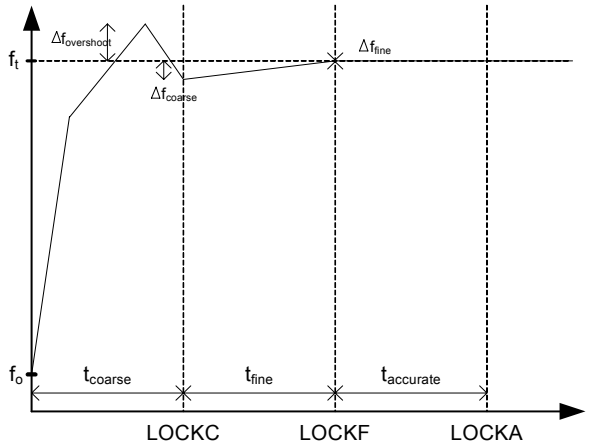


Figure 14-4. DFLL Locking in Closed Loop



14.5.3.3 Enabling the DFLL

Before configuring the DFLL, only the EN bit in the DFLL0CONF register should be written to one. Do not write to any other bits in DFLL0CONF register. The DFLL is now ready for configuration. When PCLKSR.DFLL0RDY is cleared the DFLL clock is ready to be used, if an accurate clock is required, the clock is not ready to be used before DFLL0LOCKF or DFLL0LOCKA is asserted.

14.5.3.4 Disabling the DFLL

Writing a zero to DFLL0CONF.EN disables the DFLL. Do not write to any other bits in DFLL0CONF register when disabling the DFLL. After disabling the DFLL the PCLKSR.DFLL0RDY bit will not be cleared.

14.5.3.5 Re-enabling the DFLL

After disabling the DFLL the PCLKSR.DFLL0RDY bit will not be cleared. If the DFLL is to be re-enabled after disabling it, do not wait for PCLKSR.DFLL0RDY to be set to one. Enable the DFLL by writing the DFLL0CONF.EN bit to one, and do not change any of the other values in the DFLL0CONF register.

14.5.3.6 Spread Spectrum Generator (SSG)

When the DFLL is used as the main clock source for the chip, the EMI radiated from the chip will be synchronous to the VCO frequency. To provide better EMC capabilities the DFLL can provide a clock with the energy spread in the frequency domain. This is done by adding or subtracting values from the FINE value. Enable the SSG by writing a one to the EN bit in the DFLL0SSG register.

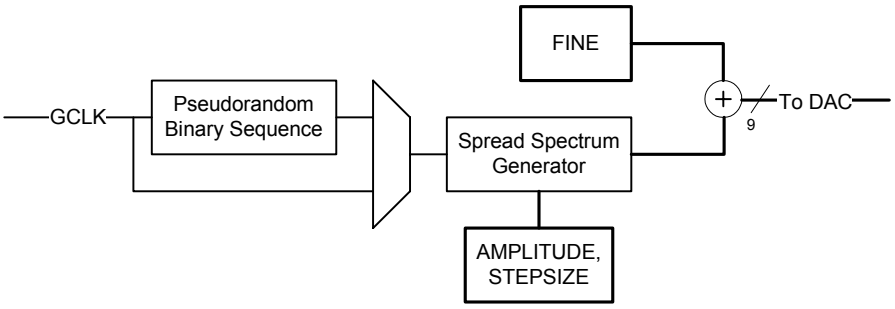
A generic clock sets the rate at which the SSG changes the frequency of the DFLL clock to generate a spread spectrum (CLK_DFLLIF_DITHER). This is the same clock used by the dithering mechanism. The frequency of this clock should be higher than CLK_DFLLIF_REF to ensure that the DFLLIF can lock. Please refer to the Generic clocks section for details.

Optionally, the clock ticks can be qualified by a pseudorandom binary sequence (PRBS) if the PRBS bit in the DFLL0SSG register is set. This reduces the modulation effect of CLK_DFLLIF_DITHER frequency onto f_{vco}.

The amplitude of the frequency variation can be selected by setting the AMPLITUDE field in DFLL0SSG register. If the AMPLITUDE is set to zero the SSG will toggle on the LSB of the FINE value, setting AMPLITUDE to one the SSG will add the sequence {1,-1, 0} to the FINE value on every source clock cycle.

The step size of the SSG is set in the STEPSIZE bit field in the DFLL0SSG register. Setting the STEPSIZE to zero or one will result in a step size equal to one. If the step size is set to *n* the output value from the SSG will be incremented/decremented by *n* on every tick of the source clock.

Figure 14-5. Spread Spectrum Generator Block Diagram.



The Spread Spectrum Generator can operate both in open and closed loop mode.

14.5.3.7 Start-up

The DFLL has very short start-up time. When waking up from a sleep mode where the DFLL has been turned off, and the DFLL clock was the main clock before going to sleep, the DFLL will be re-enabled and start running with the same configuration as before going to sleep even if the reference clock is not available. The locks will not be lost. When the reference clock has restarted, the FINE tracking will quickly compensate for any drift in frequency during sleep.

14.5.3.8 Accuracy

There are mainly three factors that decides the accuracy of the VCO frequency:

- FINE resolution: The frequency step between two FINE values. Refer to the Electrical Characteristics chapter.
- Reference frequency: The reference frequency should be below 100 kHz for optimal accuracy
- The accuracy of the reference clock.

14.5.3.9 Internal synchronization

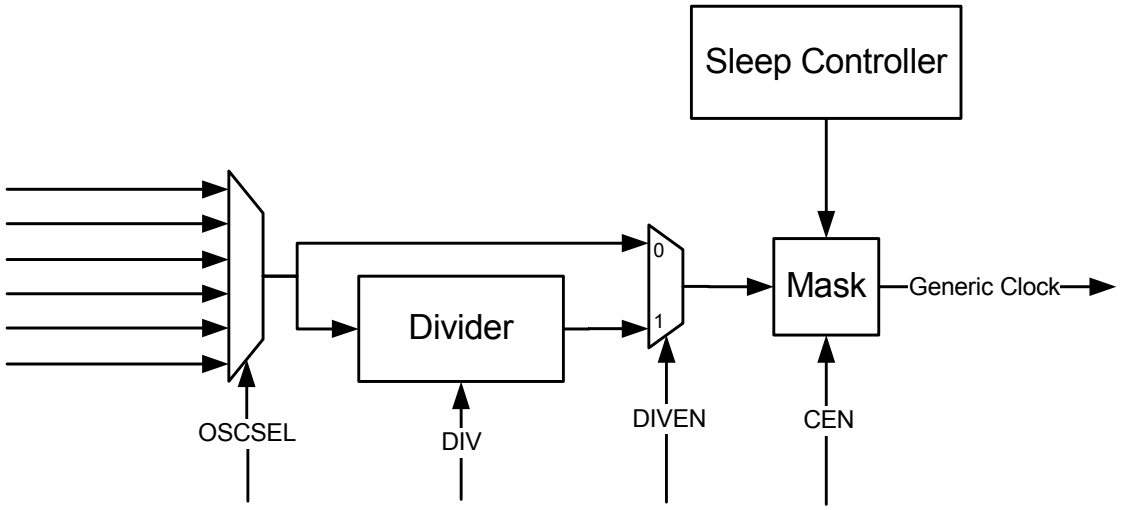
Due to multiple clock domains in the DFLLIF, values in configuration registers need to be synchronized to other clock domains. Thus, before writing to a DFLLIF configuration register check that the DFLL0RDY bit in PCLKSR register is high before writing. A write to a configuration register while DFLL0RDY is low will be ignored.

14.5.4 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 the table on Table 14-10 on page 235. 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 14-6. Generic clock generation



14.5.4.1 *Enabling a generic clock*

A generic clock is enabled by writing 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 DIVEN to one and the division factor to DIV, resulting in the output frequency:

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

14.5.4.2 *Disabling a generic clock*

The generic clock can be disabled by writing CEN to zero 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 CEN to zero, 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.

14.5.4.3 *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.

14.5.4.4 Generic clock implementation

In AT32UC3L, there are six generic clocks. These are allocated to different functions as shown in Table 14-2. Note that only GCLK4-0 are routed out.

Table 14-2. Generic clock allocation

Clock number	Function
0	DFLLIF main reference and GCLK0 pin (CLK_DFLLIF_REF)
1	DFLLIF dithering and ssg reference and GCLK1 pin (CLK_DFLLIF_DITHER)
2	AST and GCLK2 pin
3	PWMA and GCLK3 pin
4	CAT, ACIFB and GCLK4 pin
5(1)	GLOC

Note: 1. GCLK5 is not routed out.

14.5.5 Brown Out Detection (BOD)

The Brown-Out Detector (BOD) monitors the VDDCORE supply pin and compares the supply voltage to the brown-out detection level, as set in BOD.LEVEL. The BOD is disabled by default, but can be enabled either by software or by flash fuses. The 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 BOD output is available in bit PCLKSR.BODDET bit.

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

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

See Electrical Characteristics for parametric details.

Although it is not recommended to override default factory settings, it is still possible to override these default values 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.

14.5.6 Bandgap

The Flash memory, the Brown-Out Detector (BOD) and the temperature sensor 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 BOD. 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.

14.5.7 Voltage Regulator (VREG)

The embedded voltage regulator can be used to provide the VDDCORE voltage from the VDDIN. It is controlled by the VREGCR register. The voltage regulator is turned on by default at startup but can be turned off by software if an external power is provided on the VDDCORE

The 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 to override default factory settings, it is still possible to override these default values 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.

The voltage regulator interface also includes control bits for the POR33 (Power-On Reset 3.3V) detector that monitors the VDDIN voltage during power-up. The POR33 is on by default but can be turned off by software to reduce power consumption. The 3.3V Supply Monitor can then be used to monitor the VDDIN power supply (see page 200).

The RC32K oscillator must be turned on to disabled the POR33. Once the POR33 had ben disabled, the RC32K oscillator can be turned off again. Note that if the POR33 is re-enabled later, it will generate a reset.

14.5.8 System RC Oscillator (RCSYS)

The system RC oscillator (RCSYS) has a 3 cycles startup time, and is always available except in Static 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 to override default factory settings, it is still possible to override these default values 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.

14.5.9 3.3 V Supply Monitor (SM33)

The 3.3V supply monitor (SM33) is a specific voltage detector for the VDDIN voltage. It will indicate if the VDDIN voltage is above the minimum required input voltage for the voltage regulator (typically 1.75V). The user can choose to generate either a reset or an interrupt when the VDDIN voltage drops below this limit. If reset is selected, this will generate a POR reset.

In order to reduce power consumption, the SM33 can operate in “sampling mode”. In sampling mode, the SM33 is periodically turned on for a small time (just enough for making the measurement) then it is turned off for a longer time.

By default, the SM33 operates in “sampling” mode during DeepStop and Static mode and in continuous mode for other sleep modes. Sampling mode can also be forced during sleep modes other than DeepStop and Static and during normal operation.

The user can select the period of the checks in “sampling” mode through the SM33.SAMPFREQ. The sampling mode uses the 32kHz RC oscillator (RC32K) as a clock. The 32kHz RC oscillator is turned on automatically with sampling mode is used.

The POR33 is automatically turned off when enabling the SM33. The SM33 can then be used to monitor the VDDIN power supply.

14.5.10 Temperature Sensor

The Temperature Sensor is connected to the ADC channel 9. It is enabled by writing one to the EN bit in the TSENS register. Temperature sensor does not have calibration register.

Please refer to the Electrical Characteristics chapter for Temperature/Voltage curves.

14.5.11 120MHz RC Oscillator (RC120M)

The 120MHz RC 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 197. 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.

14.5.12 Backup Registers

Four 32-bit registers with are available to keep values when the chip is in Shutdown mode. Those registers will keep their content even when the VDDCORE, VDDIN and VDDIO powers are removed. The backup registers can be accessed by reading and writing BR0, BR1, BR2 and BR3 registers.

After writing to one of the backup registers the user must wait until the Backup Register Interface Ready (BRIFARDY) bit in the PCLKSR register is set before writing to another backup register. Writes to the backup register while BRIFARDY is zero will be discarded. The BRIFARDY can also trigger an interrupt if the corresponding bit is set in the IMR register.

After powering up the device the Backup Register Interface Valid (BRIFAVALID) bit in the PCLKSR register is cleared, indicating that the contents of the backup registers has not been written and contains the reset value. After writing one of the backup registers the BRIFAVALID bit is set. During writes to the backup registers (when BRIFARDY is zero) BRIFAVALID will be zero. If a reset occurs when the BRIFARDY is zero, BRIFAVALID will be cleared after the reset, indicating that the contents of the backup registers are not valid. If BRIFARDY was one when the reset occurred, BRIFAVALID will be one and the contents are the same as before the reset.

The user must ensure that BRIFAVALID and BRIFARDY is both set before reading the backup register values.

14.5.13 32kHz RC Oscillator (RC32K)

The 32kHz RC oscillator (RC32K) is enabled by default after reset, and output on PA20. The clock is available on the pad until the bit PM.PPCR.RC32OUT has been cleared in the Power Manager or a different peripheral function has been chosen on port PA20 (port PA20 will start with peripheral function "F" by default).

The RC32K can be used as source for the generic clocks, as described in "Generic clocks" on page 197.

The clock is enabled by writing one to EN bit in RC32KCR register and disabled by writing zero to the EN bit. The oscillator is also automatically turned on when the sampling mode is requested for the SM33. In this case, clearing the EN bit will not stop the RC32K until the sampling mode is no longer requested.

14.5.14 Interrupts

The SCIF has 14 interrupt sources:

- AE - Access Error:
  - Set when a protected SCIF register was accessed without first being correctly unlocked.
- BRIFARDY - Backup Register Interface Ready.
  - Set on 0 to 1 transition on the PCLKSR.BRIFARDY bit is detected.
- DFLL0RCS - DFLL Reference Clock Stopped:
  - Set on 0 to 1 transition on the PCLKSR.DFLLRCS bit is detected.
- DFLL0RDY - DFLL Ready:
  - Set on 0 to 1 transition on the PCLKSR.DFLLRDY bit is detected.
- DFLL0LOCKLOSTA - DFLL lock lost on Accurate value:
  - Set on 0 to 1 transition on the PCLKSR.DFLLLOCKLOSTA bit is detected.
- DFLL0LOCKLOSTF - DFLL lock lost on Fine value:
  - Set on 0 to 1 transition on the PCLKSR.DFLLLOCKLOSTF bit is detected.
- DFLL0LOCKLOSTC - DFLL lock lost on Coarse value:
  - Set on 0 to 1 transition on the PCLKSR.DFLLLOCKLOSTC bit is detected.
- DFLL0LOCKA - DFLL Locked on Accurate value:
  - Set on 0 to 1 transition on the PCLKSR.DFLLLOCKA bit is detected.
- DFLL0LOCKF - DFLL Locked on Fine value:
  - Set on 0 to 1 transition on the PCLKSR.DFLLLOCKF bit is detected.
- DFLL0LOCKC - DFLL Locked on Coarse value:
  - Set on 0 to 1 transition on the PCLKSR.DFLLLOCKC bit is detected.
- BODDET - Brown out detection:
  - Set on 0 to 1 transition on the PCLKSR.BODDET bit is detected.
- SM33DET - Supply Monitor 3.3V Detector:
  - Set on 0 to 1 transition on the PCLKSR.SM33DET bit is detected.
- VREGOK - Voltage Regulator OK:
  - Set on 0 to 1 transition on the PCLKSR.VREGOK bit is detected.
- OSCRDY - OSCReady:
  - Set on 0 to 1 transition on the PCLKSR.OSCRDY bit is detected.
- OSC32RDY - 32KHz Oscillator Ready:
  - Set on 0 to 1 transition on the PCLKSR.OSC32RDY bit is detected.

The interrupt sources will generate an interrupt request if the corresponding bit in the Interrupt Mask Register is set. The interrupt sources are ORed together to form one interrupt request. The SCIF 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 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). Because all the interrupt sources are ORed together, the interrupt request from the SCIF will remain active until all the bits in ISR are cleared.

14.6 User Interface

Table 14-3. 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	Oscillator Control Register	OSCCTRL0	Read/Write	0x00000000
0x0020	Oscillator 32 Control Register	OSCCTRL32	Read/Write	0x00000004
0x0024	DFLL Config Register	DFLL0CONF	Read/Write	0x00000000
0x0028	DFLL Multiplier Register	DFLL0MUL	Write-only	0x00000000
0x002C	DFLL Step Register	DFLL0STEP	Read/Write	0x00000000
0x0030	DFLL Spread Spectrum Generator Control Register	DFLL0SSG	Write-only	0x00000000
0x0034	DFLL Ratio Register	DFLL0RATIO	Read-only	0x00000000
0x0038	DFLL Synchronization Register	DFLL0SYNC	Write-only	0x00000000
0x003C	BOD Level Register	BOD	Read/Write	0x00000000
0x0040	Bandgap Calibration Register	BGCR	Read/Write	0x00000000
0x0044	Voltage Regulator Calibration Register	VREGCR	Read/Write	0x00000007
0x0048	System RC Oscillator Calibration Register	RCCR	Read/Write	0x00000000
0x004C	Supply Monitor 33 Calibration Register	SM33	Read/Write	0x00000000
0x0050	Temperature Sensor Calibration Register	TSENS	Read/Write	0x00000000
0x0058	120MHz RC Oscillator Control Register	RC120MCR	Read/Write	0x00000000
0x005C-0x0068	Backup Registers	BR	Read/Write	0x00000000
0x006C	32kHz RC Oscillator Control Register	RC32KCR	Read/Write	0x00000000
0x0070	Generic Clock Control0	GCCTRL0	Read/Write	0x00000000
0x0074	Generic Clock Control1	GCCTRL1	Read/Write	0x00000000
0x0078	Generic Clock Control2	GCCTRL2	Read/Write	0x00000000
0x007C	Generic Clock Control3	GCCTRL3	Read/Write	0x00000000
0x0080	Generic Clock Control4	GCCTR4	Read/Write	0x00000000
0x03C8	Oscillator 0 Version Register	OSC0VERSION	Read-only	-(1)
0x03CC	32 KHz Oscillator Version Register	OSC32VERSION	Read-only	-(1)
0x03D0	DFLL Version Register	DFLLIFVERSION	Read-only	-(1)
0x03D4	BOD Version Register	BODIFAVERSION	Read-only	-(1)

Table 14-3. SCIF Register Memory Map

Offset	Register	Register Name	Access	Reset
0x03D8	Voltage Regulator Version Register	VREGIFBVERSION	Read-only	-(1)
0x03DC	System RC Oscillator Version Register	RCOSCIFAVERSION	Read-only	-(1)
0x03E0	3.3V Supply Monitor Version Register	SM33IFAVERSION	Read-only	-(1)
0x03E4	Temperature Sensor Version Register	TSENSIFAVERSION	Read-only	-(1)
0x03EC	120MHz RC Oscillator Version Register	RC120MIFAVERSION	Read-only	-(1)
0x03F0	Backup Register Interface Version Register	BRIFAVERSION	Read-only	-(1)
0x03F4	32kHz RC Oscillator Version Register	RC32KIFAVERSION	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.

14.6.1 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
-	-	-	-	-	-	-	BRIFARDY

15	14	13	12	11	10	9	8
DFLL0RCS	DFLL0RDY	DFLL0LOCKLOSTA	DFLL0LOCKLOSTF	DFLL0LOCKLOSTC	DFLL0LOCKA	DFLL0LOCKF	DFLL0LOCKC

7	6	5	4	3	2	1	0
BODDET	SM33DET	VREGOK	-	-	-	OSC0RDY	OSC32RDY

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.

14.6.2 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
-	-	-	-	-	-	-	BRIFARDY

15	14	13	12	11	10	9	8
DFLL0RCS	DFLL0RDY	DFLL0LOCKLOSTA	DFLL0LOCKLOSTF	DFLL0LOCKLOSTC	DFLL0LOCKA	DFLL0LOCKF	DFLL0LOCKC

7	6	5	4	3	2	1	0
BODDET	SM33DET	VREGOK	-	-	-	OSC0RDY	OSC32RDY

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.

14.6.3 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
-	-	-	-	-	-	-	BRIFARDY

15	14	13	12	11	10	9	8
DFLL0RCS	DFLL0RDY	DFLL0LOCKLOSTA	DFLL0LOCKLOSTF	DFLL0LOCKLOSTC	DFLL0LOCKA	DFLL0LOCKF	DFLL0LOCKC

7	6	5	4	3	2	1	0
BODDET	SM33DET	VREGOK	-	-	-	OSC0RDY	OSC32RDY

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.

14.6.4 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
-	-	-	-	-	-	-	BRIFARDY

15	14	13	12	11	10	9	8
DFLL0RCS	DFLL0RDY	DFLL0LOCKLOSTA	DFLL0LOCKLOSTF	DFLL0LOCKLOSTC	DFLL0LOCKA	DFLL0LOCKF	DFLL0LOCKC

7	6	5	4	3	2	1	0
BODDET	SM33DET	VREGOK	-	-	-	OSC0RDY	OSC32RDY

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.

14.6.5 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
-	-	-	-	-	-	-	BRIFARDY

15	14	13	12	11	10	9	8
DFLL0RCS	DFLL0RDY	DFLL0LOCKLOSTA	DFLL0LOCKLOSTF	DFLL0LOCKLOSTC	DFLL0LOCKA	DFLL0LOCKF	DFLL0LOCKC

7	6	5	4	3	2	1	0
BODDET	SM33DET	VREGOK	-	-	-	OSC0RDY	OSC32RDY

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.

14.6.6 Power and Clocks Status Register

Name: PCLKSR

Access Type: Read-only

Offset: 0x0014

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	BRIFAVALID	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BRIFARDY
15	14	13	12	11	10	9	8
DFLL0RCS	DFLL0RDY	DFLL0LOCKLOSTA	DFLL0LOCKLOSTF	DFLL0LOCKLOSTC	DFLL0LOCKA	DFLL0LOCKF	DFLL0LOCKC
7	6	5	4	3	2	1	0
BODDET	SM33DET	VREGOK	-	-	-	OSC0RDY	OSC32RDY

- **BRIFAVALID: Backup Register Interface Valid**
  0: The values in the backup registers are not valid.
  1: The values in the backup registers are valid.
- **BRIFARDY: Backup Register Interface Ready**
  0: The backup register interface is busy updating the backup registers. Writes to BRn will be discarded.
  1: The backup register interface is ready to accept new writes to the backup registers.
- **DFLLRCS: DFLL0 Reference Clock Stopped**
  0: The DFLL0 reference clock is running, or has never been enabled.
  1: The DFLL0 reference clock has stopped or is too slow.
- **DFLL0RDY: DFLL0 Synchronization Ready**
  0: Read or write to DFLL0 registers is invalid
  1: Read or write to DFLL0 registers is valid
- **DFLLLOCKLOSTA: DFLL lock lost on Accurate value**
  0: DFLL has not lost its Accurate lock or has never been enabled.
  1: DFLL has lost its Accurate lock, either by disabling the DFLL or due to faulty operation.
- **DFLLLOCKLOSTF: DFLL lock lost on Fine value**
  0: DFLL has not lost its Fine lock or has never been enabled.
  1: DFLL has lost its Fine lock, either by disabling the DFLL or due to faulty operation.
- **DFLLLOCKLOSTC: DFLL lock lost on Coarse value**
  0: DFLL has not lost its Coarse lock or has never been enabled.
  1: DFLL has lost its Coarse lock, either by disabling the DFLL or due to faulty operation.
- **DFLLLOCKA: DFLL Locked on Accurate value**
  0: DFLL is unlocked on Accurate value.
  1: DFLL is locked on Accurate value, and is ready to be selected as clock source with an accurate output clock.
- **DFLLLOCKF: DFLL Locked on Fine value**

0: DFLL is unlocked on Fine value.

1: DFLL is locked on Fine value, and is ready to be selected as clock source with a highly accurate output clock.

- **DFLLLOCKC: DFLL Locked on Coarse value**
  0: DFLL is unlocked on Coarse value.
  1: DFLL is locked on Fine value, and is ready to be selected as clock source with medium accuracy on the output clock.
- **BODDET: Brown out detection**
  0: No BOD Event.
  1: BOD has detected that power supply is going below BOD reference value.
- **SM33DET: Supply Monitor 3.3V Detector**
  0: SM33 not enabled or the supply voltage is above the SM33 threshold.
  1: SM33 enabled and the supply voltage is below the SM33 threshold.
- **VREGOK: Voltage Regulator OK**
  0: Voltage regulator not enabled or not ready.
  1: Voltage regulator has reached its output threshold value after being enabled.
- **OSC0RDY: OSC0 Ready**
  0: Oscillator not enabled or not ready.
  1: 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.

14.6.7 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.

14.6.8 Oscillator Control Register

Name: OSCCTRL0

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
-	-	-	-	-	-	-	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 14-4. Startup Time for Oscillator 0

STARTUP	Number of System RC oscillator clock cycle	Approximative Equivalent time (RCSYS = 115kHz)
0	0	0
1	64	560us
2	128	1.1ms
3	2048	18ms
4	4096	36ms
5	8192	71ms
6	16384	142ms
7	Reserved	Reserved

- **AGC: Automatic Gain Control**
  For test purposes
- **GAIN: Gain**
  Set the gain for the oscillator as described in Table 14-5.

Table 14-5. Gain for Oscillator 0

GAIN	Description
0	Oscillator is used with gain G0 (XIN from 0.4 MHz to 12.0 MHz).
1	Oscillator is used with gain G1 (XIN from 12.0 MHz to 16.0 MHz).
2	Oscillator is used with gain G2 (XIN from 16.0 MHz to 20.0 MHz).
3	Oscillator is used with gain G3 (used for e.g. increasing S/N ratio, better drive strength for high ESR crystals.

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

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.

14.6.9 32KHz Oscillator Control Register

Name: OSCCTRL32

Access Type: Read/Write

Offset: 0x0020

Reset Value: 0x00000004

31	30	29	28	27	26	25	24
RESERVED	-	-	-	-	-	-	-
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
-	-	-	-	EN1K	EN32K	PINSEL	OSC32EN

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

- **RESERVED**
  This bit should always be written to zero.
- **STARTUP: Oscillator Startup Time**
  Select startup time for 32 KHz oscillator

Table 14-6. Startup time for 32 KHz oscillator

STARTUP	Number of RC oscillator clock cycle	Approximative Equivalent time (RCOsc = 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 14-7. Operation mode for 32 KHz oscillator

MODE	Description
0	External clock connected to XIN32, XOUT32 can be used as I/O (no crystal)
1	Crystal mode. Crystal is connected to XIN32/XOUT32.
2	Reserved
3	Reserved
4	Crystal and high current mode. Crystal is connected to XIN32/XOUT32.
5	Reserved
6	Reserved
7	Reserved

- **EN1K: Enable the 1 KHz output**
  0: 1 KHz output is disabled
  1: 1 KHz output is enabled
- **EN32K: Enable the 32 KHz output**
  0: 32 KHz output is disabled
  1: 32 KHz output is enabled
- **PINSEL: Select pins used for 32 KHz crystal oscillator**
  0: Default pins
  1: Alternate pins: XIN32_2 pin is used instead of XIN32 pin, XOUT32_2 pin is used instead of XOUT32
- **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.

14.6.10 DFLL0 Configuration Register

Name: DFLL0CONF

Access Type: Read/Write

Offset: 0x0024

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
COARSE							

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

15	14	13	12	11	10	9	8
FINE							

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

- **COARSE: Coarse calibration register value**
  Set the value of the coarse calibration register. If in closed loop mode, this field is Read-only.
- **FINE: FINE calibration register value**
  Set the value of the fine calibration register. If in closed loop mode, this field is Read-only.
- **DITHER: Enable Dithering**
  0: The fine LSB input to the VCO is constant
  1: The fine LSB input to the VCO is dithered to achieve fractional approximation to the correct multiplication ratio
- **MODE: Mode Selection**
  0: DFLL in Open Loop operation.
  1: DFLL in Closed Loop operation.
- **EN: Enable**
  0: DFLL is disabled.
  1: DFLL 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.

14.6.11 DFLL0 Multiplier Register

Name: DFLL0MUL

Access Type: Read/Write

Offset: 0x0028

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
IMUL[15:8]							

23	22	21	20	19	18	17	16
IMUL[7:0]							

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

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

- **IMUL: DFLL Integer Multiply Factor**
  This field, together with FMUL, determines the ratio of the DFLL output frequency to the source clock frequency. The IMUL field is used as the integer part, while the FMUL field is used as the fractional part.
  In open loop mode, writing to this register has no effect.
- **FMUL: DFLL Fractional Multiply Factor**
  This field, together with IMUL, determines the ratio of the DFLL output frequency to the source clock frequency. The IMUL field is used as the integer part, while the FMUL field is used as the fractional part.
  In open loop mode, writing to this register has no effect.

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.

14.6.12 DFLL0 Maximum Step Register

Name: DFLL0STEP

Access Type: Read/Write

Offset: 0x002C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	FSTEP[8]

23	22	21	20	19	18	17	16
FSTEP[7:0]							

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

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

- **FSTEP: Fine Maximum Step**
  This indicated the maximum step size during fine adjustment in closed-loop mode. When adjusting to a new frequency, the expected overshoot of that frequency depends on this step size.
- **CSTEP: Coarse Maximum Step**
  This indicated the maximum step size during coarse adjustment in closed-loop mode. When adjusting to a new frequency, the expected overshoot of that frequency depends on this step size.

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.

14.6.13 DFLL0 Spread Spectrum Generator Control Register

Name: DFLL0SSG

Access Type: Read/Write

Offset: 0x0030

Reset Value: 0x00000000

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

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

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

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

- **STEPSIZE: SSG Step Size.**
  Sets the step size of the spread spectrum. If set to zero or one, the value added to the FINE bits will be incremented/decremented with one on every positive edge of the input clock. If set to *n*, the value added to the FINE bits will be incremented/decremented with *n* on every positive edge of the input clock.
- **AMPLITUDE: SSG Amplitude.**
  Sets the amplitude of the spread spectrum. If set to zero, only the LSB of the FINE bits will be affected. If set to one, the sequence {1, 0, -1, 0} will be added to FINE bits, etc.
- **PRBS: Pseudo Random Bit Sequence**
  0: Each spread spectrum frequency is applied at constant intervals
  1: Each spread spectrum frequency is applied at pseudo-random intervals
- **EN: Enable**
  0: SSG is disabled.
  1: SSG 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.

14.6.14 DFLL0 Ratio Register

Name: DFLL0RATIO

Access Type: Read-only

Offset: 0x0034

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
RATIODIFF[15:8]							

23	22	21	20	19	18	17	16
RATIODIFF[7:0]							

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

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

- **RATIODIFF: Multiplication Ratio Difference**
  In closed-loop mode, this field indicates the error in the ratio between the VCO frequency and the target frequency.
- **NUMREF: Numerical Reference**
  The number of reference clock cycles used to measure the VCO frequency equals 2^NUMREF.

14.6.15 DFLL0 Synchronization Register

Name: DFLL0SYNC

Access Type: Write-only

Offset: 0x0038

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: Synchronization**
  To be able to read the current value of DFLL0CONF or DFLL0RATIO in closed-loop mode, this bit should be written to one. The updated value is available in DFLL0CONF and DFLL0RATIO when PCLKSR.DFLL0RDY goes high.

14.6.16 BOD Control Register

Name: BOD

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
-	-	-	-	-	-	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: Fuse Calibration Done**
  Set to 1 when the CTRL, HYST and LEVEL fields 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 BOD reset.
- **CTRL: BOD Control**

Table 14-8. Operation mode for BOD

CTRL	Description
0	BOD is off
1	BOD is enabled and can reset the chip
2	BOD is enabled and but cannot reset the chip. Only interrupt will be sent to interrupt controller, if enabled in the IMR register.
3	Reserved

- **HYST: BOD Hysteresis**
  0: No hysteresis
  1: Hysteresis on.
- **LEVEL: BOD Level**
  This field sets the triggering threshold of the BOD. See Electrical Characteristics for actual voltage levels.
  Note that any change to the LEVEL field of the BOD register should be done with the BOD 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.

14.6.17 Bandgap Calibration Register

Name: BGCR

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
-	-	-	-	-	-	-	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 BOD 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.

14.6.18 Voltage Regulator Calibration Register

Name: VREGCR

Access Type: Read/Write

Offset: 0x0044

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	POR33MASK	POR33STATUS	POR33EN	DEEPDIS	FCD

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

7	6	5	4	3	2	1	0
ON	VREGOK	EN	-	-	SELVDD		

- **SFV: Store Final Value**
  0: The register is read/write
  1: The register is read-only, to protect against further accidental writes.
- **INTPD: Internal Pulldown**
  This bit is used for test purposes only.
  0: The voltage regulator output is not pulled to ground
  1: The voltage regulator output has a pulldown to ground
- **POR33MASK: Power-On Reset 3.3V output Mask**
  0: Power-On Reset 3.3V is not masked
  1: Power-On Reset 3.3V is masked
- **POR33STATUS: Power-On Reset 3.3V Status**
  0: Power-On Reset 3.3V is currently disabled
  1: Power-On Reset 3.3V is currently enabled
  This bit is a read-only bit. Writing to this bit has no effect.
- **POR33EN: Power-On Reset 3.3V Enable**
  0: Disable the 3.3V POR detector
  1: Enable the 3.3V POR detector
- **DEEPDIS: Disable Regulator Deep Mode**
  0: Regulator will enter deep mode in low-power sleep modes for lower power consumption
  1: Regulator will stay in full-power mode in all sleep modes for shorter start-up time
- **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 Regulator. See Electrical Characteristics for voltage values.
- **ON: Voltage Regulator On Status**

0: The voltage regulator is currently turned off

1: The voltage regulator is currently turned on

This bit is a read-only bit. Writing to this bit has no effect.

- **VREGOK: Voltage Regulator OK Status**

  0: The voltage regulator is disabled or has not yet reached a stable output voltage

  1: The voltage regulator has reached the output voltage threshold level after being enabled

  This bit is a read-only bit. Writing to this bit has no effect

- **EN: Enable the voltage regulator**

  0: The voltage regulator is disabled

  1: The voltage regulator is enabled

  Note: This bit is set to one after a Power On Reset

- **SELVDD: Select VDD**

  Output voltage of the Voltage Regulator. 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.

14.6.19 RC Oscillator Calibration Register

Name: RCCR

Access Type: Read/Write

Offset: 0x0048

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.

14.6.20 Supply Monitor 33 Calibration Register

Name: SM33

Access Type: Read/Write

Offset: 0x004C

Reset Value: 0x00000000

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

- **SAMPFREQ: Sampling Frequency**
  Selects the sampling mode frequency of the 3.3V supply monitor. In sampling mode, the SM33 performs a measurement every $2^{(SAMPFREQ+5)}$ cycles of the internal 32kHz RC oscillator.
- **ONSM: Supply Monitor On Indicator**
  0: The supply monitor is off.
  1: The supply monitor is on.
  This bit is read-only. Writing to this bit has no effect.
- **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 CALIB field has been updated by the Flash fuses after a reset.
- **CALIB: Calibration Value**
  Calibration Value for the SM33.
- **FS: Force Sampling Mode**
  0: Sampling mode is enabled in DeepStop and Static mode only.
  1: Sampling mode is always enabled.
- **CTRL: Supply Monitor Control**
  Selects the operating mode for the SM33

Table 14-9. Operation mode for SM33

CTRL	Description
0	SM33 is off

Table 14-9. Operation mode for SM33

CTRL	Description
1	SM33 is enabled and can reset the chip
2	SM33 is enabled and but cannot reset the chip. Only interrupt will be sent to interrupt controller, if enabled in the IMR register.
3	SM33 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.

14.6.21 Temperature Sensor Calibration Register

Name: TSENS

Access Type: Read/Write

Offset: 0x0050

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: Temperature Sensor Enable**
  0: Temperature sensor is disabled
  1: Temperature sensor 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.

14.6.22 120MHz RC Oscillator Configuration Register

Name: RC120M

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.

14.6.23 Backup Register n

Name: BRn

Access Type: Read/Write

Offset: 0x005C-0x0068

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]							

This is a set of general purpose read-write registers. Data stored in these registers is retained when the device is shut off. Before writing to these registers the user must ensure that BRIFARDY in the PCLKSR register is not set.

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.

14.6.24 32kHz RC Oscillator Configuration Register

Name: RC32KCR

Access Type: Read/Write

Offset: 0x006C

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: RC32K 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.

14.6.25 Generic Clock Control

Name: GCCTRLn

Access Type: Read/Write

Offset: 0x0070+n*0x0004

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 design.

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

Table 14-10. Generic Clock Sources

OSCSEL	Clock/Oscillator	Description
0	RCSYS	System RC oscillator clock
1	OSC32K	Output clock from OSC32K
2	DFLL	Output clock from DFLL
3	OSC0	Output clock from Oscillator
4	RC120M	Output from 120MHz RCOSC
5	CLK_CPU	The clock the CPU runs on
6	CLK_HSB	High Speed Bus clock
7	CLK_PBA	Peripheral Bus A clock
8	CLK_PBB	Peripheral Bus B clock
9	RC32K	Output from 32KHz RCOSC
10	Reserved	
11	CLK_1K	1KHz output clock from OSC32K
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.

14.6.26 Oscillator 0 Version Register

Name: OSC0VERSION

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.

14.6.27 32KHz Oscillator Version Register

Name: OSC32VERSION

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.

14.6.28 Digital Frequency Locked Loop Version Register

Name: DFLLIF VERSION

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.

14.6.29 Brown-Out Detector Version Register

Name: BODIFAVERSION

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.

14.6.30 Voltage Regulator Version Register

Name: VREGIFBVERSION

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.

14.6.31 RC Oscillator Version Register

Name: RCOSCIFAVERSION

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.

14.6.32 3.3V Supply Monitor Version Register

Name: SM33IFAVERSION

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.

14.6.33 Temperature Sensor Version Register

Name: TSENSIFAVERSION

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.

14.6.34 120MHz RC Oscillator Version Register

Name: RC120MIFAVERSION

Access Type: Read-only

Offset: 0x03EC

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.

14.6.35 Backup Register Interface Version Register

Name: BRIFAVERSION

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.

14.6.36 32kHz RC Oscillator Version Register

Name: RC32KIFAVERSION

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.

14.6.37 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.

14.6.38 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.

14.7 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 14-11. SCIF Clock Name

Module Name	Clock Name
SCIF	CLK_SCIF

Table 14-12. Register Reset Values

Register	Reset Value
OSC0VERSION	0x00000100
OSC32VERSION	0x00000101
DFLLVIFERSION	0x00000201
BODIFAVERSION	0x00000101
VREGIFBVERSION	0x00000101
RCOSCIFAVERSION	0x00000101
SM33IFAVERSION	0x00000100
TSENSIFAVERSION	0x00000100
RC120MIFAVERSION	0x00000101
BRIFAVERSION	0x00000100
RC32KIFAVERSION	0x00000100
GCLKVERSION	0x00000100
VERSION	0x00000102

15. Asynchronous Timer (AST)

Rev: 3.0.0.1

15.1 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**
- **Operation and wakeup during shutdown**
- **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**

15.2 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 five different clock sources, including the low-power 32KHz oscillator, 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.

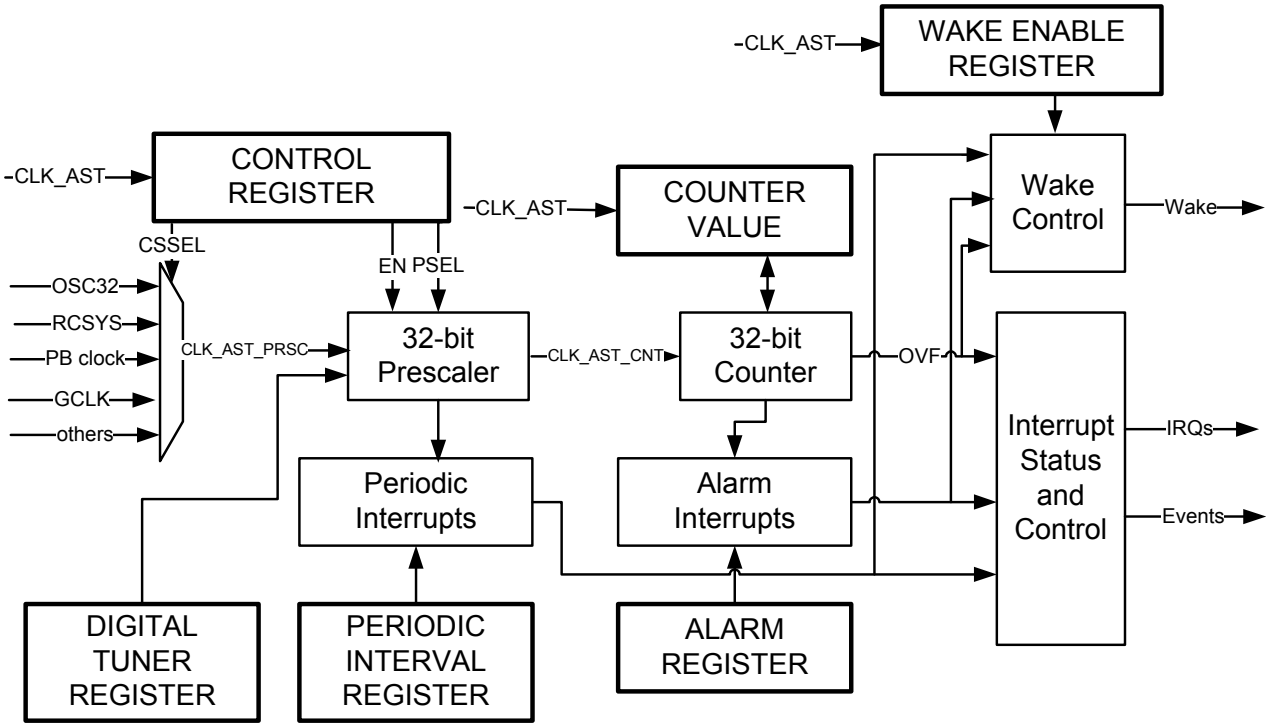
The AST can generate periodic interrupts and peripheral events from output from the prescaler, as well as alarm interrupts and peripheral events, which can trigger at any counter value. Additionally, the timer can trigger an overflow interrupt and peripheral event, and be reset on the occurrence of any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

To keep track of time during shutdown the AST can run while the rest of the core is powered off. This will reduce the power consumption when the system is idle. The AST can also wake up the system from shutdown using either the alarm wakeup, periodic wakeup. or overflow wakeup mechanisms.

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

15.3 Block Diagram

Figure 15-1. Asynchronous Timer Block Diagram



15.4 Product Dependencies

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

15.4.1 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.

15.4.2 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.
- 1KHz clock from the 32KHz oscillator (CLK_1K). This clock is only available in crystal mode, and must be enabled before use.

In Shutdown mode only the 32KHz oscillator and the 1KHz clock are available, using certain pins. Please refer to the Power Manager chapter for details.

15.4.3 Interrupts

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

15.4.4 Peripheral Events

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

15.4.5 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.

15.5 Functional Description

15.5.1 Initialization

Before enabling the AST, the internal AST clock CLK_AST_PRSC must be enabled, following the procedure specified in Section 15.5.1.1. 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).

15.5.1.1 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

15.5.1.2 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 the CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero

15.5.2 Basic Operation

15.5.2.1 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.

15.5.2.2 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 an alarm occurs (see Section 15.5.3.2 on page 256. This 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.

15.5.2.3 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 (ARn) must be written in time/date format for the alarm to trigger correctly.

15.5.3 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 15.5.3.1., and the generation of the ALARM interrupt is described in Section 15.5.3.2. 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 15.5.8 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.

15.5.3.1 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 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.

15.5.3.2 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.

15.5.4 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 15.5.3.1. The ALARMn peripheral event(s) is generated the same way as the ALARM interrupt, as described in Section 15.5.3.2. The OVF peripheral event is generated the same way as the OVF interrupt, as described in Section 15.5.3-

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.

15.5.5 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 noe 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.

15.5.6 Shutdown Mode

If the AST is configured to use a clock that is available in Shutdown mode, the AST can be used to wake up the system from shutdown. Both the alarm wakeup, periodic wakeup, and overflow wakeup mechanisms can be used in this mode.

When waking up from Shutdown mode all control registers will have the same value as before the shutdown was entered, except the Interrupt Mask Register (IMR). IMR will be reset with all interrupts turned off. The software must first reconfigure the interrupt controller and then enable the interrupts in the AST to again receive interrupts from the AST.

The CV register will be updated with the current counter value directly after wakeup from shutdown. The SR will show the status of the AST, including the status bits set during shutdown operation.

When waking up the system from shutdown the CPU will start executing code from the reset start address.

15.5.7 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. The resulting frequency is

$$f_{TUNED} = f_0 \left(1 \pm \frac{1}{\left(\frac{1}{VALUE}\right) \cdot (2^{(EXP+8)}) - 1} \right)$$

for $VALUE > 0$, where f_0 is the original frequency of the prescaler. VALUE and EXP are chosen by writing the selected value to the corresponding filed in DTR. If $VALUE = 0$, the frequency is unchanged.

15.5.8 Synchronization

As the prescaler and counter operate asynchronously from the user interface, the AST needs a few clock cycles to synchronize the values written to the CR, CV, SCR, WER, EVE, EVD, PIRn, ARn, 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.

15.6 User Interface

Table 15-1. 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 Modue 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.

15.6.1 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

- **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.

15.6.2 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]							

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

15.6.3 Status Register

Name: SR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	CLKRDY	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

- **CLKRDY: 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, PIRn, and DTR.
  1: The AST is busy and will discard writes to CR, CV, SCR, WER, EVE, EVD, ARn, 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.

15.6.4 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	CLKRDY	-	-	-	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 SR and the corresponding interrupt request.

15.6.5 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

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

15.6.6 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

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

15.6.7 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x00000000

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

15.6.8 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

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.

15.6.9 Alarm Register 0

Name: AR0

Access Type: Read/Write

Offset: 0x20

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]							

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

15.6.10 Alarm Register 1

Name: AR1

Access Type: Read/Write

Offset: 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]							

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

15.6.11 Periodic Interval Register 0

Name: PIR0

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
-	-	-	-	-	-	-	-

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

- **INSEL: Interval Select**
  The PER0 bit in SR will be set when the INSEL bit in the prescaler has a 0-to-1 transition.

15.6.12 Periodic Interval Register 1

Name: PIR1

Access Type: Read/Write

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
-	-	-	INSEL				

- **INSEL: Interval Select**
  The PER1 bit in SR will be set when the INSEL bit in the prescaler has a 0-to-1 transition.

15.6.13 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

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

Table 15-2. Clock Source Selection

CSSEL	Clock Source
0	System RC oscillator (RCSYS)
1	32KHz oscillator (OSC32K)
2	PB clock
3	Generic clock (GCLK)
4	1KHz clock from 32KHz oscillator (CLK_1K)

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

15.6.14 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				

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

- **ADD:**
  0: The resulting frequency is $f = f0\left(1 - \frac{1}{\left(\frac{1}{VALUE}\right) \cdot 2^{(EXP+8)} - 1}\right)$ for $VALUE > 0$.

  1: The resulting frequency is $f = f0\left(1 + \frac{1}{\left(\frac{1}{VALUE}\right) \cdot 2^{(EXP+8)} - 1}\right)$ for $VALUE > 0$.

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

15.6.15 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

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.

15.6.16 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

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.

15.6.17 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.

15.6.18 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					

- **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.

15.6.19 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 not implemented.
  1: Digital tuner implemented.
- **DTREXPWA: Digital Tuner Exponent Writeable**
  0: Digital tuner exponent is a constant value. Writes to EXP field in DTR will be discarded.
  1: Digital tuner exponent is chosen by writing to EXP field in DTR.
- **DTREXPVALUE: Digital Tuner Exponent Value**
  Digital tuner exponent value if DTEXPWA 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.

15.6.20 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.

15.7 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. Please refer to the Power Manager chapter for details.

Table 15-3. AST Configuration

Feature	AST
Number of alarm comparators	1
Number of periodic comparators	1
Digital tuner	On

Table 15-4. AST Clocks

Clock Name	Description
CLK_AST	Clock for the AST bus interface
GCLK	The generic clock used for the AST is GCLK2
PB clock	Peripheral Bus clock from the PBA clock domain

Table 15-5. Register Reset Values

Register	Reset Value
VERSION	0x00000300
PARAMETER	0x00004103

16. Watchdog Timer (WDT)

Rev: 4.0.2.0

16.1 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**

16.2 Overview

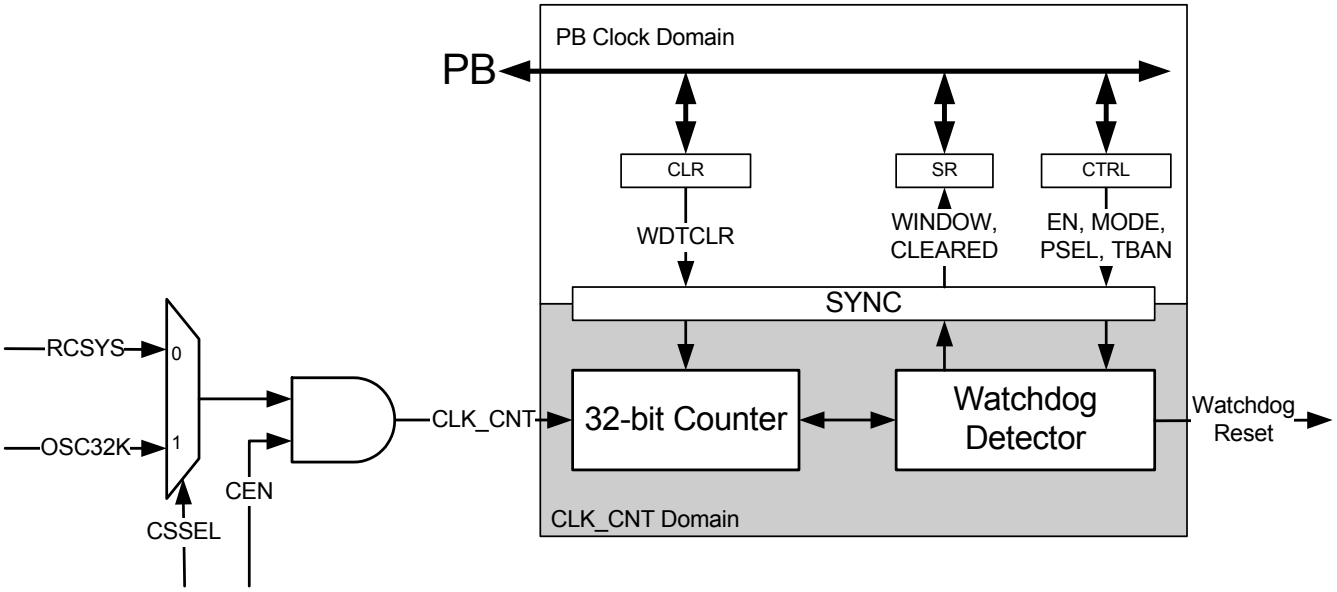
The Watchdog Timer (WDT) will reset the device unless it is periodically serviced by the software. This allows the device to recover from a condition that has caused the system to be unstable.

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.

16.3 Block Diagram

Figure 16-1. WDT Block Diagram



16.4 Product Dependencies

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

16.4.1 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.

16.4.2 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.

16.4.3 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.

16.4.4 Fuses

The WDT can be enabled at reset. This is controlled by the WDTAUTO fuse, see Section 16.5.4 for details. Please refer to the Fuse Settings section in the Flash Controller chapter for details about WDTAUTO and how to program the fuses.

16.5 Functional Description

16.5.1 Basic Mode

16.5.1.1 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.

16.5.1.2 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.

16.5.1.3 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:

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

16.5.1.4 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.

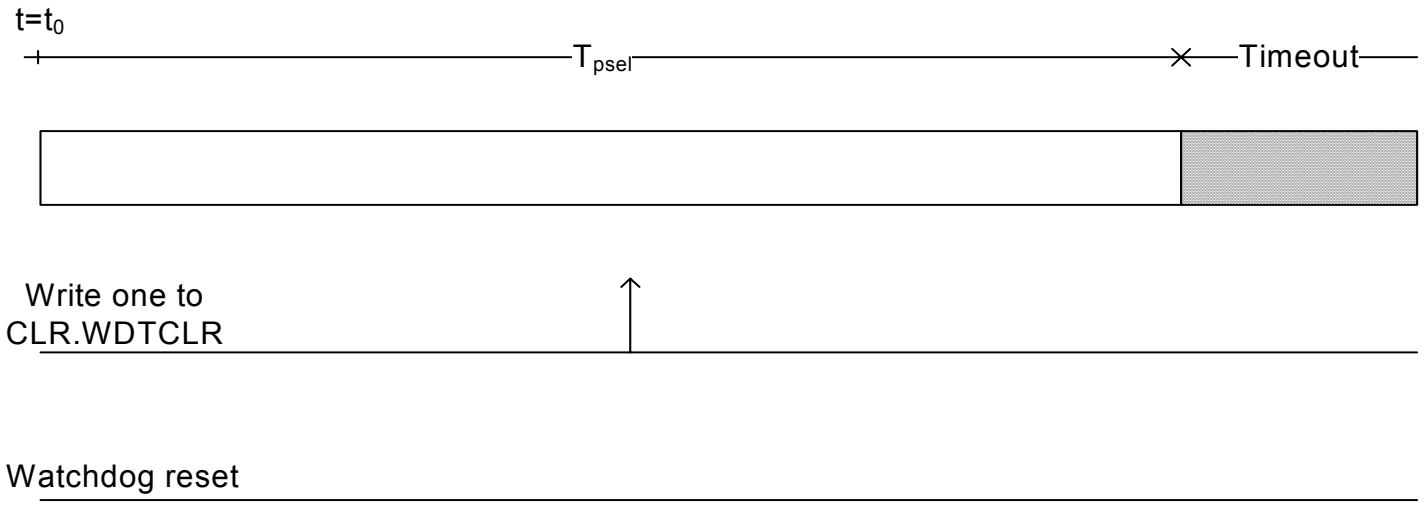
16.5.1.5 Clearing the WDT Counter

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 be ignored.

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.

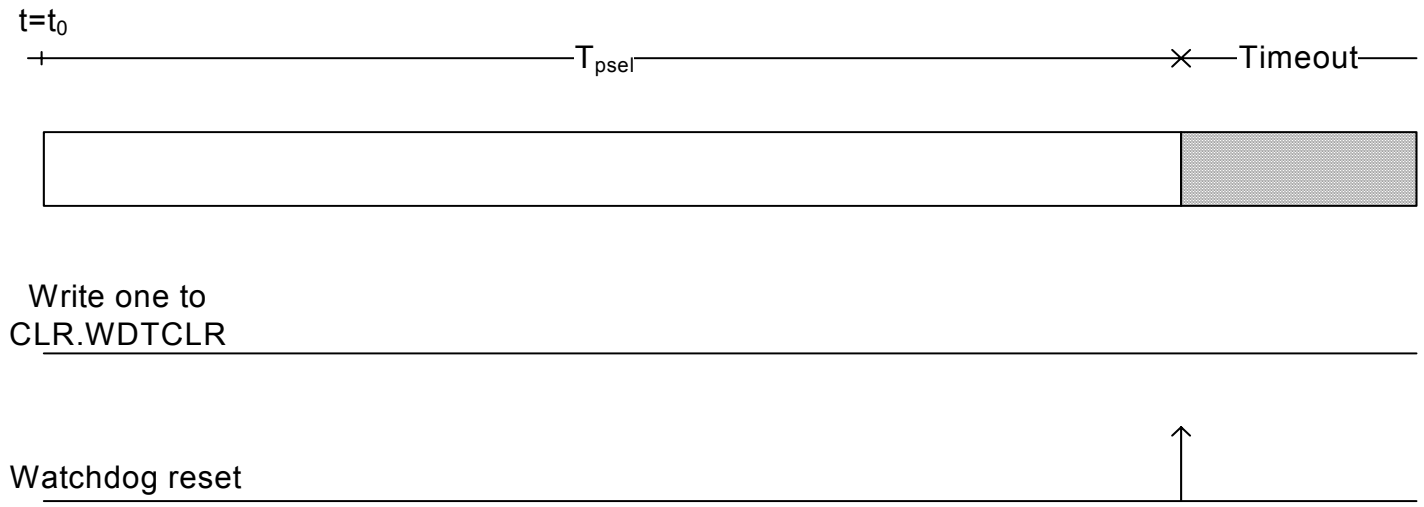
If the WDT counter is periodically cleared within T_{psel} no watchdog reset will be issued, see Figure 16-2 on page 284.

Figure 16-2. 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 16-3 on page 284.

Figure 16-3. Basic Mode WDT Timing Diagram, no clear within T_{psel}.



16.5.1.6 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.

16.5.2 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 16-5 on page 286. 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 16-4 on page 285.

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 16-4. Window Mode WDT Timing Diagram

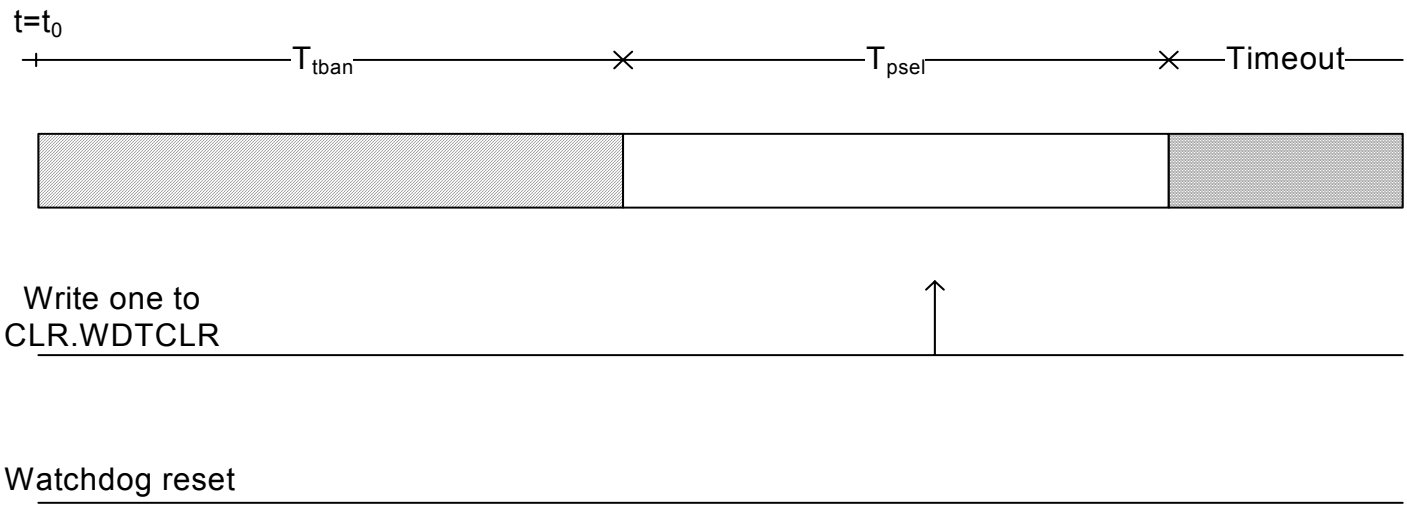
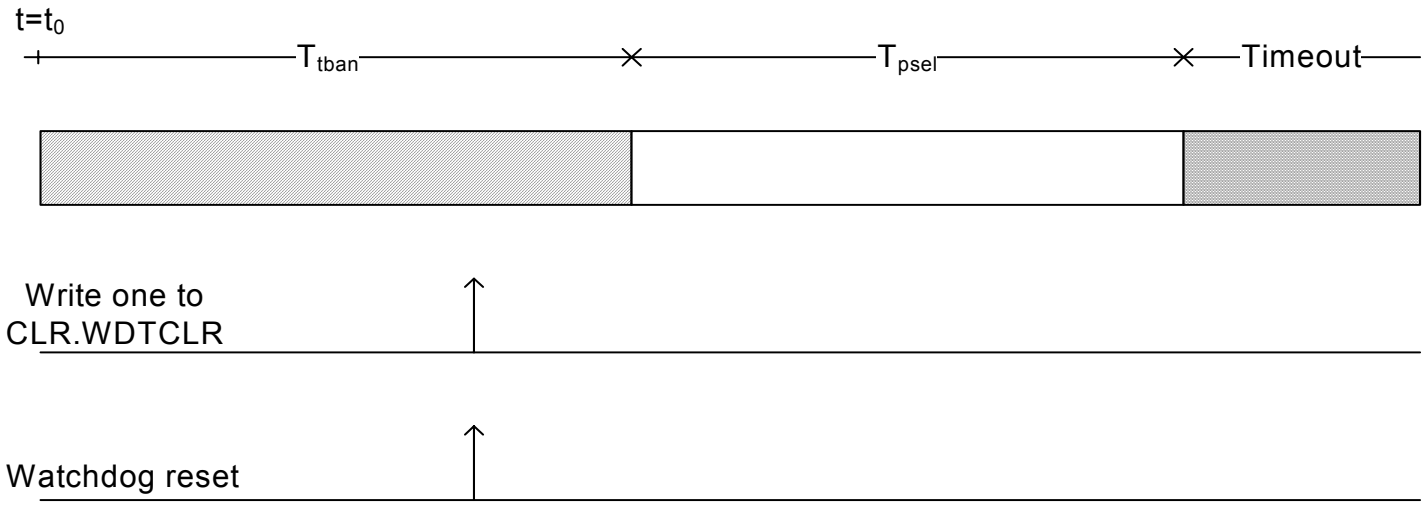


Figure 16-5. Window Mode WDT Timing Diagram, clearing within T_{tban}, resulting in watchdog reset.



16.5.3 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.

16.5.4 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.

16.5.5 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).

16.6 User Interface

Table 16-1. 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.

16.6.1 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.

16.6.2 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.

16.6.3 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.

16.6.4 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.

16.7 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 16-2. Module clock name

Module name	Clock name
MODULE	CLK_WDT

Table 16-3. Register Reset Values

Register	Reset Value
VERSION	0x00000402

17. External Interrupt Controller (EIC)

Rev: 3.0.1.0

17.1 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**
- **Maskable NMI interrupt**

17.2 Overview

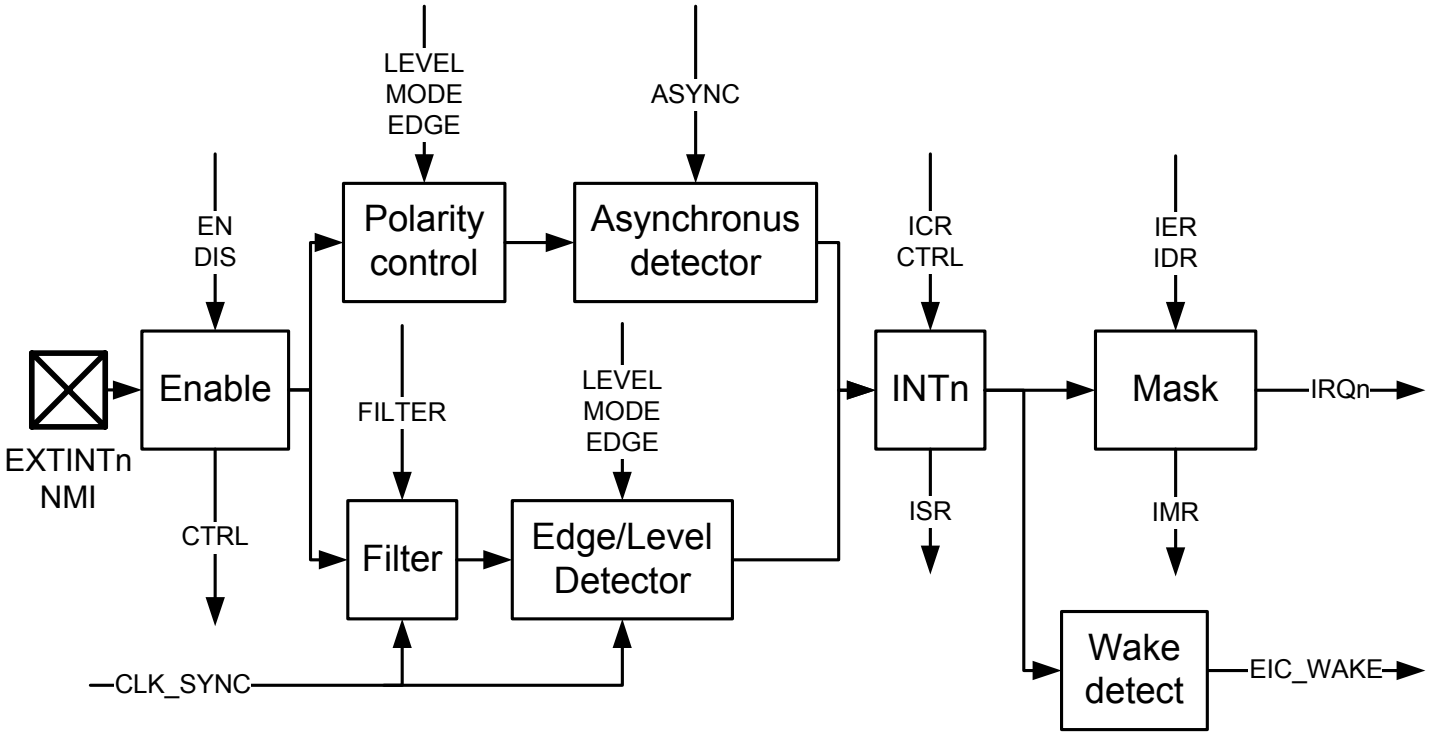
The External Interrupt Controller (EIC) allows pins to be configured as external interrupts. Each external interrupt has its own interrupt request and can be individually masked. Each external interrupt can generate an interrupt on rising or falling edge, or high or low level. Every interrupt input has 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.

17.3 Block Diagram

Figure 17-1. EIC Block Diagram



17.4 I/O Lines Description

Table 17-1. I/O Lines Description

Pin Name	Pin Description	Type
NMI	Non-Maskable Interrupt	Input
EXTINTn	External Interrupt	Input

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 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.

17.5.2 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.

17.5.3 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.

17.5.4 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.

17.5.5 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.

17.6 Functional Description

17.6.1 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.

17.6.2 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 17-2 and Figure 17-3 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 17-2 and Figure 17-3 for examples (FILTER on).

Figure 17-2. Timing Diagram, Synchronous Interrupts, High Level or Rising Edge

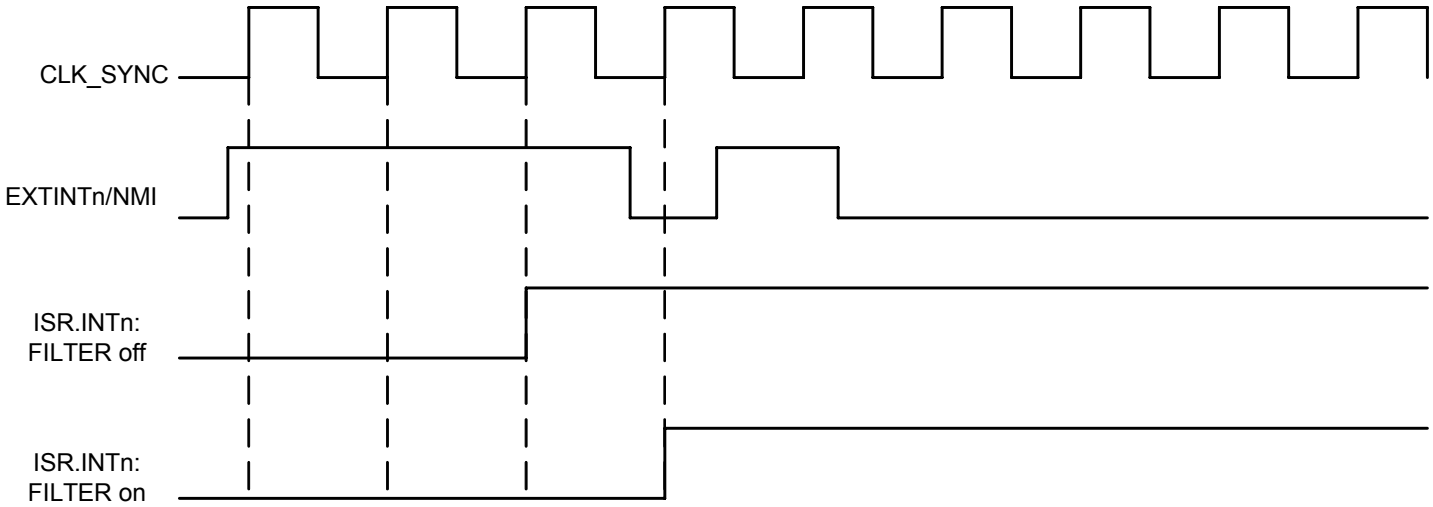
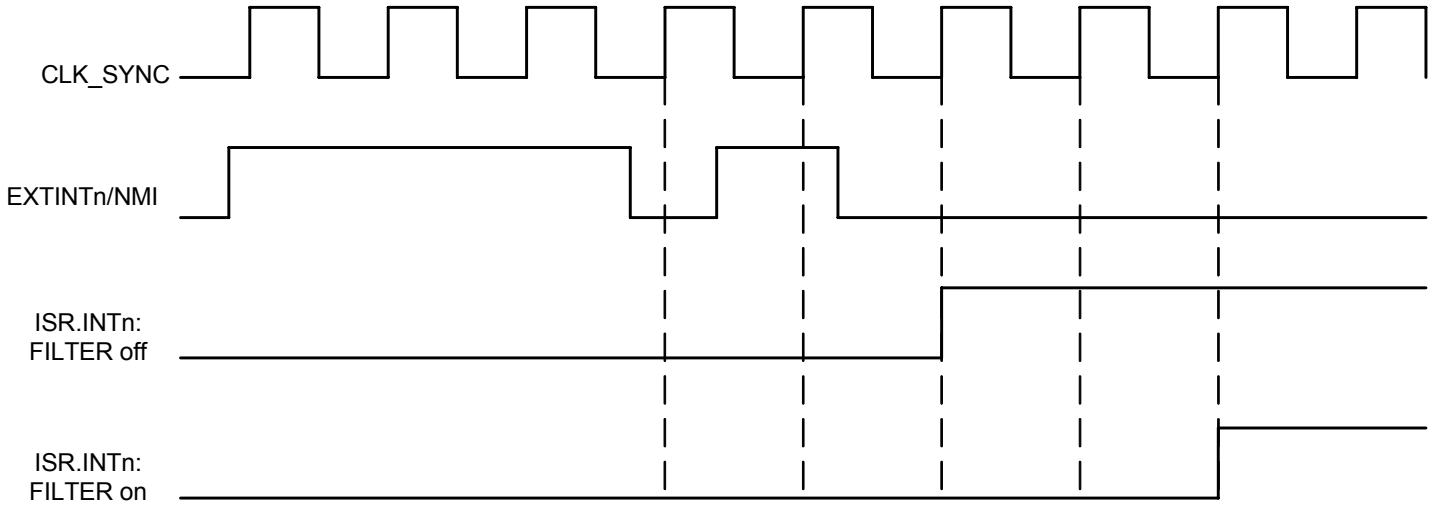


Figure 17-3. Timing Diagram, Synchronous Interrupts, Low Level or Falling Edge



17.6.3 Non-Maskable Interrupt

The NMI supports the same features as the external interrupts, and is accessed through the same registers. The description in Section 17.6.1 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.

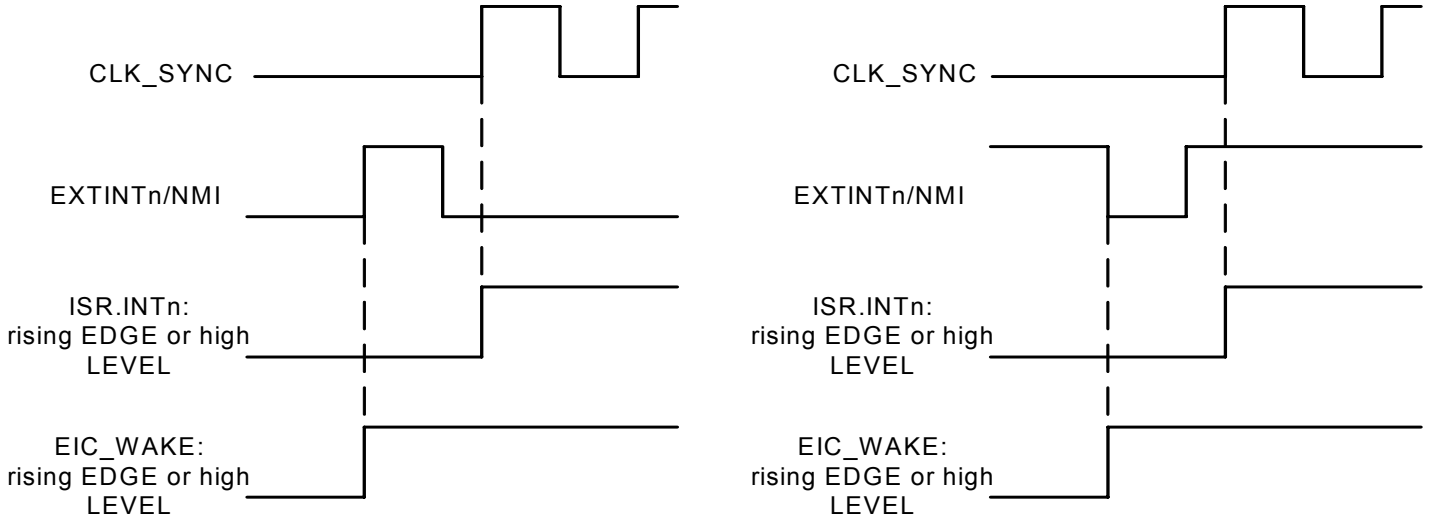
17.6.4 Asynchronous Interrupts

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 17-4 on page 299 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 17-4. Timing Diagram, Asynchronous Interrupts



17.6.5 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.

17.7 User Interface

Table 17-2. 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
0x3CF	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.

17.7.1 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.

17.7.2 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.

17.7.3 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.

17.7.4 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.

17.7.5 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.

17.7.6 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.

17.7.7 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.

17.7.8 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.

17.7.9 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.

17.7.10 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.

17.7.11 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.

17.7.12 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.

17.7.13 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.

17.7.14 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.

17.7.15 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.

17.8 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 17-3. Module Configuration

Feature	EIC
Number of external interrupts, including NMI	6

Table 17-4. Module Clock Name

Module Name	Clock Name
EIC	CLK_EIC

Table 17-5. Register Reset Values

Register	Reset Value
VERSION	0x00000301

18. Frequency Meter (FREQM)

Rev: 3.0.1.1

18.1 Features

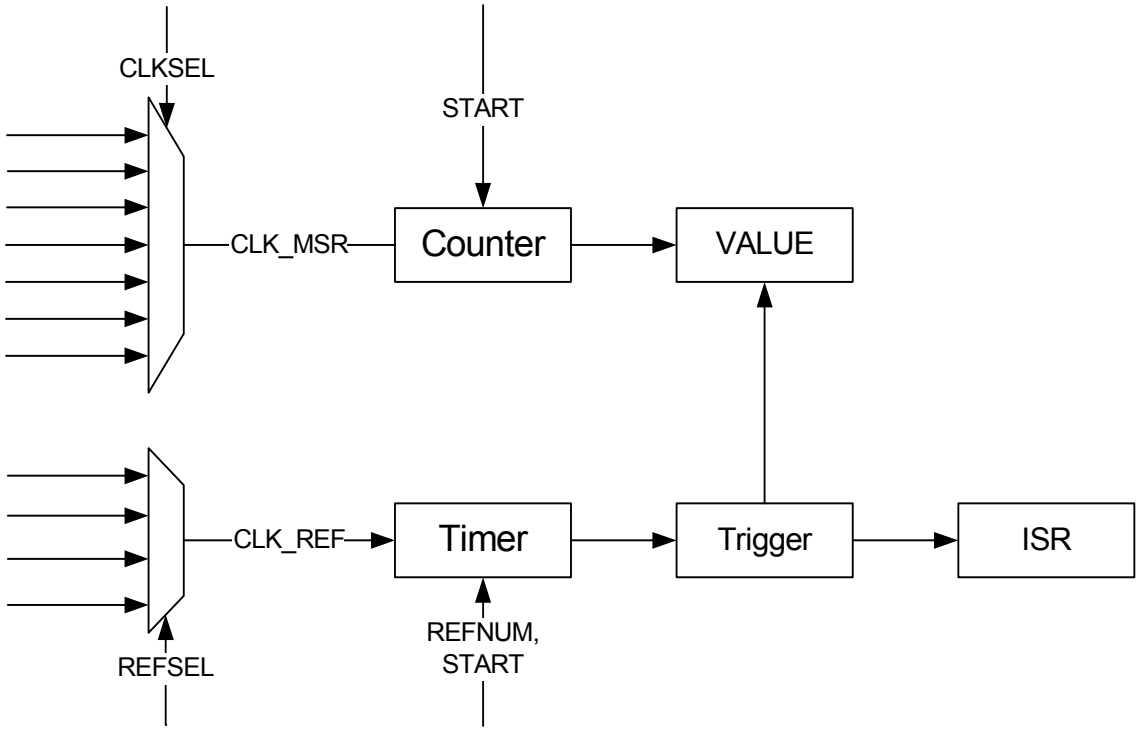
- **Accurately measures a clock frequency**
- **Selectable reference clock**
- **A selectable clock can be measured**
- **Ratio can be measured with 24-bit accuracy**

18.2 Overview

The Frequency Meter (FREQM) can be used to accurately measure the frequency of a clock by comparing it to a known reference clock.

18.3 Block Diagram

Figure 18-1. Frequency Meter Block Diagram



18.4 Product Dependencies

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

18.4.1 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.

18.4.2 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.

18.4.3 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.

18.4.4 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.

18.5 Functional Description

The FREQM accuratly measures the frequency of a clock by comparing the frequency to a known frequency:

$$f_{CLK_MSR} = (VALUE/REFNUM)*f_{CLK_REF}$$

18.5.1 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.

18.5.1.1 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.

18.5.2 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:

$$f_{CLK_MSR} = (VALUE/REFNUM)*f_{CLK_REF}$$

18.5.3 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.

18.6 User Interface

Table 18-1. 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.

18.6.1 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.

18.6.2 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**
  Setects 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.

18.6.3 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.

18.6.4 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.

18.6.5 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.

18.6.6 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.

18.6.7 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.

18.6.8 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.

18.6.9 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.

18.6.10 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.

18.7 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 18-2. Module Clock Name

Module Name	Clock Name	Description
FREQM	CLK_FREQM	Bus interface clock
	CLK_MSR	Measured clock
	CLK_REF	Reference clock

Table 18-3. Register Reset Values

Register	Reset Value
VERSION	0x00000301

Table 18-4. 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	OSC0	Output clock from Oscillator 0
5	OSC32K	Output clock from OSC32K
6	RCSYS	Output clock from RCSYS Oscillator
7	DFLL0	Output clock from DFLL0
8	Reserved	
9-14	GCLK0-5	Generic clock 0 through 5
15	RC120M AW clock	Output clock from RC120M to AW
16	RC120M	Output clock from RC120M to main clock mux
17	RC32K	Output clock from RC32K
18-31	Reserved	

Table 18-5. Clock Sources for CLK_REF

REFSEL	Clock/Oscillator	Description
0	RCSYS	System RC oscillator clock
1	OSC32K	Output clock form OSC32K
2-7	Reserved	

19. General-Purpose Input/Output Controller (GPIO)

Version: 2.1.1.5

19.1 Features

- **Each GPIO line features:**
  - **Configurable pin-change, rising-edge, or falling-edge interrupt**
  - **Configurable peripheral event generator**
  - **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**
  - **Optional locking of configuration to avoid accidental reconfiguration**

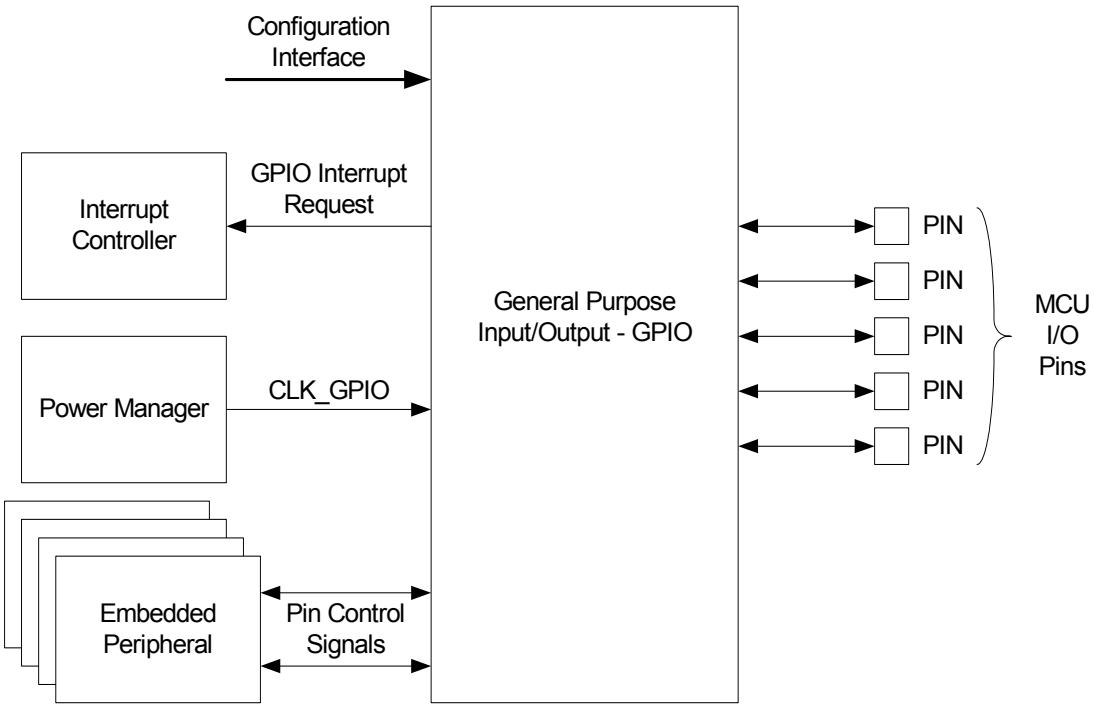
19.2 Overview

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 19.6.2.7 for details.

19.3 Block Diagram

Figure 19-1. GPIO Block Diagram



19.4 I/O Lines Description

Pin Name	Description	Type
GPIOn	GPIO pin n	Digital

19.5 Product Dependencies

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

19.5.1 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.

19.5.2 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.

19.5.3 Interrupts

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

19.5.4 Peripheral Events

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

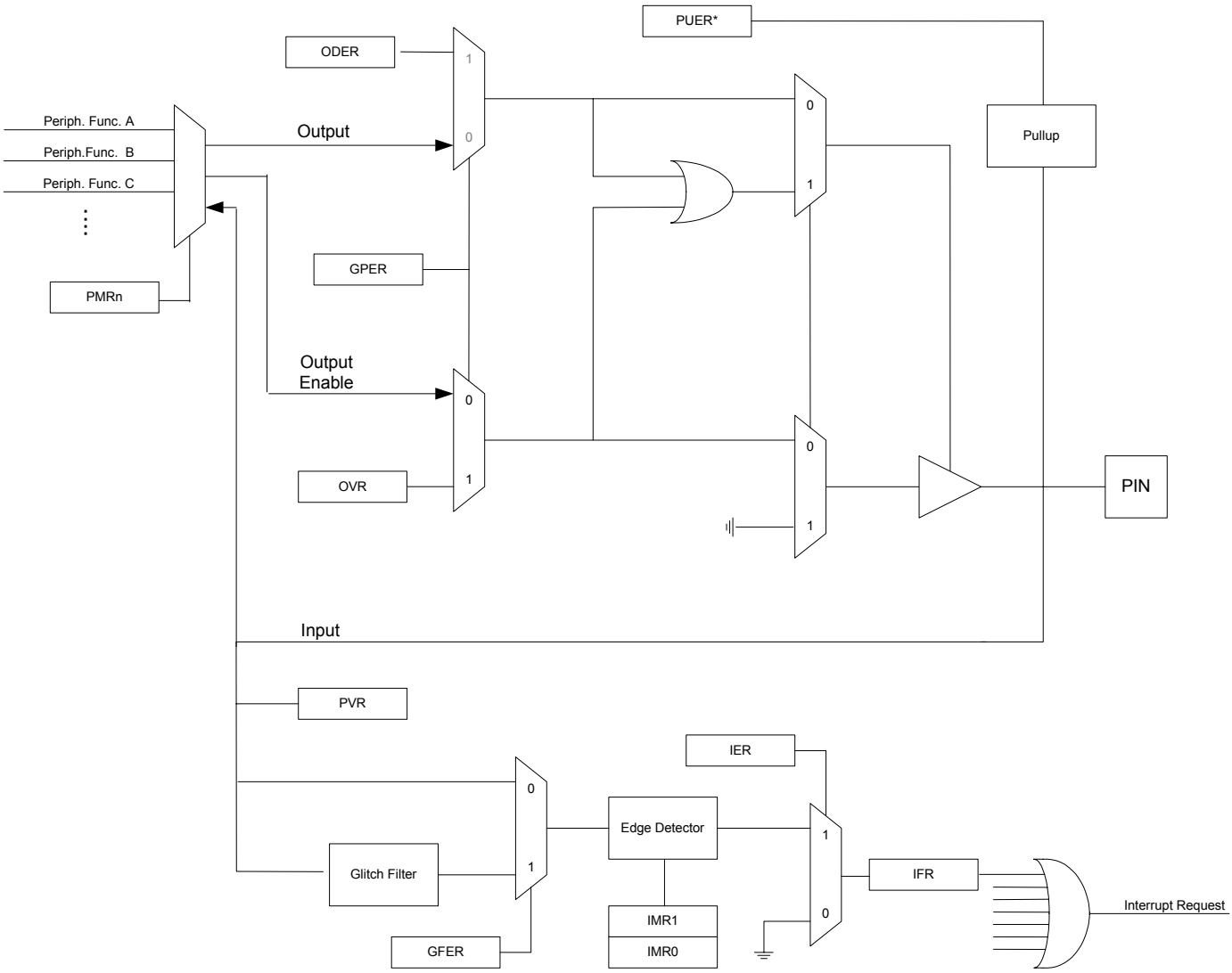
19.5.5 Debug Operation

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.

19.6 Functional Description

The GPIO controls the I/O pins of the microcontroller. The control logic associated with each pin is shown in Figure 1-2.

Figure 19-2. Overview of the GPIO



*) Register value is overrided if a peripheral function that support this function is enabled

19.6.1 Basic Operation

19.6.1.1 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 19.7.

19.6.1.2 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

19.6.1.3 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.

19.6.1.4 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).

19.6.1.5 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.

19.6.2 Advanced Operation

19.6.2.1 *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 19-1, 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 19-1. 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

19.6.2.2 *Pull-up Resistor Control*

Pull-up 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.

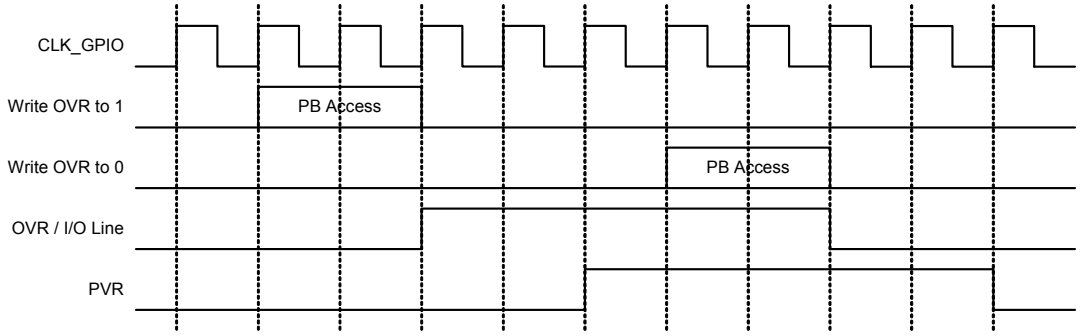
Pull-up is useful for detecting if a pin is unconnected or if a mechanical button is pressed, for various communication protocols and to keep unconnected pins from floating.

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).

19.6.2.3 *Output Pin Timings*

Figure 19-3 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 19-3. Output Pin Timings



19.6.2.4 *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.

19.6.2.5 *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.

19.6.2.6 *Interrupt Timings*

Figure 19-4 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 19-4. Interrupt Timing with Glitch Filter Disabled

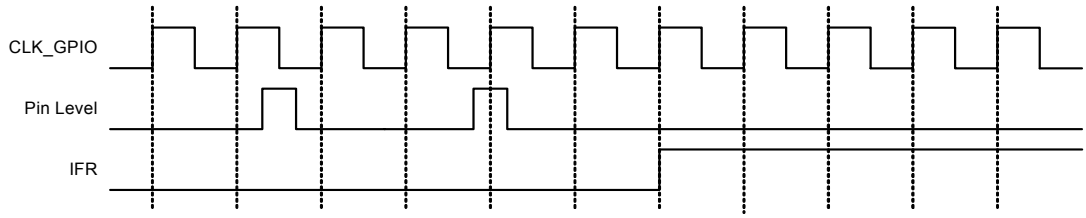
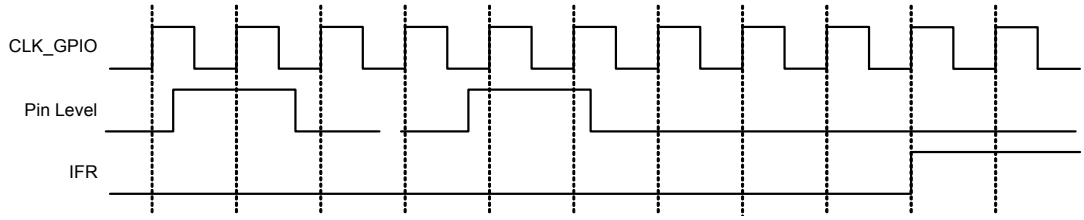


Figure 19-5 shows the timing for rising edge (or pin-change) interrupts when the glitch filter is enabled. For the pulse to be registered, it must be sampled on two subsequent rising edges. In the example, the first pulse is rejected while the second pulse is accepted and causes an interrupt request.

Figure 19-5. Interrupt Timing with Glitch Filter Enabled



19.6.2.7 *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 19.5.2 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.

19.6.2.8 *Peripheral Events*

Peripheral events allow direct peripheral to peripheral communication of specified events. See the Peripheral Event System chapter for more information.

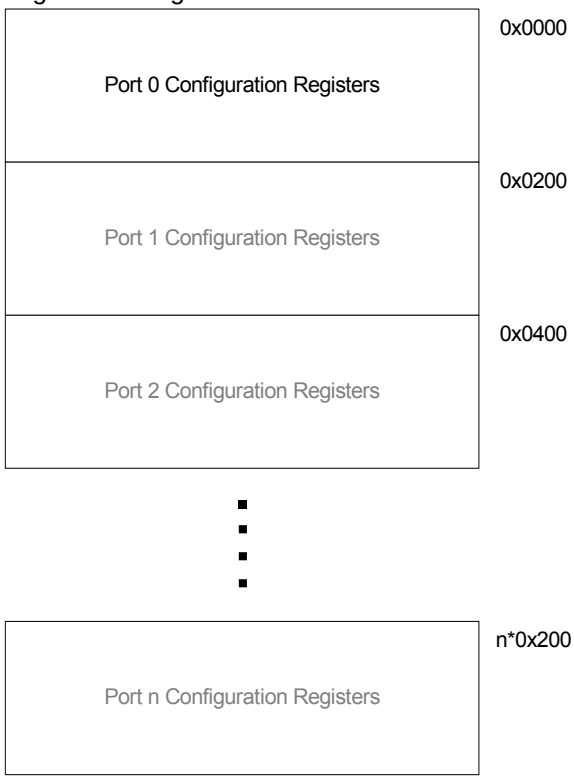
The GPIO can be programmed to output peripheral events whenever an interrupt condition is detected. The peripheral events configuration depends on the interrupt configuration. An event will be generated on the same condition as the interrupt (pin change, rising edge, or falling edge). The interrupt configuration is controlled by the IMR register. Peripheral event on a pin is enabled by writing a one to the corresponding bit in the Event Enable Register (EVER). The Peripheral Event trigger mode is shared with the interrupt trigger and is configured by writing to the IMR0 and IMR1 registers. Interrupt does not need to be enabled on a pin when peripheral events are enabled. Peripheral Events are also affected by the Input Glitch Filter settings. See Section 19.6.2.5 for more information.

A peripheral event can be generated on each GPIO pin. Each port can then have up to 32 peripheral event generators. Groups of eight peripheral event generators in each port are ORed together to form a peripheral event line, so that each port has four peripheral event lines connected to the Peripheral Event System.

19.7 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 19-6. 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 19-2 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.

19.7.1 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.

19.7.2 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 19.7.3, ensuring the LOCK mechanism itself is robust against software failure.

19.7.3 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 19-2. 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 19-2. 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 19-2. 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
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
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

Table 19-2. GPIO Register Memory Map

Offset	Register	Function	Register Name	Access	Reset	Config. Protection	Access Protection
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
0x180	Event Enable Register	Read	EVER	Read/Write	-(1)	N	N
0x184	Event Enable Register	Set	EVERS	Write-only		N	N
0x188	Event Enable Register	Clear	EVERC	Write-only		N	N
0x18C	Event Enable Register	Toggle	EVERT	Write-only		N	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.

19.7.4 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.

19.7.5 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**

19.7.6 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**

19.7.7 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

19.7.8 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.

19.7.9 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.

19.7.10 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.

19.7.11 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.

19.7.12 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.

19.7.13 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**

19.7.14 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

19.7.15 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.

19.7.16 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.

19.7.17 Event Enable Register

Name: EVER

Access: Read/Write, Set, Clear, Toggle

Offset: 0x180, 0x184, 0x188, 0x18C

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: Event Enable**
  0: Peripheral Event is disabled for the corresponding pin.
  1: Peripheral Event is enabled for the corresponding pin.

19.7.18 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.

19.7.19 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.

19.7.20 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.

19.7.21 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.

19.7.22 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.

19.8 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 19-3. Module Configuration

Feature	GPIO
Number of GPIO ports	2
Number of peripheral functions	8

Table 19-4. Implemented Pin Functions

Pin Function	Implemented	Notes
Pull-up	On all pins	Controlled by PUER or peripheral

Table 19-5. Module Clock Name

Module Name	Clock Name
GPIO	**CLK_GPIO**

The reset values for all GPIO registers are zero, with the following exceptions:

Table 19-6. Register Reset Values

Port	Register	Reset Value
0	GPER	0x0026FFAF
0	PUER	0x00000001
0	PARAMETER	0x0EFFFFFF
0	VERSION	0x00000211
1	GPER	0x00001FCF
1	PUER	0x00000000
1	PARAMETER	0x00001FFF
1	VERSION	0x00000211

20. Universal Synchronous Asynchronous Receiver Transmitter (USART)

Rev.4.4.0.5

20.1 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
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- 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

20.2 Overview

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (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, LIN and SPI buses and infrared transceivers. 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.

20.3 Block Diagram

Figure 20-1. USART Block Diagram

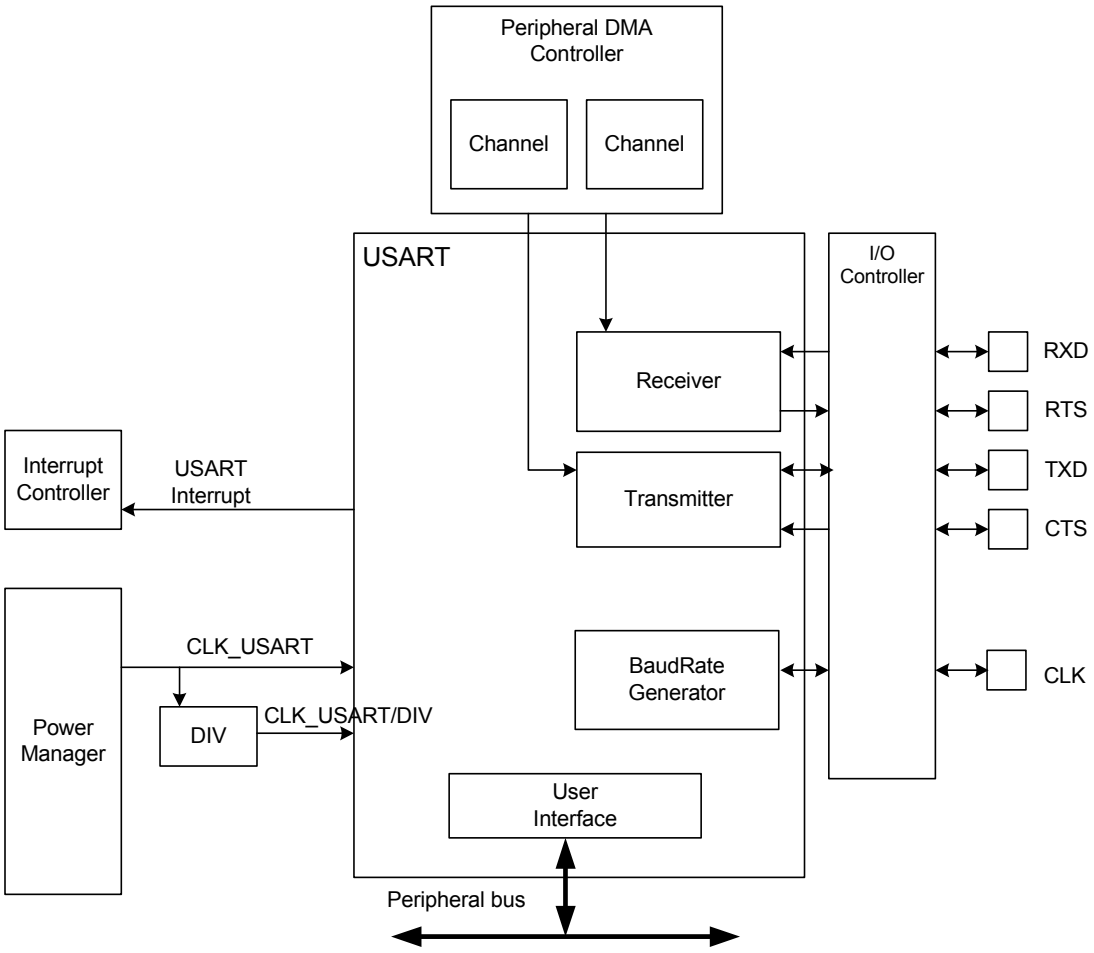


Table 20-1. SPI Operating Mode

PIN	USART	SPI Slave	SPI Master
RXD	RXD	MOSI	MISO

Table 20-1. SPI Operating Mode

PIN	USART	SPI Slave	SPI Master
TXD	TXD	MISO	MOSI
RTS	RTS	–	CS
CTS	CTS	CS	–

20.4 I/O Lines Description

Table 20-2. 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	
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

20.5 Product Dependencies

20.5.1 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.

20.5.2 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.

20.5.3 Interrupts

The USART interrupt request line is connected to the interrupt controller. Using the USART interrupt requires the interrupt controller to be programmed first.

20.6 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 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 break management
  - Optional multidrop serial communication
- 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

20.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

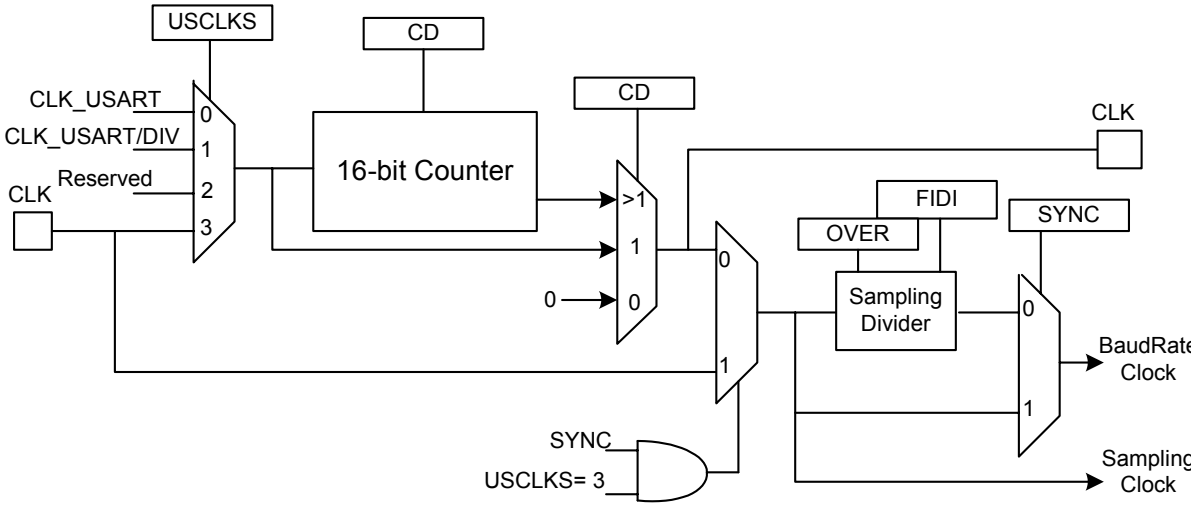
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (MR) between:

- 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 20-2. Baud Rate Generator



20.6.1.1 *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.

20.6.1.2 Baud Rate Calculation Example

Table 20-3 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 20-3. 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%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

20.6.1.3 Fractional Baud Rate in Asynchronous Mode

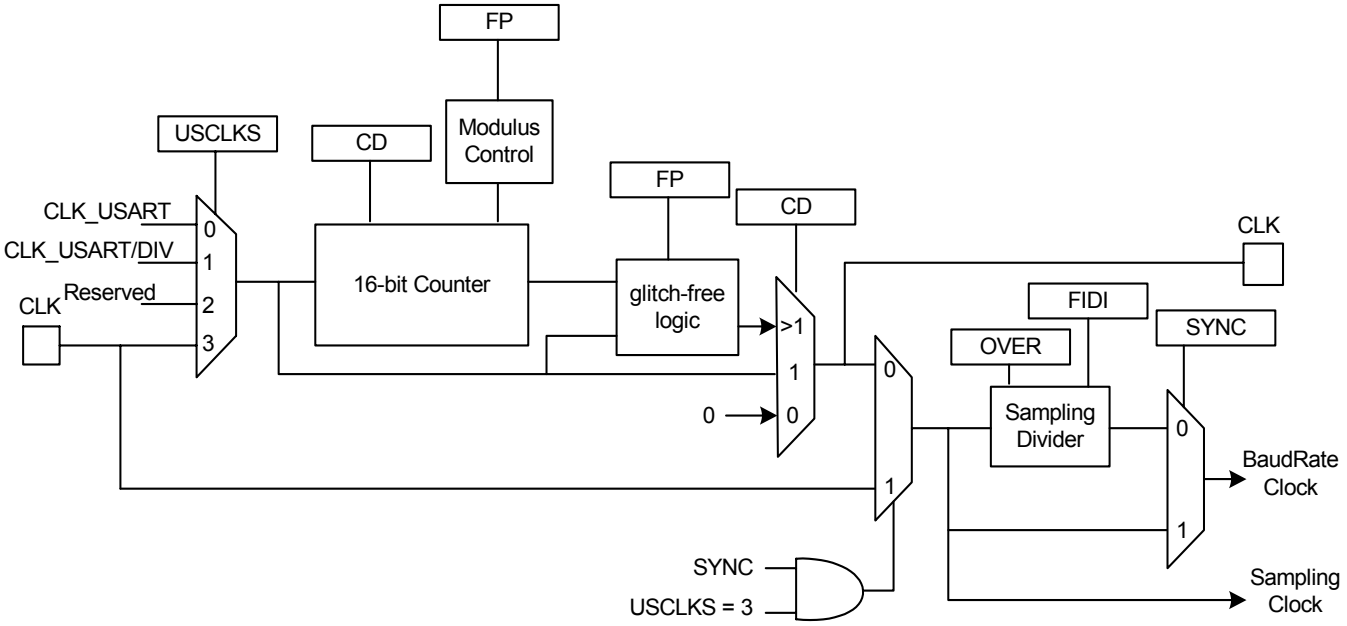
The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified 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 20-3. Fractional Baud Rate Generator



20.6.1.4 *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.

20.6.2 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.

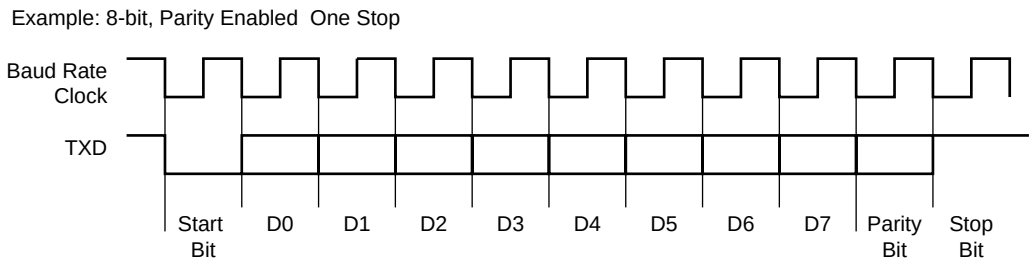
20.6.3 Synchronous and Asynchronous Modes

20.6.3.1 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 20-4. Character Transmit

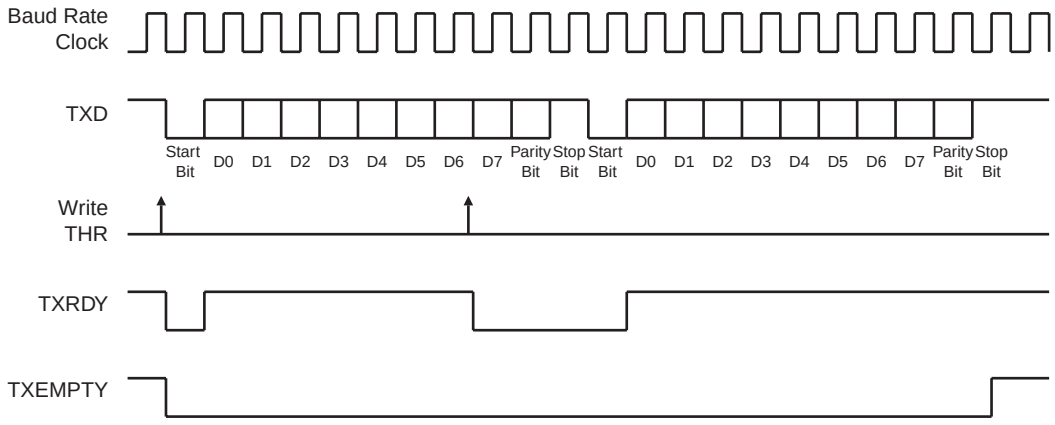


The characters are sent by writing in the Transmit Holding Register (THR). The transmitter reports two status bits in the Channel Status Register (CSR): TXRDY (Transmitter Ready), which indicates that THR is empty and TXEMPTY, which indicates that all the characters written

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 20-5. Transmitter Status



20.6.3.2 Asynchronous Receiver

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver over-samples 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 20-6 and Figure 20-7 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 20-6. Asynchronous Start Detection

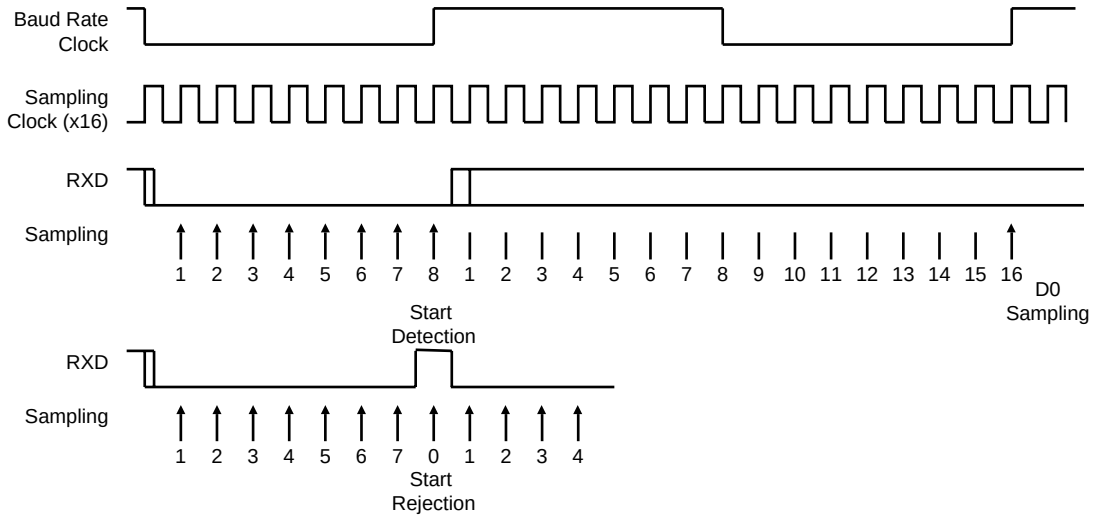
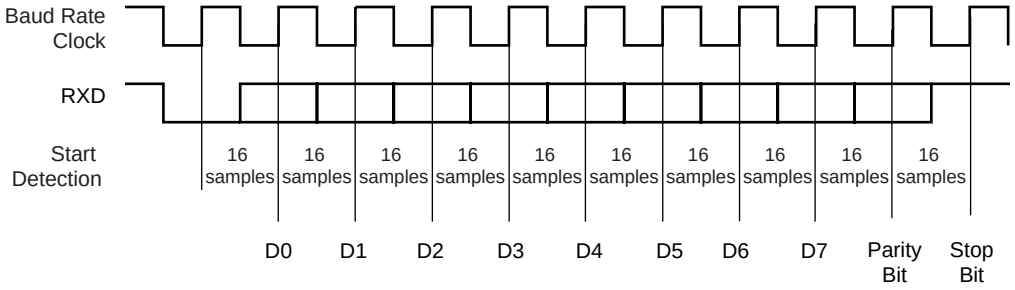


Figure 20-7. Asynchronous Character Reception

Example: 8-bit, Parity Enabled



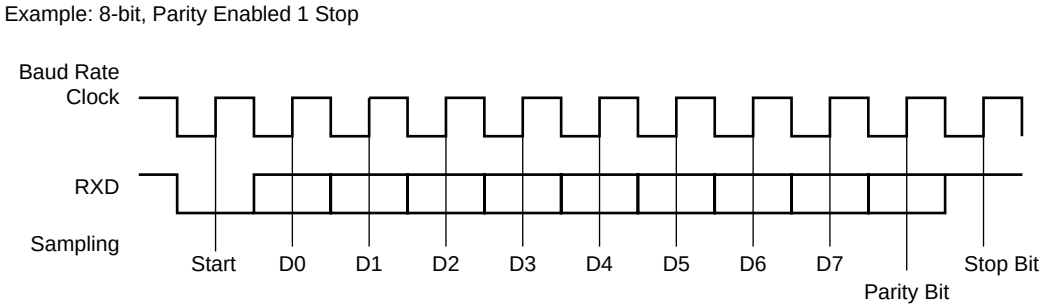
20.6.3.3 Synchronous Receiver

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 20-8 illustrates a character reception in synchronous mode.

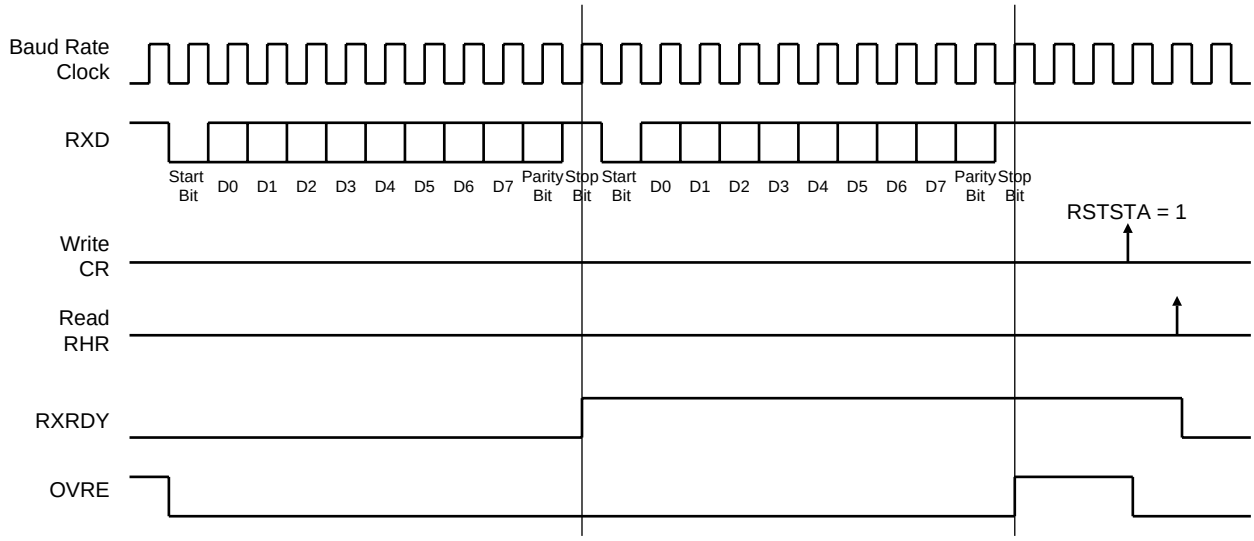
Figure 20-8. Synchronous Mode Character Reception



20.6.3.4 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 20-9. Receiver Status



20.6.3.5 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 376. 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.

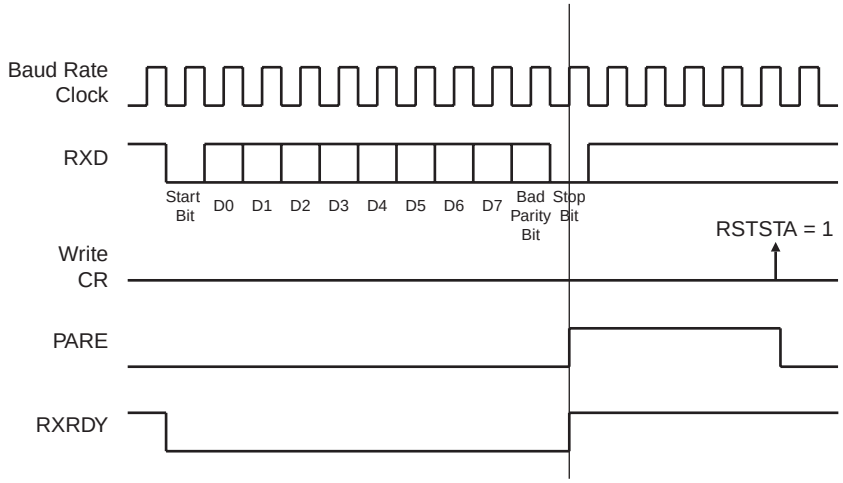
Table 20-4 shows an example of the parity bit for the character 0x41 (character ASCII "A") depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 20-4. 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 20-10 illustrates the parity bit status setting and clearing.

Figure 20-10. Parity Error



20.6.3.6 *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.

20.6.3.7 *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 20-11, 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 20-11. Timeguard Operations

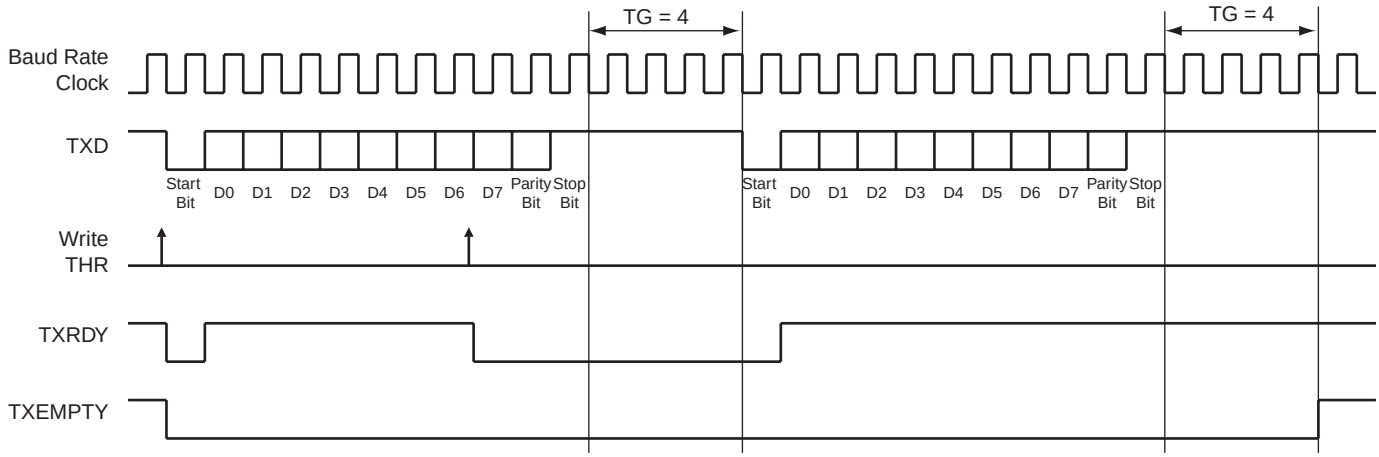


Table 20-5 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 20-5. 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

20.6.3.8 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 20-12 shows the block diagram of the Receiver Time-out feature.

Figure 20-12. Receiver Time-out Block Diagram

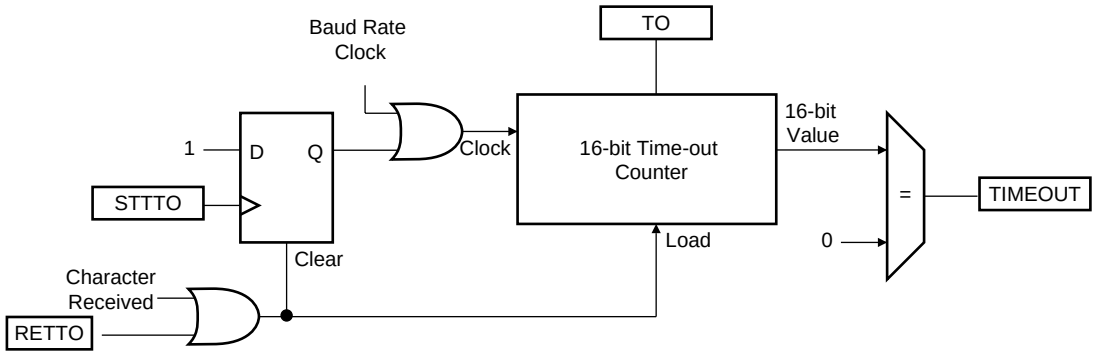


Table 20-6 gives the maximum time-out period for some standard baud rates.

Table 20-6. 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 20-6. Maximum Time-out Period (Continued)

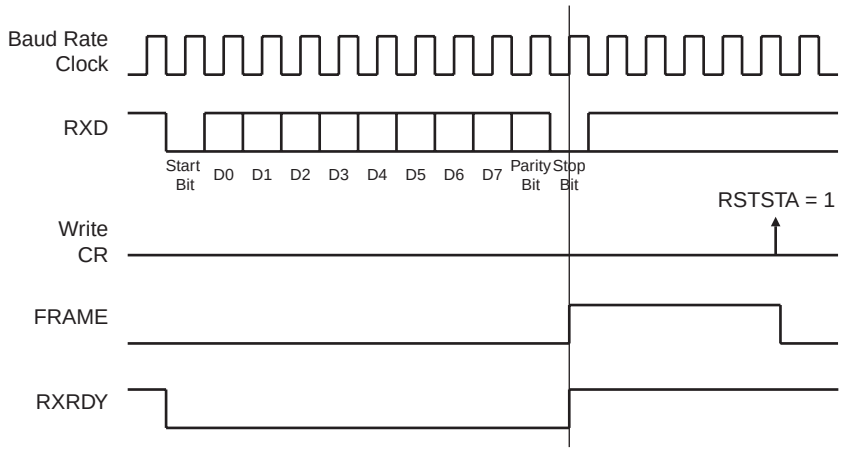
Baud Rate	Bit Time	Time-out
56000	18	1 170
57600	17	1 138
200000	5	328

20.6.3.9 *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 20-13. Framing Error Status



20.6.3.10 *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.

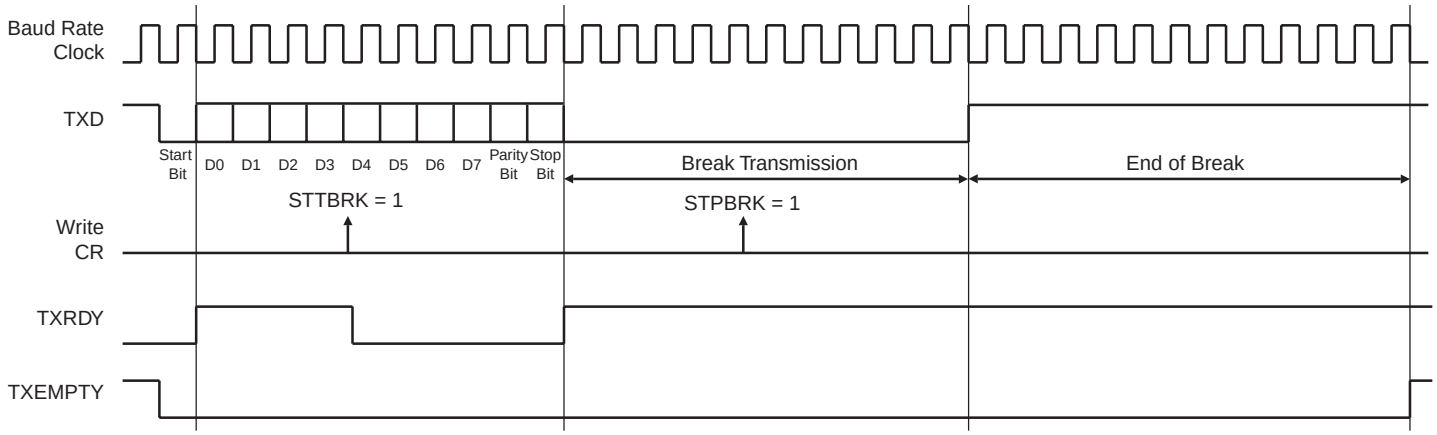
Writing CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

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 20-14 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 20-14. Break Transmission



20.6.3.11 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.

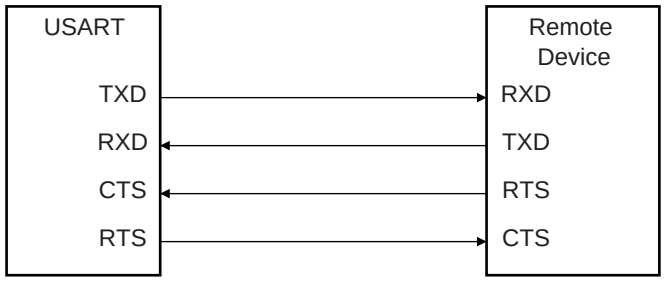
When the low stop bit is detected, the receiver asserts the RXBRK bit in CSR. This bit may be cleared by writing the Control Register (CR) with the bit RSTSTA at 1.

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.

20.6.3.12 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 20-15.

Figure 20-15. 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 20-16 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 20-16. Receiver Behavior when Operating with Hardware Handshaking

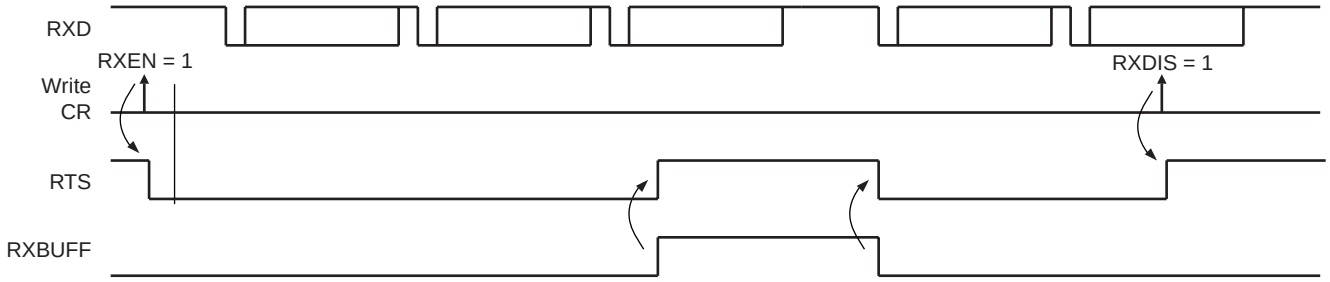
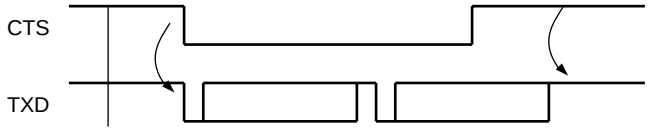


Figure 20-17 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 20-17. Transmitter Behavior when Operating with Hardware Handshaking



20.6.4 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.

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.

20.6.4.1 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).

20.6.4.2 Baud Rate

In SPI Mode, the baudrate generator operates in the same way as in USART synchronous mode: See Section “20.6.1.4” on page 370. 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.

20.6.4.3 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 20-7. SPI Bus Protocol Mode

SPI Bus Protocol Mode	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 20-18. SPI Transfer Format (CPHA=1, 8 bits per transfer)

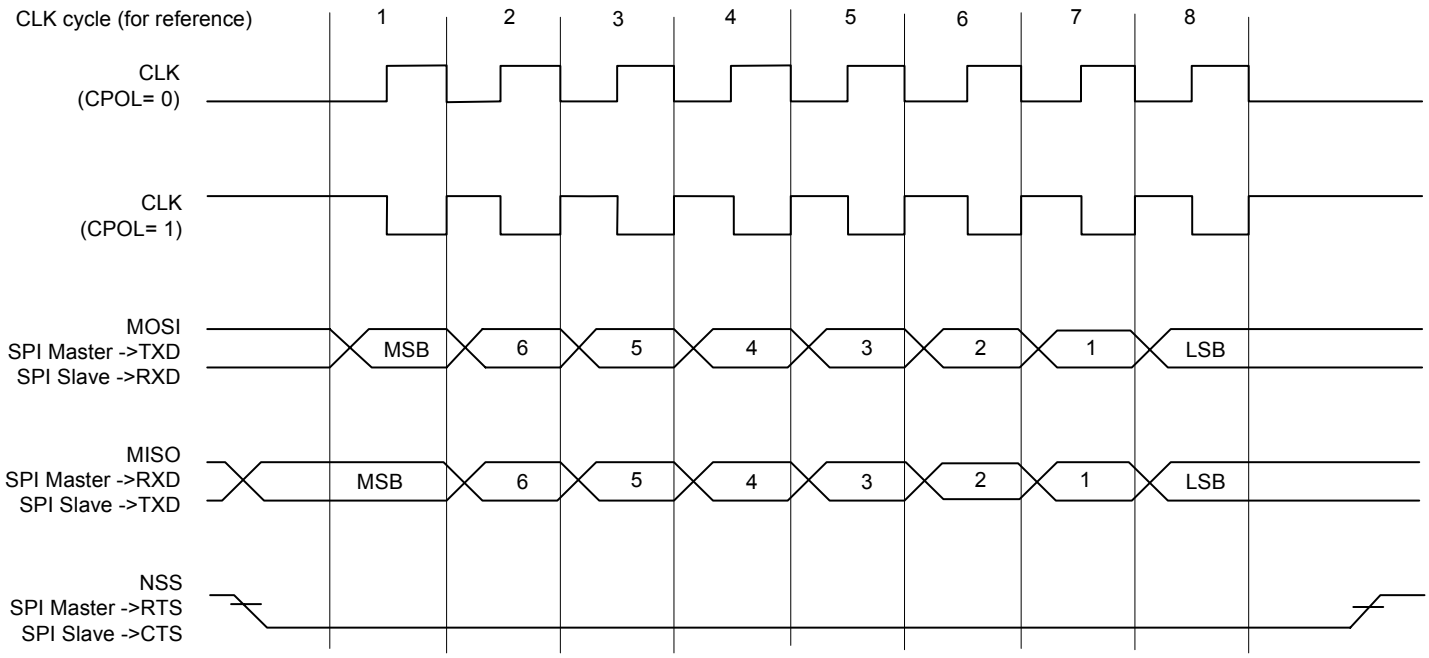
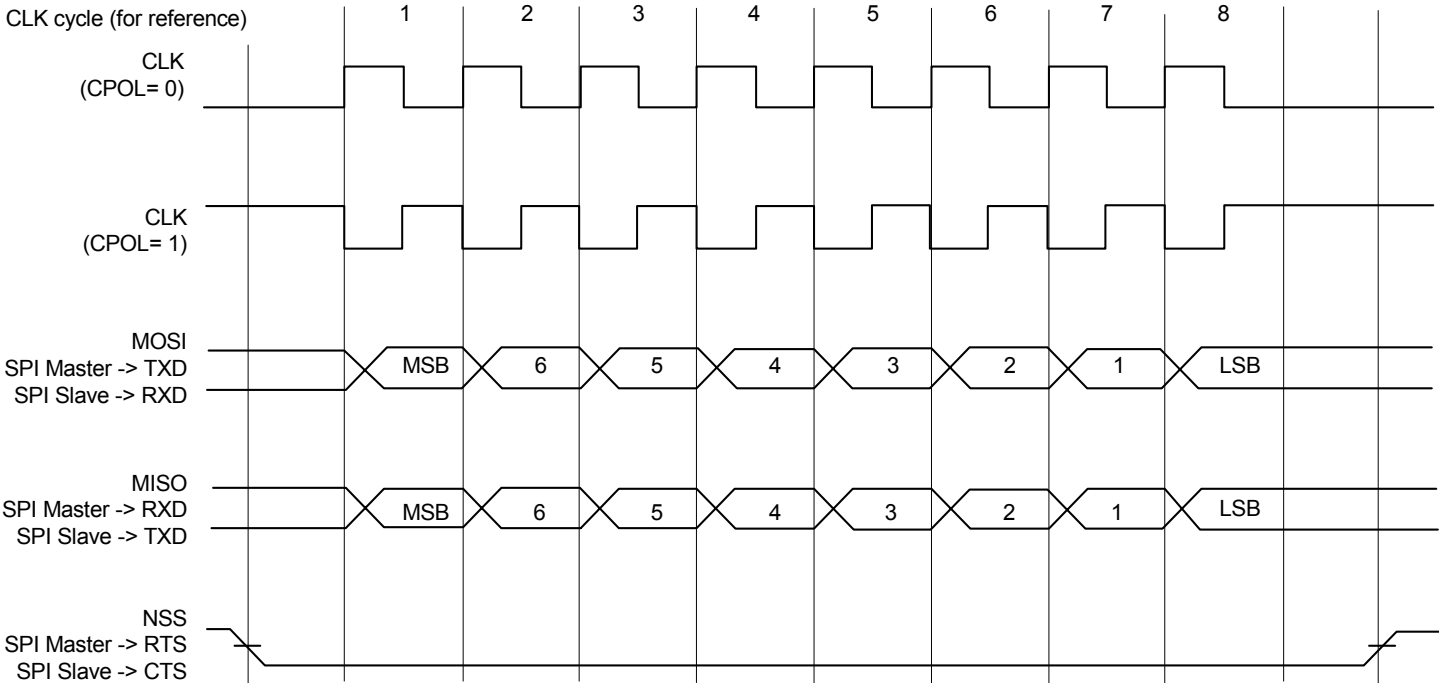


Figure 20-19. SPI Transfer Format (CPHA=0, 8 bits per transfer)



20.6.4.4 Receiver and Transmitter Control

See Section “20.6.2” on page 371.

20.6.4.5 Character Transmission

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.

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.

20.6.4.6 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.

20.6.4.7 Receiver Timeout

Because the receiver baudrate clock is active only during data transfers in SPI Mode, a receiver timeout is impossible in this mode, whatever the Time-out value is (field TO) in the Time-out Register (RTOR).

20.6.5 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.

20.6.5.1 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 20.6.5.2)

20.6.5.2 Receiver and Transmitter Control

See Section “20.6.2” on page 371.

20.6.5.3 Character Transmission

See Section “20.6.3.1” on page 371.

20.6.5.4 Character Reception

See Section “20.6.3.4” on page 374.

20.6.5.5 *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.

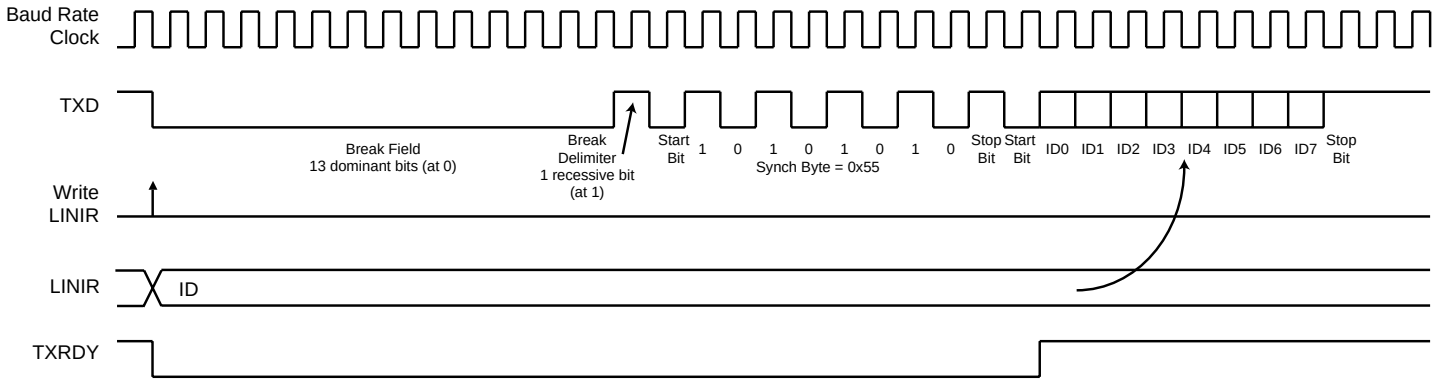
The header is transmitted as soon as the identifier is written in the LIN Identifier register (LINIR). At this moment the flag TXRDY falls.

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 20.6.5.8).

The flag TXRDY rises when the identifier character is transferred into the Shift Register of the transmitter.

Figure 20-20. Header Transmission



20.6.5.6 *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.

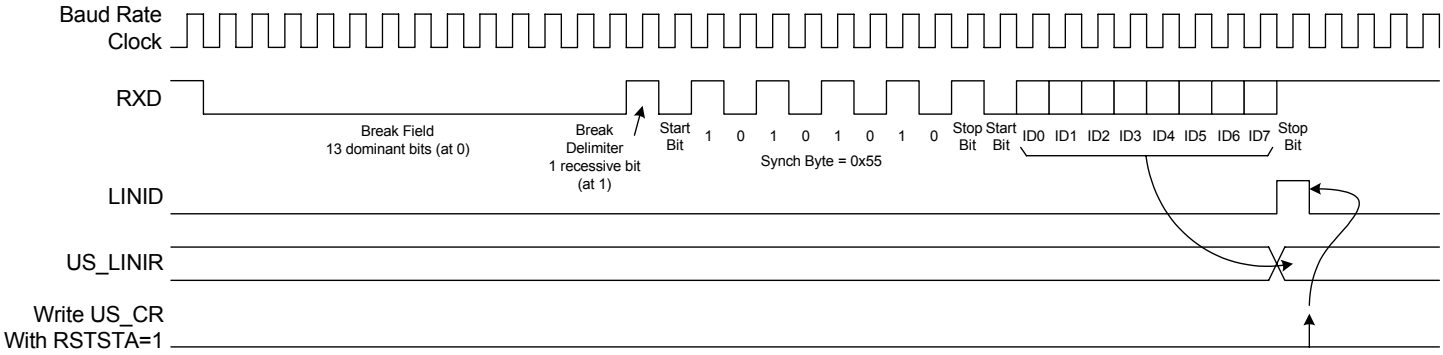
The USART uses a break detection threshold of 11 nominal bit times at the actual baud rate. At any time, if 11 consecutive recessive bits are detected on the bus, the USART detects a Break Field. As long as a Break Field has not been detected, the USART stays idle and the received data are not taken 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 20.6.5.7). If the received Synch character is not 0x55, an Inconsistent Synch Field error is generated (see Section 20.6.6).

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 20.6.5.8).

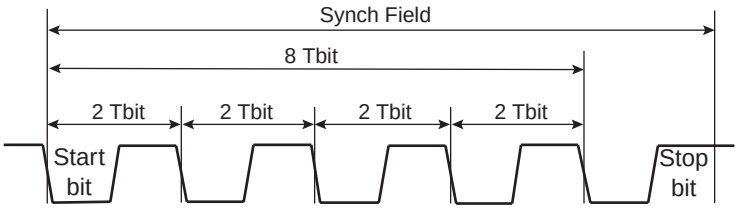
Figure 20-21. Header Reception



20.6.5.7 *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 20-22. Synch Field

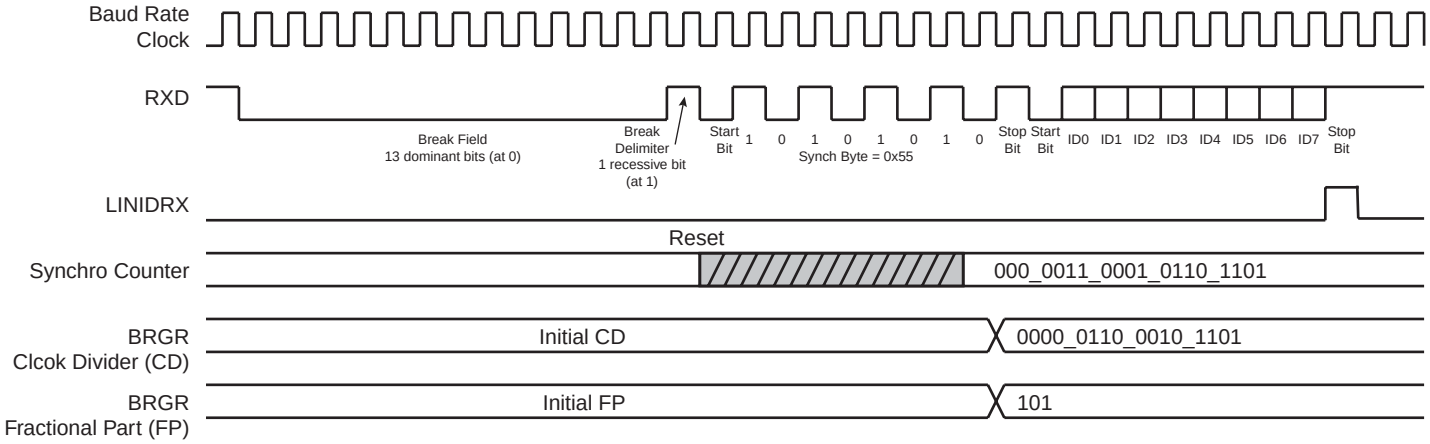


The time measurement is made by a 19-bit counter clocked by the sampling clock (see Section 20.6.1).

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 (CD) and the 3 least significant bits of this value (the remainder) gives the new fractional part (FP).

When the Synch Field has been received, the clock divider (CD) and the fractional part (FP) are updated in the Baud Rate Generator register (BRGR).

Figure 20-23. 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).

$$\text{Baudrate_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{Over}) + \beta] \times \text{Baudrate}}{8 \times F_{SLAVE}}\right)\%$$

$$\text{Baudrate_deviation} = \left(100 \times \frac{[\alpha \times 8 \times (2 - \text{Over}) + \beta] \times \text{Baudrate}}{8 \times \left(\frac{F_{TOL_UNSYNCH}}{100}\right) \text{x} F_{Nom}}\right)\%$$

$$-0{,}5 \le \alpha \le +0{,}5 \quad -1 < \beta < +1$$

$F_{TOL_UNSYNCH}$ 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:

$$F_{NOM}(\min) = \left(100 \times \frac{[0{,}5 \times 8 \times (2 - \text{Over}) + 1] \times \text{Baudrate}}{8 \times \left(\frac{-15}{100} + 1\right) \times 1\%} \right) \text{Hz}$$

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:

$$-4 \le \alpha \le +4 \qquad -1 < \beta < +1$$

It follows from that, a minimum value for the nominal clock frequency:

$$F_{NOM}(\min) = \left(100 \times \frac{[4 \times 8 \times (2 - \text{Over}) + 1] \times \text{Baudrate}}{8 \times \left(\frac{-15}{100} + 1\right) \times 1\%} \right) \text{Hz}$$

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

20.6.5.8 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 20.6.3.5). 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.

20.6.5.9 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 20.7.12).

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

20.6.5.10 Response Data Length

The LIN response data length is the number of data fields (bytes) of the response excluding the checksum.

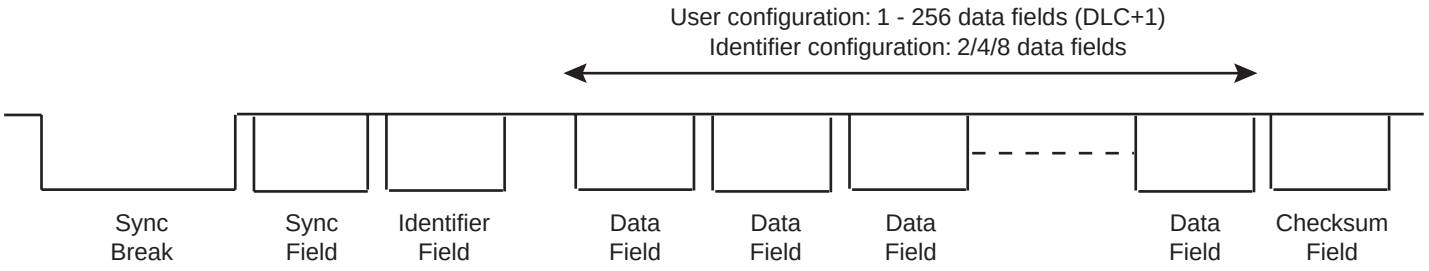
The response data length can either be configured by the user or be defined automatically by bits 4 and 5 of the Identifier (compatibility to LIN Specification 1.1). The user can choose between these two modes by the DLM bit of the LIN Mode register (LINMR):

- 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 20-8. 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 20-24. Response Data Length



20.6.5.11 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 20.6.5.10).

20.6.5.12 *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):

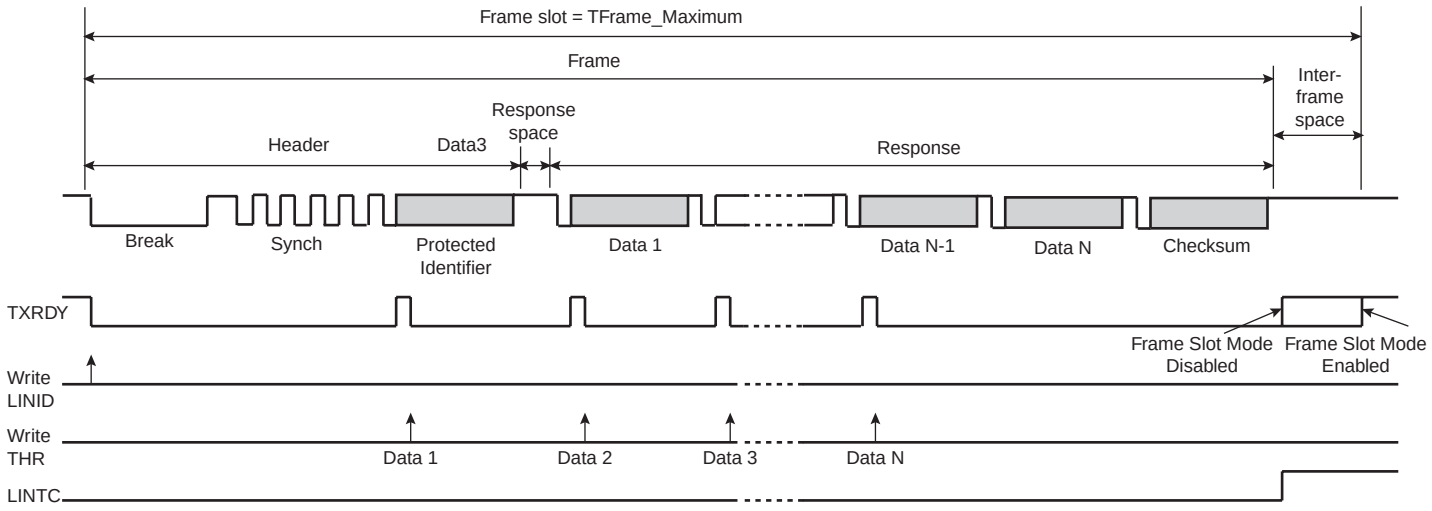
- THeader_Nominal = 34 x TBit
- TResponse_Nominal = 10 x (NData + 1) x TBit
- TFrame_Maximum = 1.4 x (THeader_Nominal + TResponse_Nominal + 1)(Note:)
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1 + 1) + 1) x TBIT
- TFrame_Maximum = (77 + 14 x DLC) x TBIT

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 20-25. Frame Slot Mode



20.6.6 LIN Errors

20.6.6.1 Bit Error

This error is generated when the 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.

20.6.6.2 Inconsistent Synch Field Error

This error is generated in Slave node configuration if the Synch Field character received is other than 0x55.

20.6.6.3 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).

20.6.6.4 Checksum Error

This error is set if the received checksum is wrong. This error can be generated only if the checksum feature is enabled (CHKDIS = 0).

20.6.6.5 Slave Not Responding Error

This error is set 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 20.6.5.12). This error is disabled if the USART does not expect any message (NACT = PUBLISH or NACT = IGNORE).

20.6.7 LIN Frame Handling

20.6.7.1 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 20-26. Master Node Configuration, NACT = PUBLISH

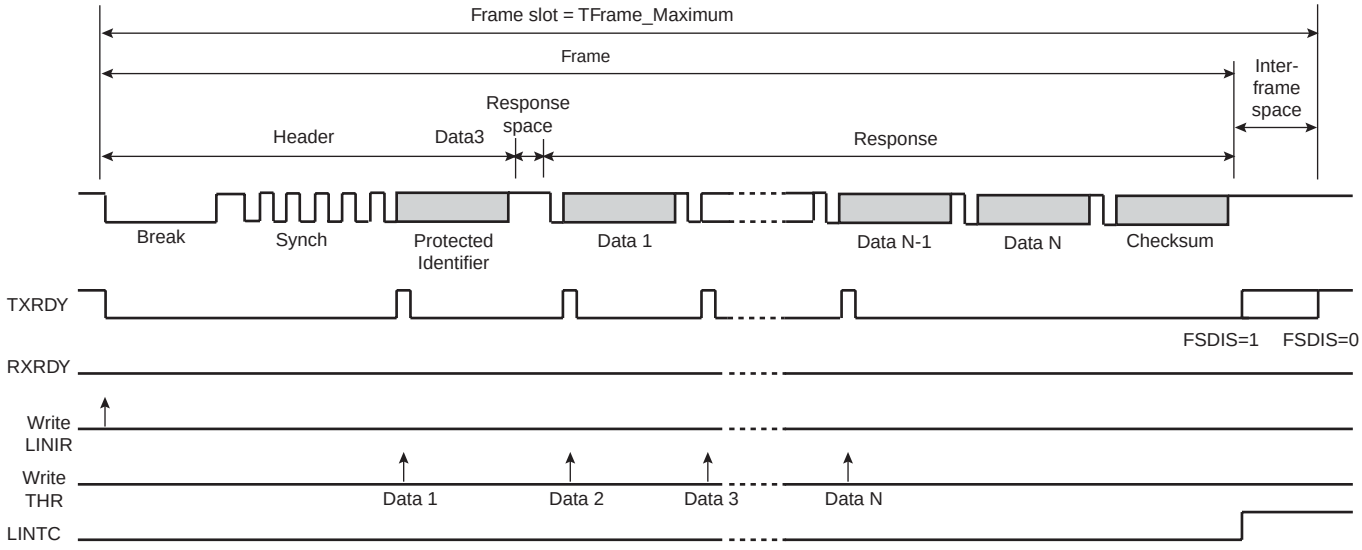


Figure 20-27. Master Node Configuration, NACT=SUBSCRIBE

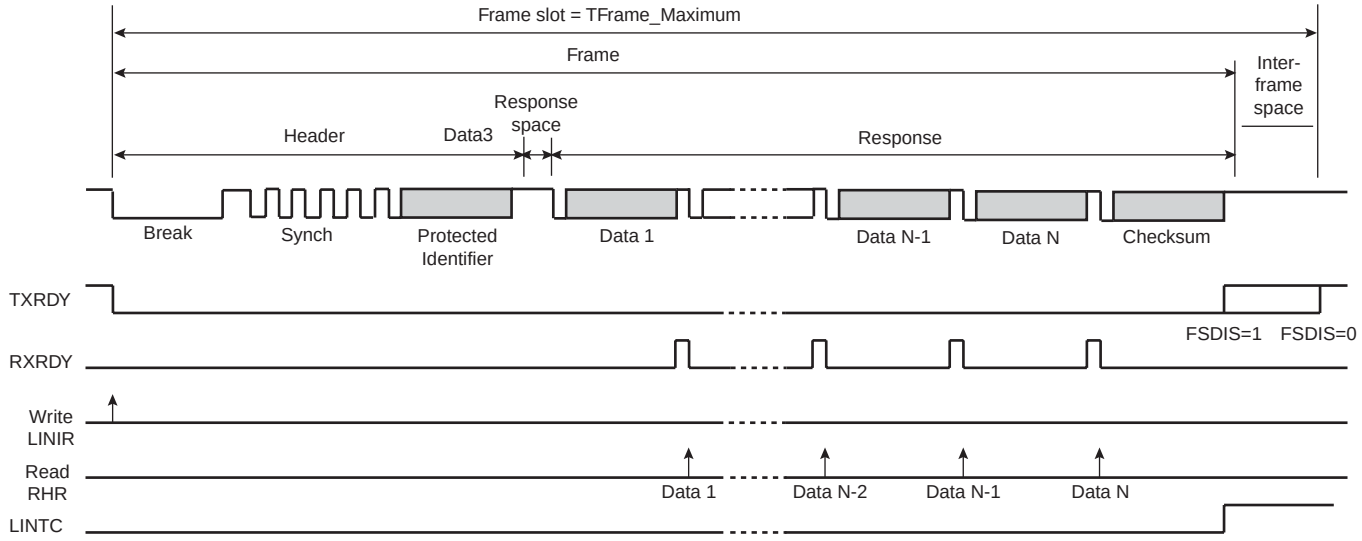
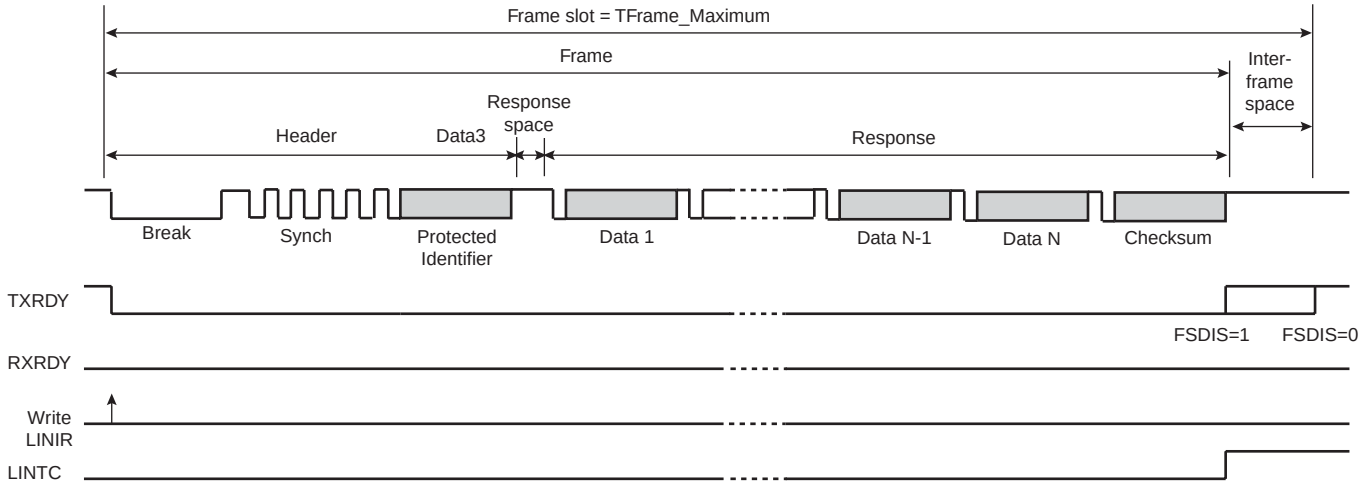


Figure 20-28. Master Node Configuration, NACT=IGNORE



20.6.7.2 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 20-29. Slave Node Configuration, NACT = PUBLISH

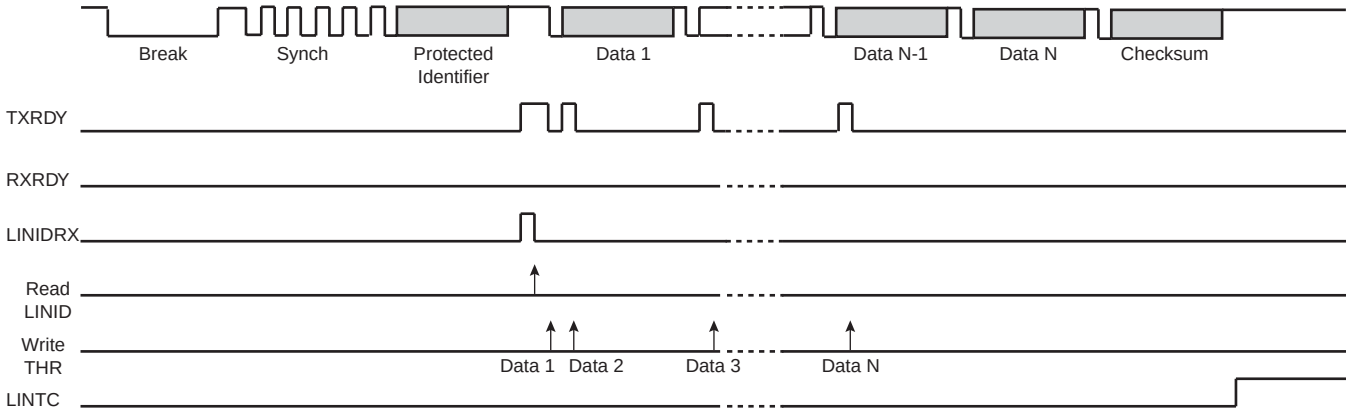


Figure 20-30. Slave Node Configuration, NACT = SUBSCRIBE

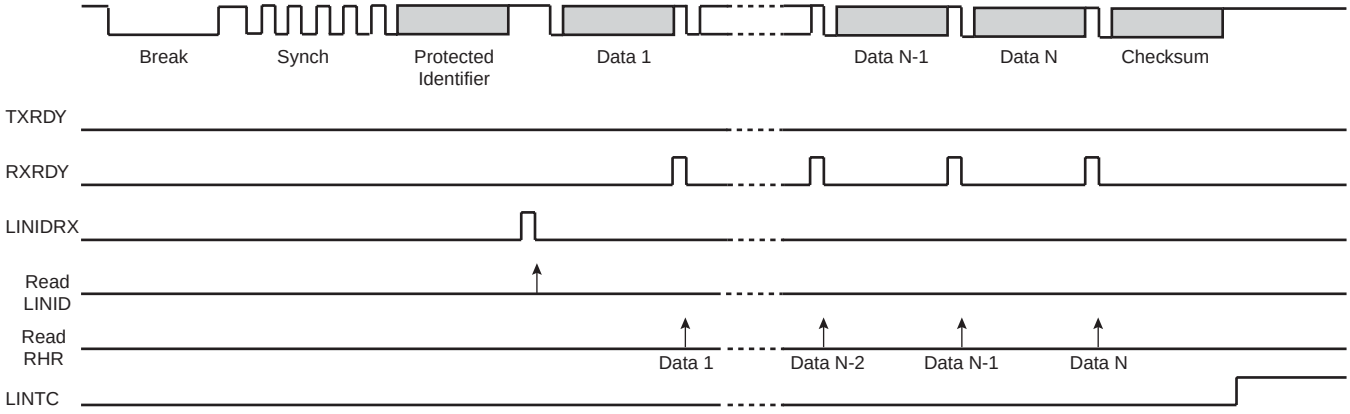
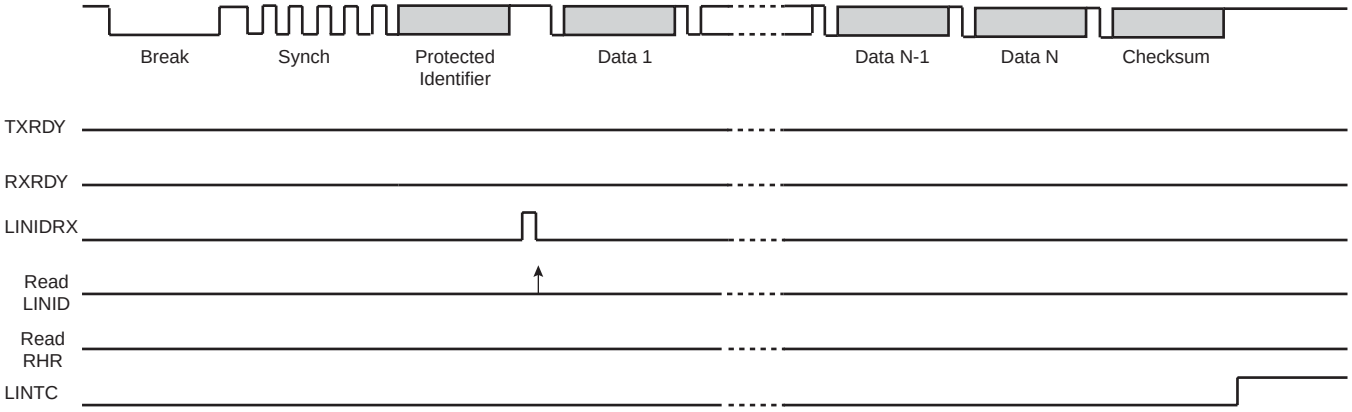


Figure 20-31. Slave Node Configuration, NACT = IGNORE



20.6.8 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.

20.6.8.1 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 20-32. Master Node with Peripheral DMA Controller (PDCM=1)

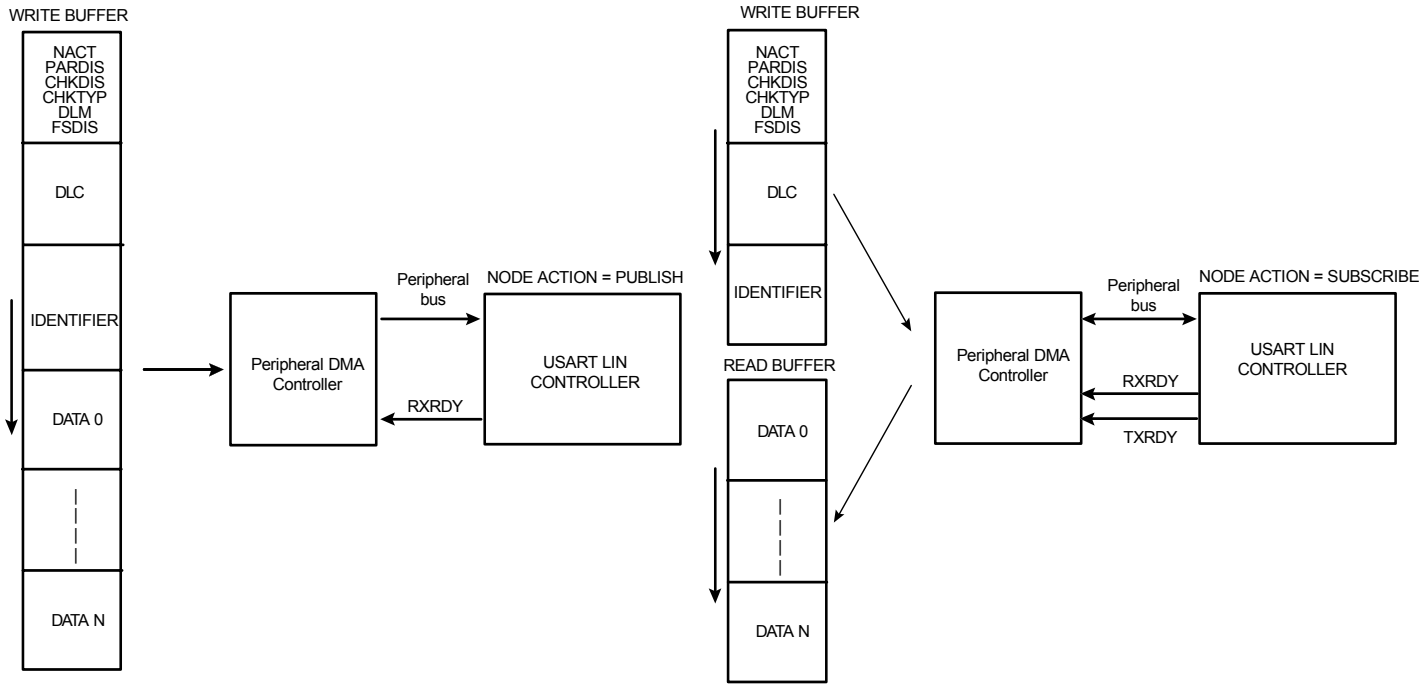
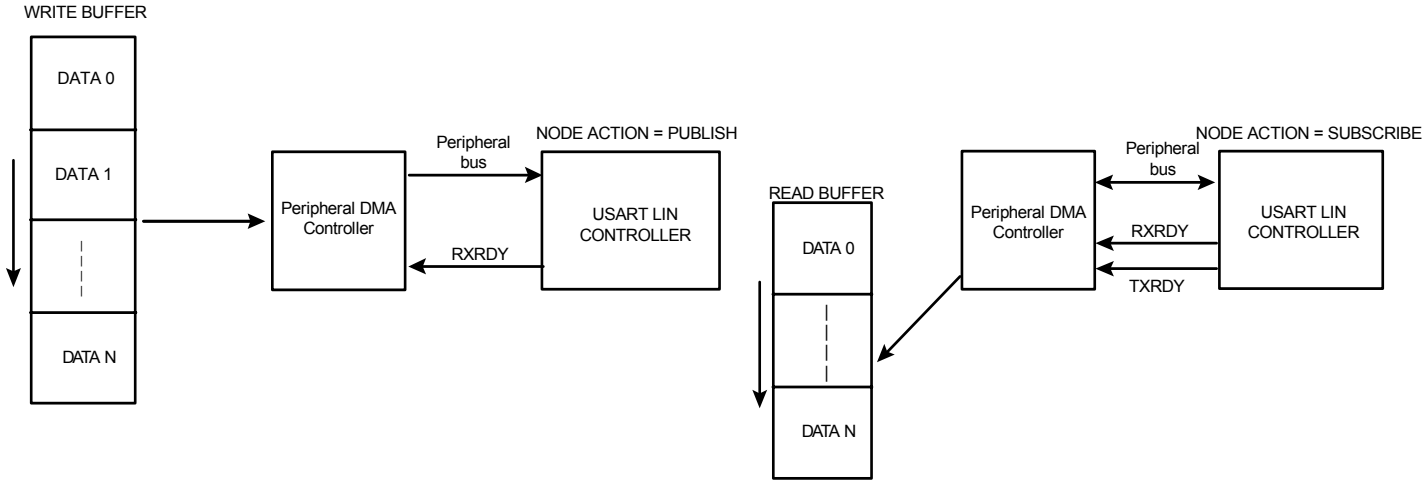


Figure 20-33. Master Node with Peripheral DMA Controller (PDCM=0)



20.6.8.2 Slave Node Configuration

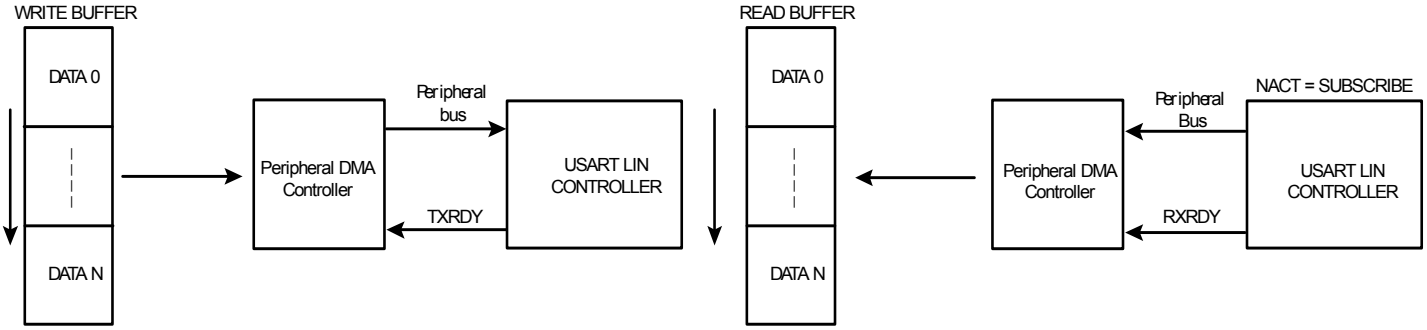
In this configuration, the Peripheral DMA Controller transfers only the DATA. The Identifier must be read by the user in the LIN Identifier register (LINIR). The LIN mode must be written by the user in the LIN Mode register (LINMR).

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 20-34. Slave Node with Peripheral DMA Controller



20.6.9 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.

20.6.10 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 20-9. 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

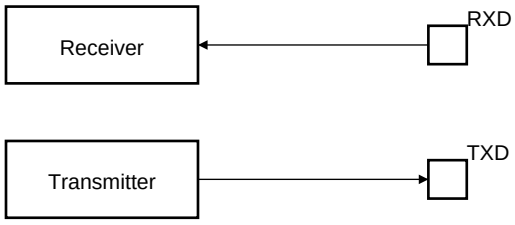
20.6.11 Test Modes

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.

20.6.11.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

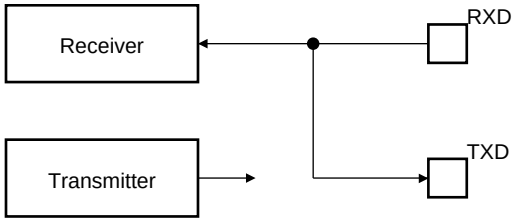
Figure 20-35. Normal Mode Configuration



20.6.11.2 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 20-36. 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.

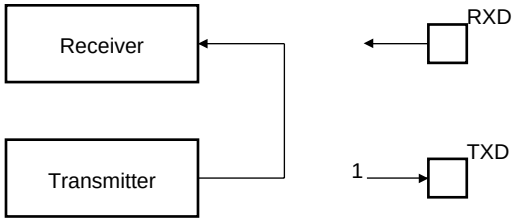
Figure 20-36. Automatic Echo Mode Configuration



20.6.11.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 20-37. 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.

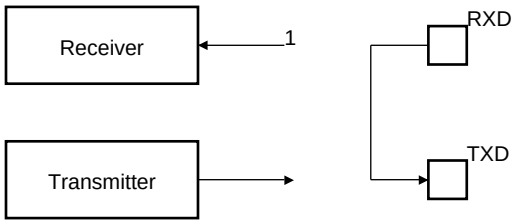
Figure 20-37. Local Loopback Mode Configuration



20.6.11.4 *Remote Loopback Mode*

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 20-38. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 20-38. Remote Loopback Mode Configuration



20.6.12 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 411
- "Baud Rate Generator Register" on page 421
- "Receiver Time-out Register" on page 422
- "Transmitter Timeguard Register" on page 423
- "" on page 424
- "" on page 424
- "Manchester Configuration Register" on page 84

20.7 User Interface

Table 20-10. 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
0x0054	LIN Mode Register	LINMR	Read-write	0x00000000
0x0058	LIN Identifier Register	LINIR	Read-write	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.

20.7.1 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	–	–
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).
- **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.

- **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, 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.

20.7.2 Mode Register

Name: MR

Access Type: Read-write

Offset: 0x4

Reset Value: -

31	30	29	28	27	26	25	24
–	–	–	FILTER	–			

23	22	21	20	19	18	17	16
–	–	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			

- **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).
- **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.
- **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 20-11.

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 20-12.

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 20-13.

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 20-14.

CHRL		Character Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

- **USCLKS: Clock Selection**

Table 20-15.

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 20-16.

MODE				Mode of the USART
0	0	0	0	Normal
0	0	1	0	Hardware Handshaking
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

20.7.3 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x8

Reset Value: -

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
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.

20.7.4 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0xC

Reset Value: -

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
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.

20.7.5 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x10

Reset Value: -

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
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.

20.7.6 Channel Status Register

Name: CSR

Access Type: Read-only

Offset: 0x14

Reset Value: -

31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
CTS	–	–	–	CTSIC	–	–	–
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

- **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.
- **CTS: Image of CTS Input**
  - 0: CTS is at 0.
  - 1: CTS 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.
- **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.

20.7.7 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.

20.7.8 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.

20.7.9 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 20-17.

	SYNC = 0		**SYNC = 1 or MODE = SPI (Master or Slave)**	
CD	**OVER = 0**	**OVER = 1**		
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	

20.7.10 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.

20.7.11 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.

20.7.12 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
–	–	–	–	–	–	–	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	

- **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

20.7.13 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.

20.7.14 Write Protect Mode Register

Register Name: WPMR

Access Type: Read-write

Offset: 0xE4

Reset Value: See Table 20-10

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 411
- "Baud Rate Generator Register" on page 421
- "Receiver Time-out Register" on page 422
- "Transmitter Timeguard Register" on page 423
- "" on page 424
- "" on page 424
- "Manchester Configuration Register" on page 84

20.7.15 Write Protect Status Register

Register Name: WPSR

Access Type: Read-only

Offset: 0xE8

Reset Value: See Table 20-10

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.

20.7.16 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			

7	6	5	4	3	2	1	0
VERSION							

- **VARIANT**

  Reserved. No functionality associated.

- **VERSION**

  Version of the module. No functionality associated.

20.8 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. Please refer to the Power Manager chapter for details.

Table 20-18. USART Configuration

Feature	USART0	USART1	USART2	USART3
Receiver Time-out Counter Size (Size of the RTOR.TO field)	17 bit	17 bit	17 bit	17 bit
DIV Value for divided CLK_USART	8	8	8	8

Table 20-19. USART Clocks

Module Name	Clock Name	Description
USART0	CLK_USART0	Clock for the USART0 bus interface
USART1	CLK_USART1	Clock for the USART1 bus interface
USART2	CLK_USART2	Clock for the USART1 bus interface
USART3	CLK_USART3	Clock for the USART1 bus interface

Table 20-20. Register Reset Values

Register	Reset Value
VERSION	0x00000440

21. Serial Peripheral Interface (SPI)

Rev. 2.1.1.1

21.1 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**

21.2 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.

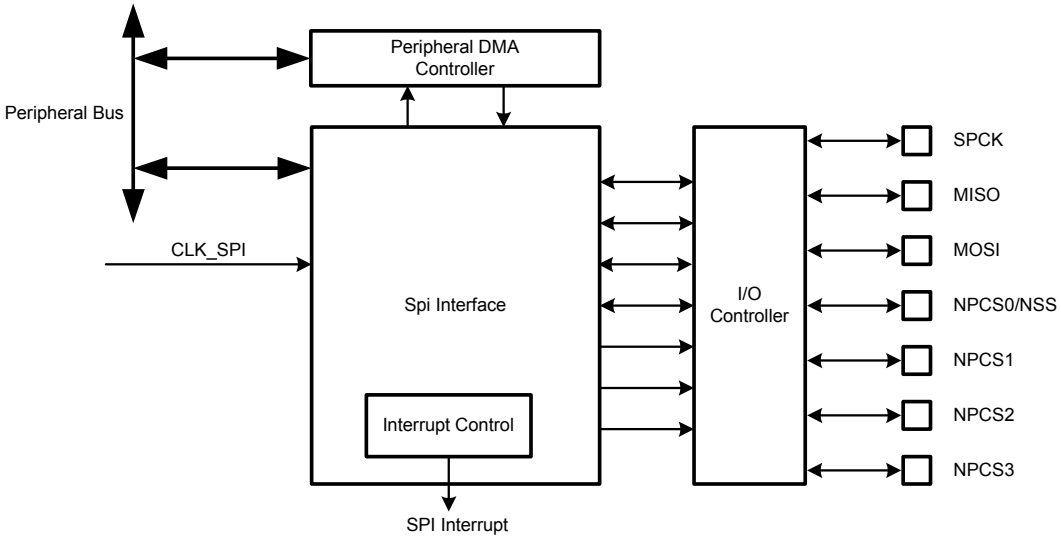
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

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.

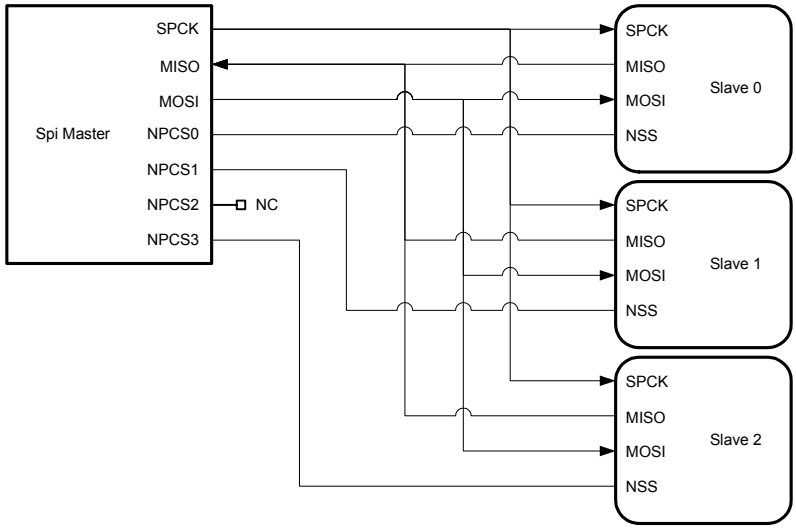
21.3 Block Diagram

Figure 21-1. SPI Block Diagram



21.4 Application Block Diagram

Figure 21-2. Application Block Diagram: Single Master/Multiple Slave Implementation



21.5 I/O Lines Description

Table 21-1. 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

21.6 Product Dependencies

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

21.6.1 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.

21.6.2 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.

21.6.3 Interrupts

The SPI interrupt request line is connected to the interrupt controller. Using the SPI interrupt requires the interrupt controller to be programmed first.

21.7 Functional Description

21.7.1 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.

21.7.2 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 21-2 on page 435 shows the four modes and corresponding parameter settings.

Table 21-2. SPI modes

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 21-3 on page 435 and Figure 21-4 on page 436 show examples of data transfers.

Figure 21-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)

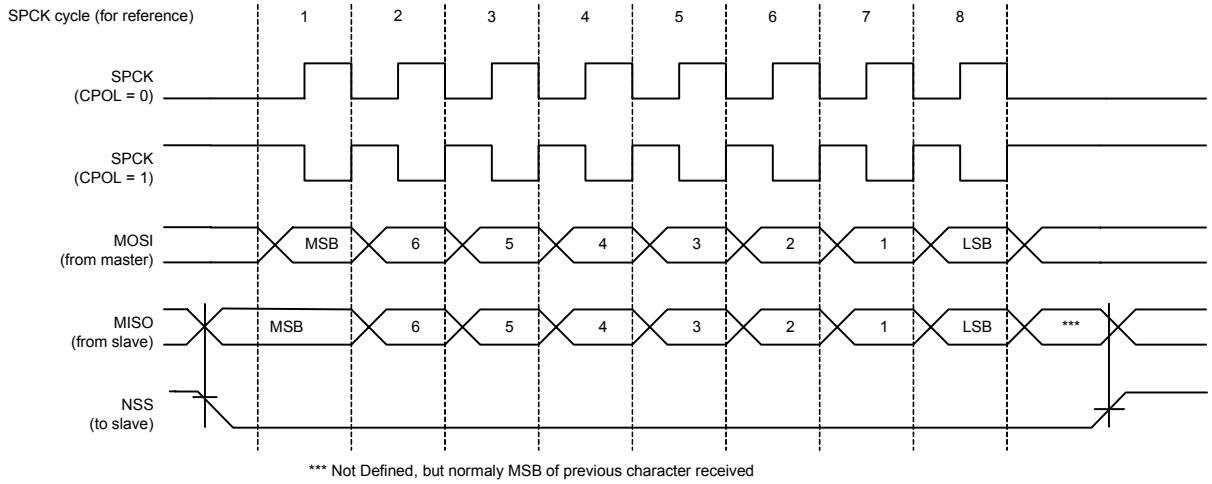
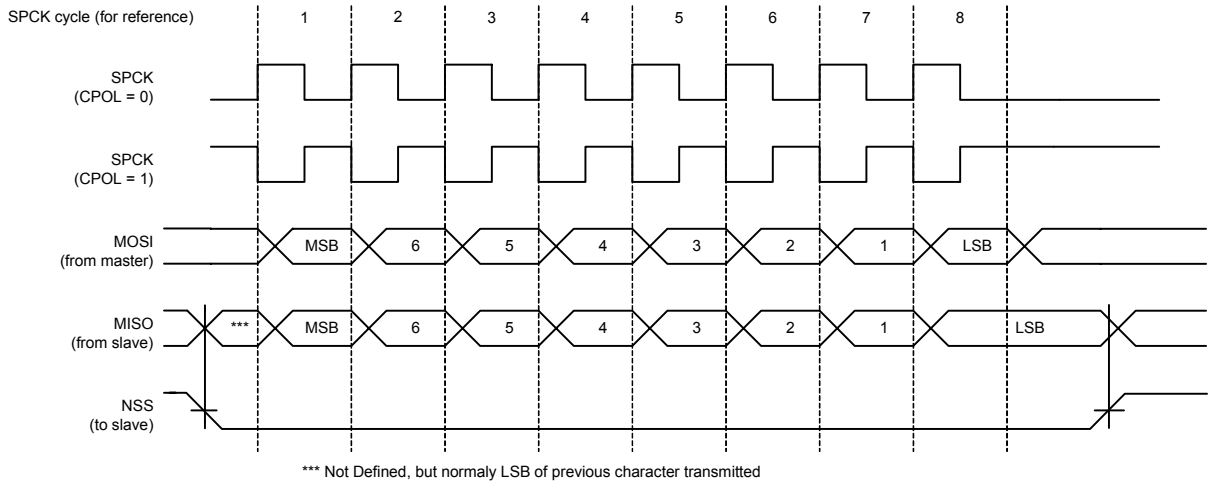


Figure 21-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



21.7.3 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.

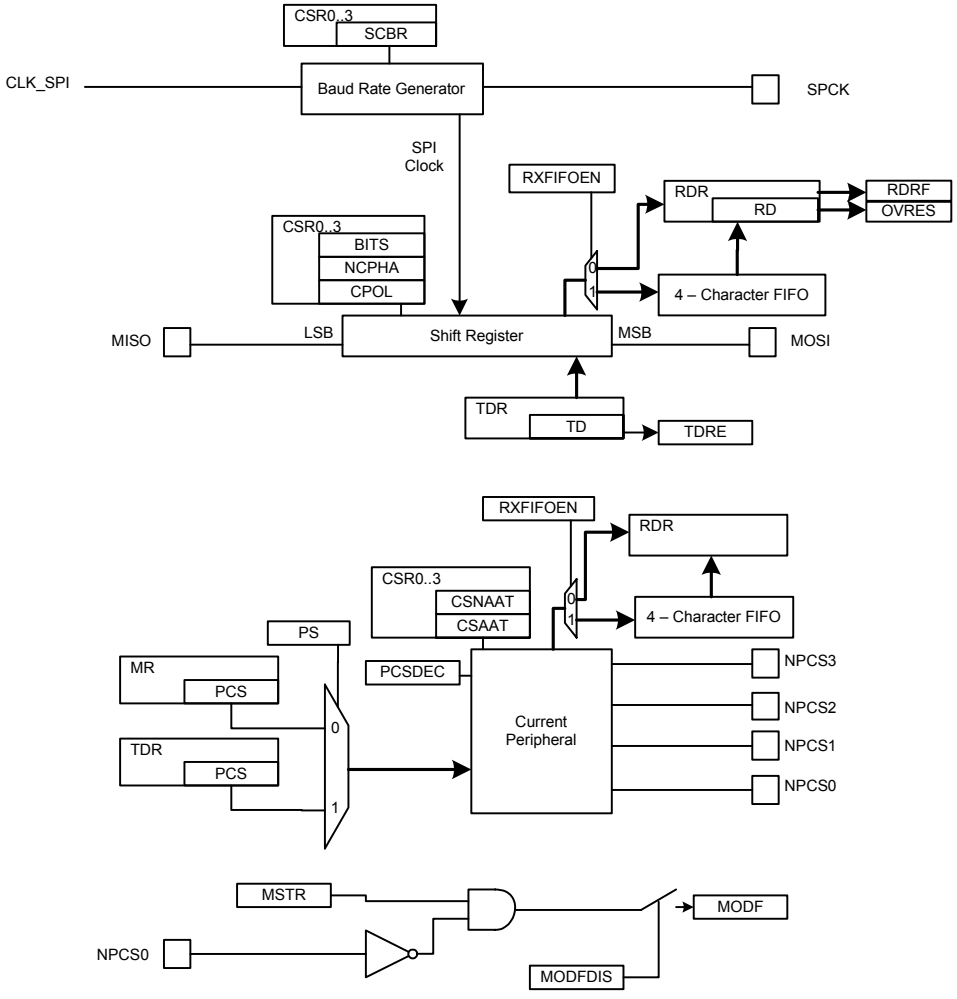
During reception, received data are transferred from the Shift Register to the reception FIFO. The FIFO can contain up to 4 characters (both Receive Data and Peripheral Chip Select fields). While a character of the FIFO is unread, the Receive Data Register Full bit in SR remains high (SR.RDRF). Characters 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 21.7.3.8.

In master mode, if the received data is not read fast enough compared to the transfer rhythm imposed by the write accesses in the TDR, some overrun errors may occur, even if the FIFO is enabled. To insure a perfect data integrity of received data (especially at high data rate), the mode Wait Data Read Before Transfer can be enabled in the MR register (MR.WDRBT). When this mode is activated, no transfer starts while received data remains unread in the RDR. When data is written to the TDR and if unread received data is stored in the RDR, the transfer is paused until the RDR is read. In this mode no overrun error can occur. Please note that if this mode is enabled, it is useless to activate the FIFO in reception.

Figure 21-5 on page 437shows a block diagram of the SPI when operating in master mode. Figure 21-6 on page 438 shows a flow chart describing how transfers are handled.

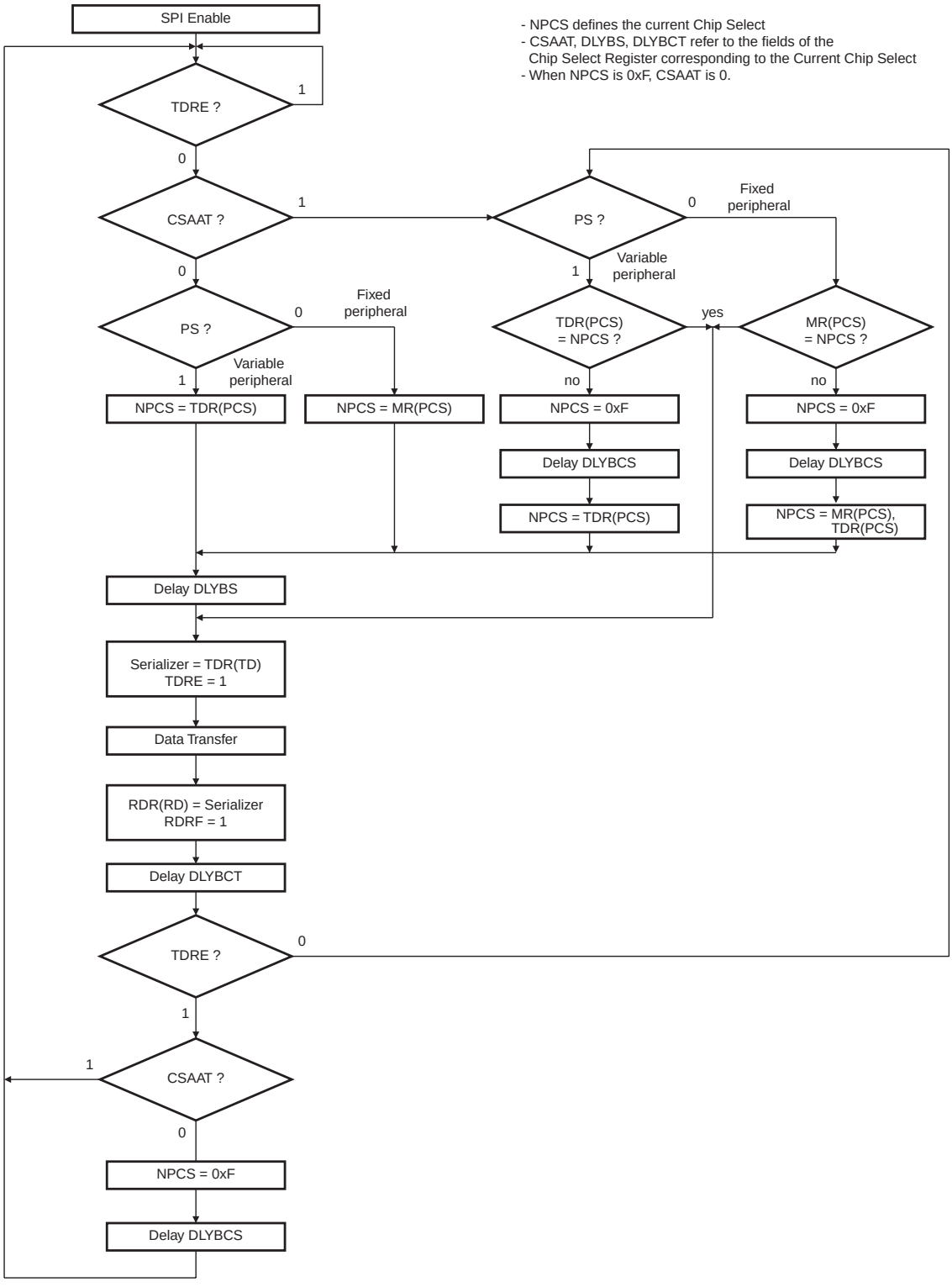
21.7.3.1 Master mode block diagram

Figure 21-5. Master Mode Block Diagram



21.7.3.2 Master mode flow diagram

Figure 21-6. Master Mode Flow Diagram



21.7.3.3 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.

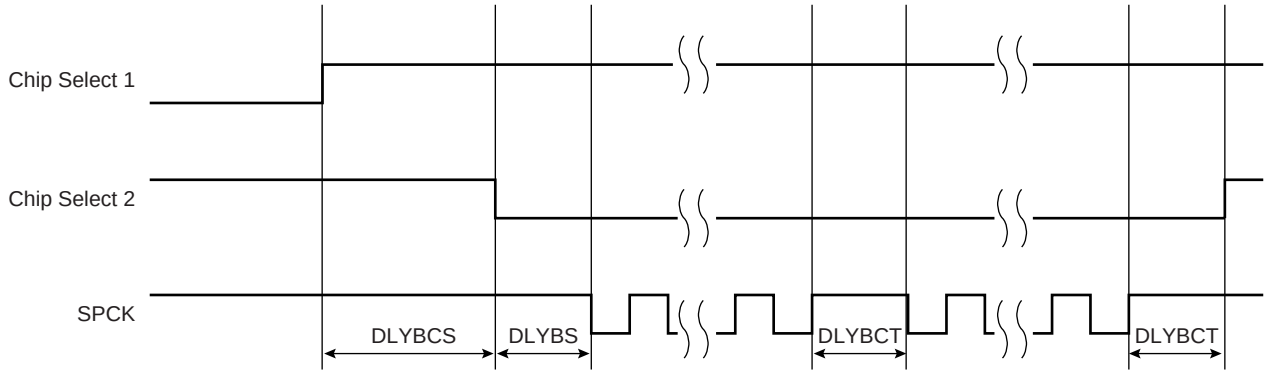
21.7.3.4 Transfer delays

Figure 21-7 on page 439 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 21-7. Programmable Delays



21.7.3.5 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.

21.7.3.6 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.

21.7.3.7 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 21-8 on page 442 shows different peripheral deselection cases and the effect of the CSRn.CSAAT and CSRn.CSNAAT bits.

21.7.3.8 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 21-8. Peripheral Deselection

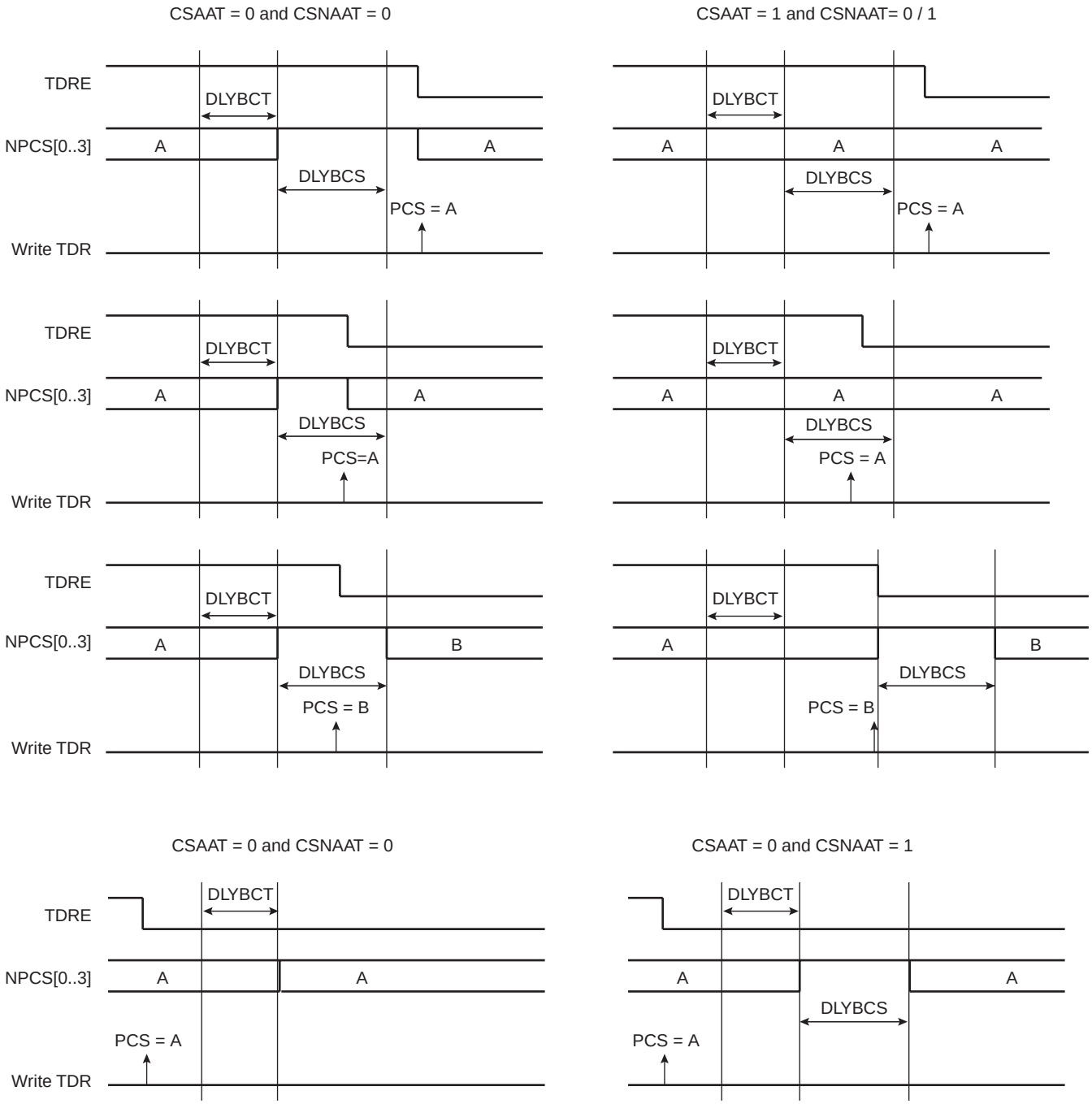


Figure 21-8 on page 442 shows different peripheral deselection cases and the effect of the CSRn.CSAAT and CSRn.CSNAAT bits.

21.7.3.9 Mode fault detection

A mode fault is detected when the SPI is configured in master mode and a low level is driven by an external master on the NPCS0/NSS signal. NPCS0, MOSI, MISO and SPCK must be configured in open drain through the I/O Controller, so that external pull up resistors are needed to guarantee high level.

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).

21.7.4 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. As long as this bit is set, data is loaded in RDR. The user has to read the SR register to clear the SR.OVRES bit.

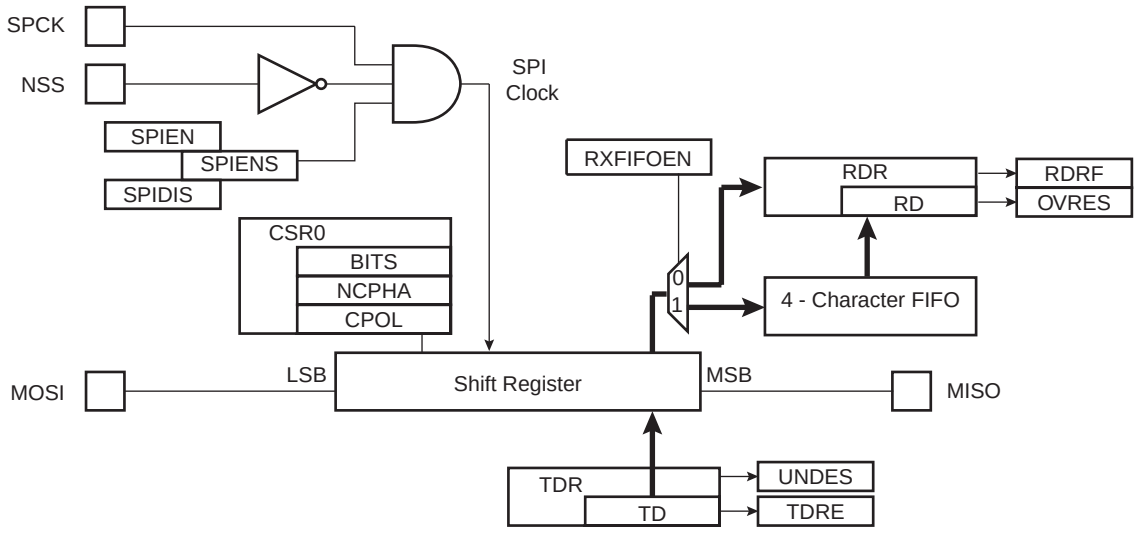
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the TDR register, the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets to zero.

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 21-9 on page 444 shows a block diagram of the SPI when operating in slave mode.

Figure 21-9. Slave Mode Functional Block Diagram



21.8 User Interface

Table 21-3. 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	0x000000F0
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.

21.8.1 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.

21.8.2 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	WDRBT	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).

- **WDRBT: Wait Data Read Before Transfer**
  1: In master mode, a transfer can start only if the RDR register is empty, i.e. does not contain any unread data. This mode prevents overrun error in reception.
  0: No Effect. In master mode, a transfer can be initiated whatever the state of the RDR register is.
- **MODFDIS: Mode Fault Detection**
  1: Mode fault detection is disabled.
  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.

21.8.3 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.

21.8.4 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.

21.8.5 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

21.8.6 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.

21.8.7 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.

21.8.8 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.

21.8.9 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.

21.8.10 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.

21.8.11 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.

21.8.12 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.

21.8.13 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

21.8.14 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.

21.8.15 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**
- **FIFORIMPL: FIFO in Reception Implemented**
- **BRPBHSB: Bridge Type is PB to HSB**
- **CSNAATIMPL: CSNAAT Features Implemented**
- **EXTDEC: External Decoder True**
- **LENNCONF: Character Length if not Configurable**
- **LENCONF: Character Length Configurable**
- **PHZNCONF: Phase is Zero if Phase not Configurable**
- **PHCONF: Phase Configurable**
- **PPNCONF: Polarity Positive if Polarity not Configurable**
- **PCONF: Polarity Configurable**
- **NCS: Number of Chip Selects**

21.8.16 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
-	-	-	-	VARIANT			

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

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

- **VARIANT**

  Reserved. No functionality associated.

- **VERSION**

  Version number of the module. No functionality associated.

21.9 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 21-4. SPI Clock Name

Module Name	Clock Name
SPI	CLK_SPI

Table 21-5.

Register	Reset Value
FEATURES	0x001F0154
VERSION	0x00000211

22. Two-Wire Master Interface (TWIM)

Rev 1.0.1.1

22.1 Features

- **Compatible with I²C standard**
  - **Multi-master support**
  - **100 and 400 kbit/s transfer speeds**
  - **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**

22.2 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 22-1 on page 472 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I²C compatible device.

Table 22-1. Atmel TWIM Compatibility with I²C Standard

I²C Standard	Atmel TWIM
Standard Mode Speed (100 KHz)	Supported
Fast Mode Speed (400 KHz)	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 22-2 on page 473 lists the compatibility level of the Atmel Two-wire Master Interface and a full SMBus compatible master.

Table 22-2. 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

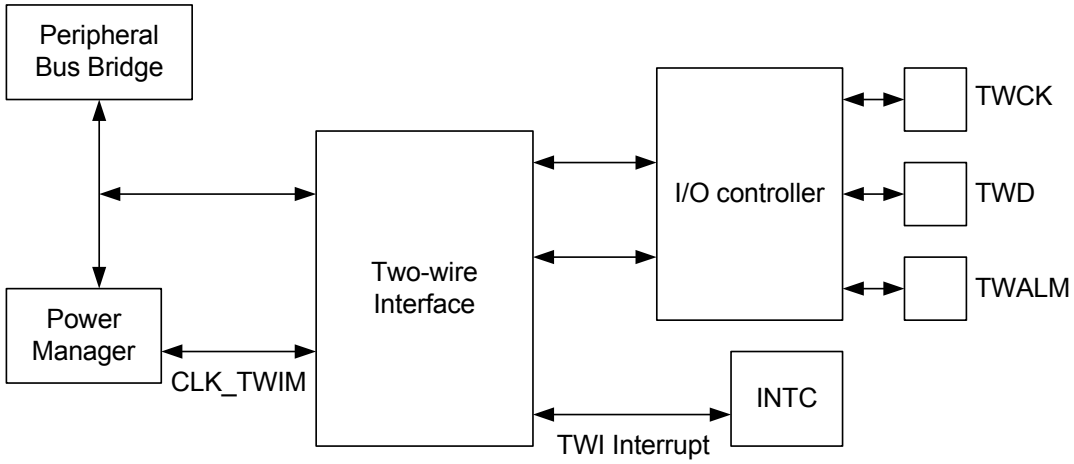
22.3 List of Abbreviations

Table 22-3. 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

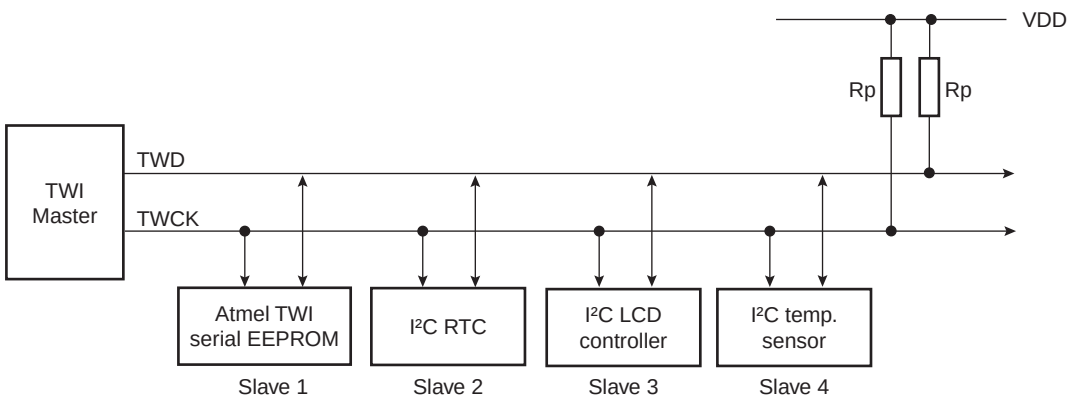
22.4 Block Diagram

Figure 22-1. Block Diagram



22.5 Application Block Diagram

Figure 22-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

22.6 I/O Lines Description

Table 22-4. 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

22.7 Product Dependencies

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

22.7.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 22-2 on page 474). 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.

22.7.2 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.

22.7.3 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.

22.7.4 Interrupts

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

22.7.5 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.

22.8 Functional Description

22.8.1 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 22-4 on page 476).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 22-4 on page 476).

- 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 22-3. START and STOP Conditions

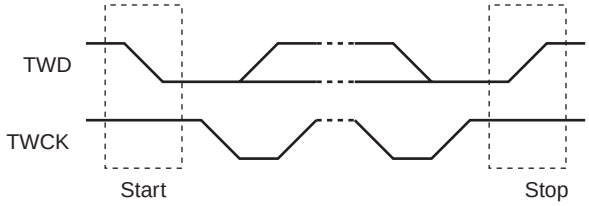
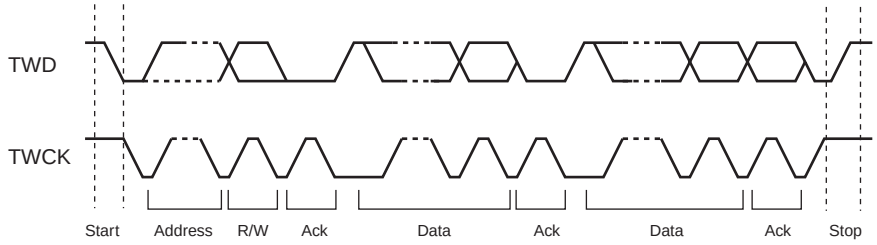


Figure 22-4. Transfer Format



22.8.2 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.

22.8.2.1 Clock Generation

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}.

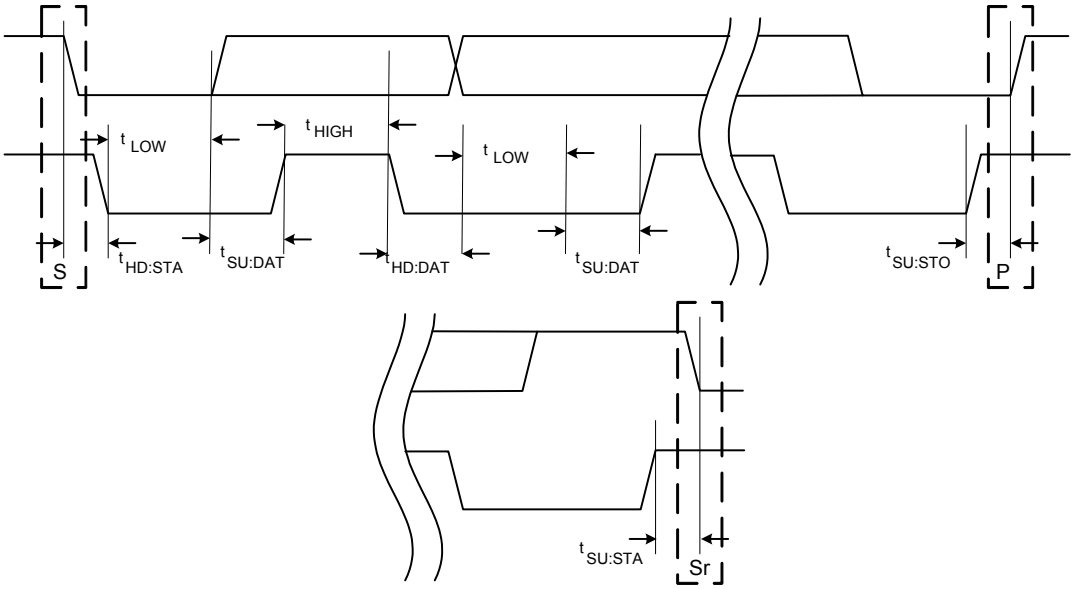
EXP: Specifies the clock prescaler setting.

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 22-5. Bus Timing Diagram



22.8.2.2 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.

22.8.3 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 22-6 on page 479 and Figure 22-7 on page 479.

Figure 22-6. Master Write with One Data Byte

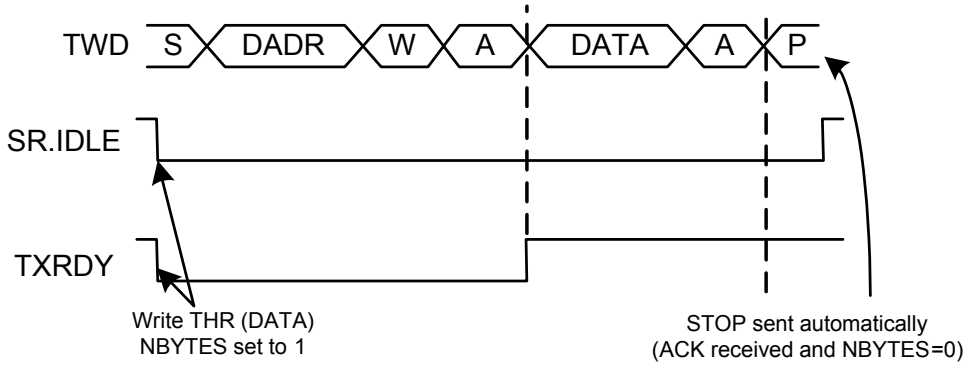
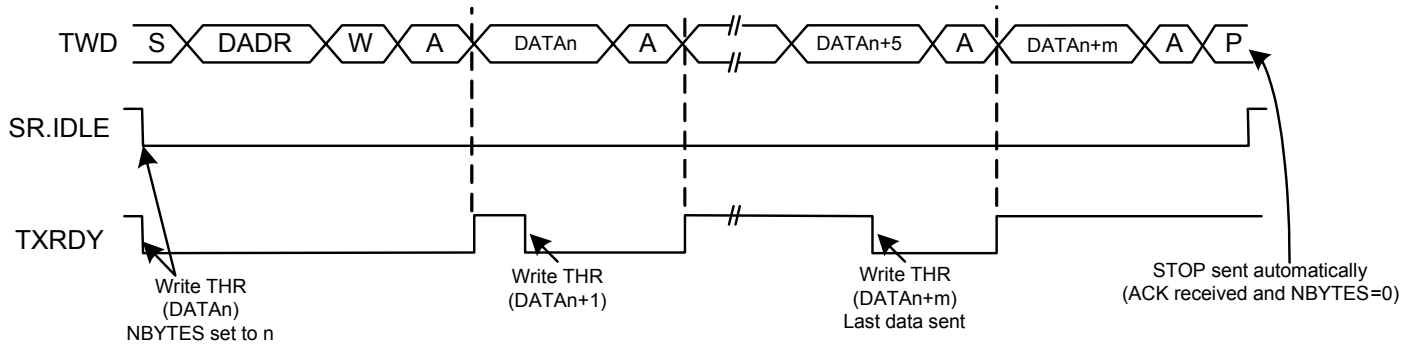


Figure 22-7. Master Write with Multiple Data Bytes



22.8.4 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 22-8. Master Read with One Data Byte

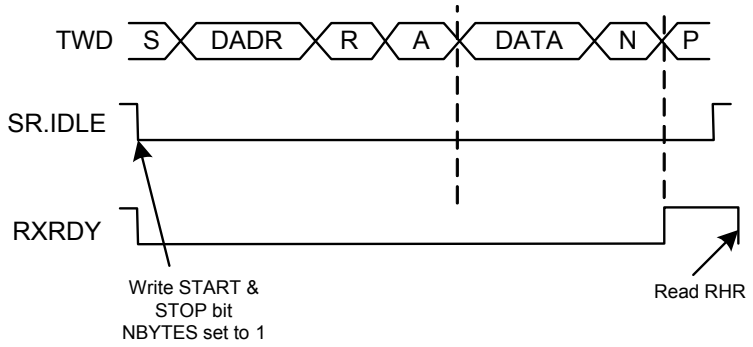
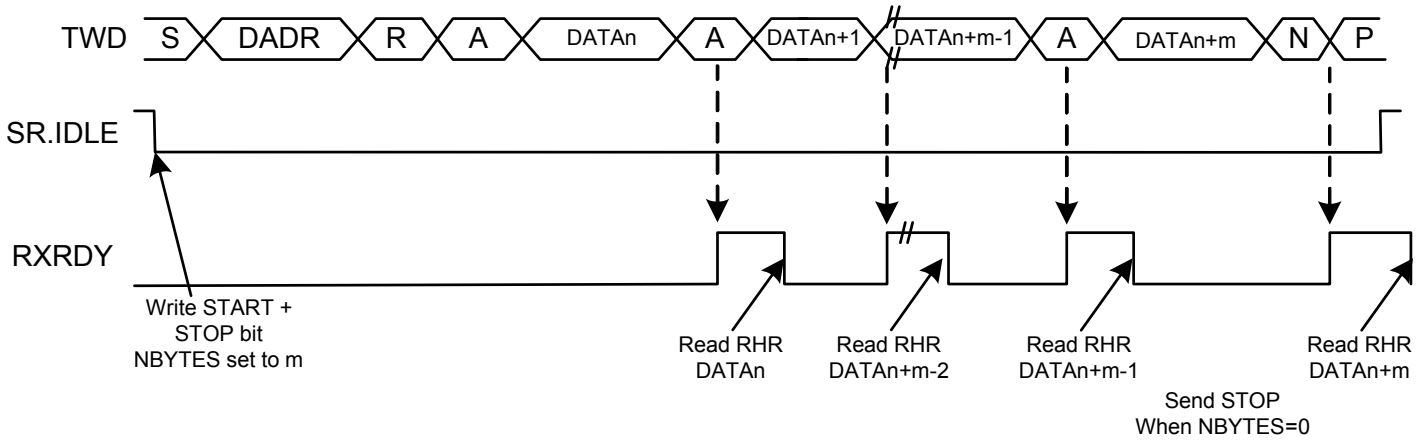


Figure 22-9. Master Read with Multiple Data Bytes



22.8.5 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:

22.8.5.1 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.

22.8.5.2 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.

22.8.6 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 22-11 on page 482.

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 22-10 on page 481).

Note: The state of the bus (busy or free) is not indicated in the user interface.

Figure 22-10. Programmer Sends Data While the Bus is Busy

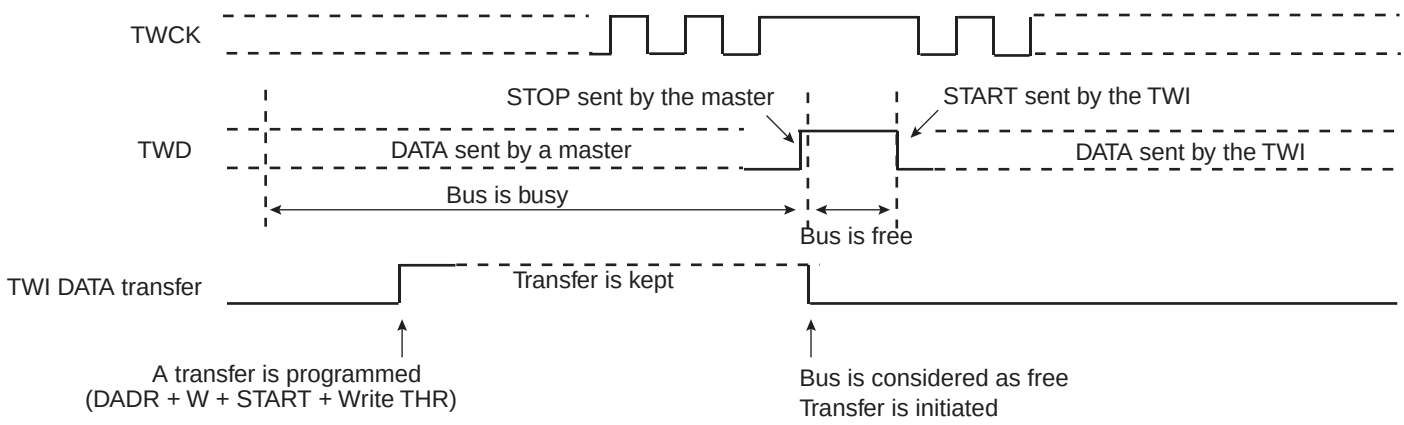
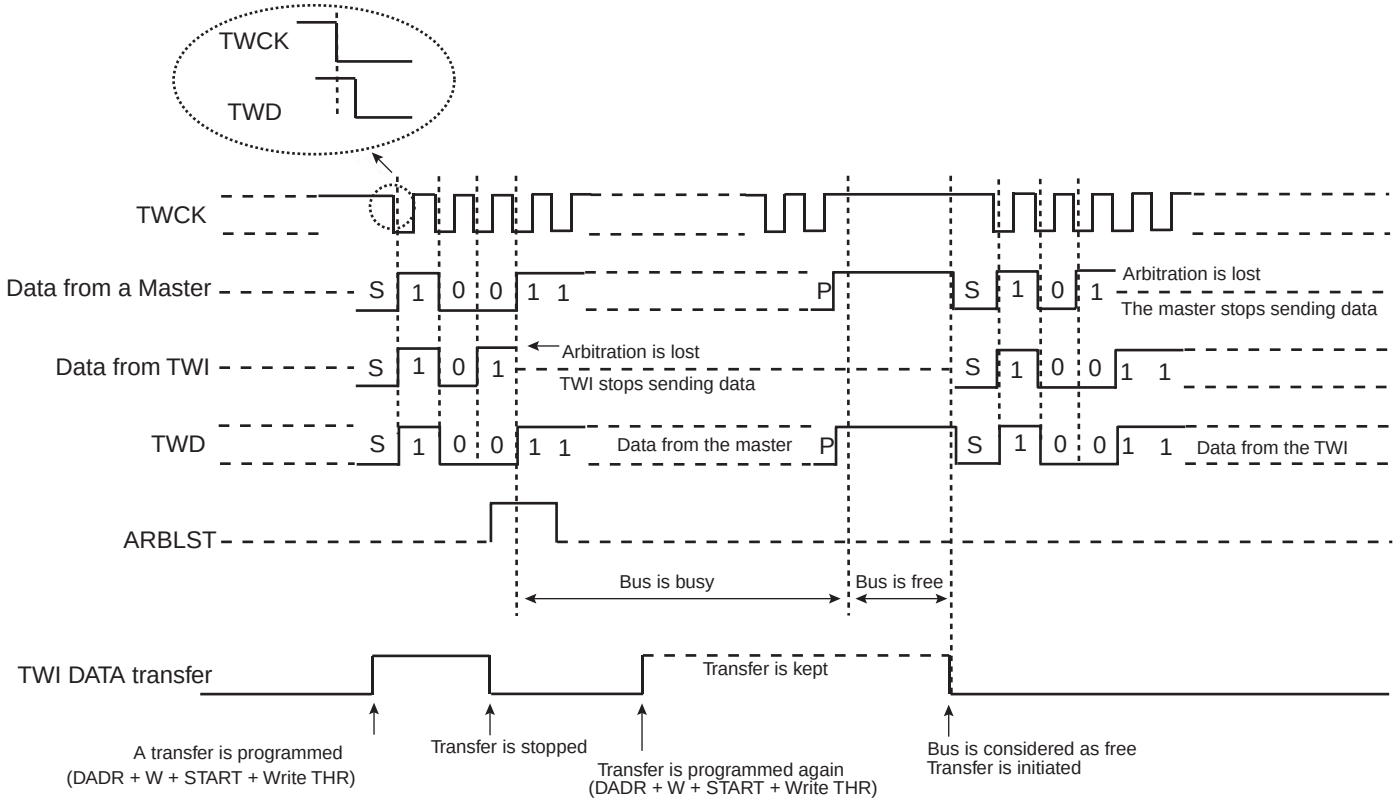


Figure 22-11. Arbitration Cases



22.8.7 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 480

22.8.7.1 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.

22.8.7.2 *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.

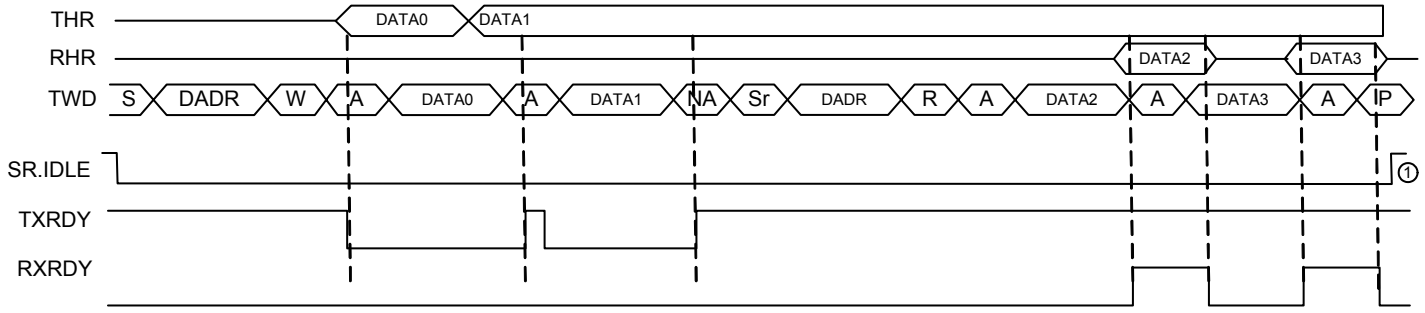
If combining several transfers, without any STOP or REPEATED START between them, remember to set the ACKLAST bit in CMDR to keep from ending each of the partial transfers with a NACK.

22.8.7.3 *Write Followed by Read*

Consider the following transfer:

START, DADR+W, DATA+A, DATA+A, REPSTART, DADR+R, DATA+A, DATA+NA, STOP.

Figure 22-12. Combining a Write and Read Transfer



To generate this transfer:

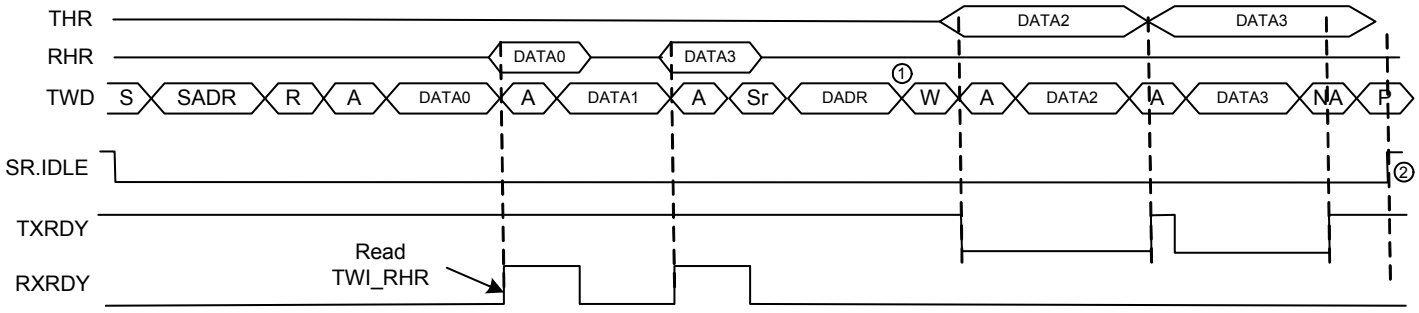
1. Program CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=0.
2. Program NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=1.
3. Wait until SR.TXRDY==1, then write first data byte to transfer to THR.
4. Wait until SR.TXRDY==1, then write second data byte to transfer to THR.
5. Wait until SR.RXRDY==1, then read first data byte received from RHR.
6. Wait until SR.RXRDY==1, then read second data byte received from RHR.

22.8.7.4 *Read Followed by Write*

Consider the following transfer:

START, DADR+R, DATA+A, DATA+NA, REPSTART, DADR+W, DATA+A, DATA+A, STOP.

Figure 22-13. 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.

22.8.8 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.

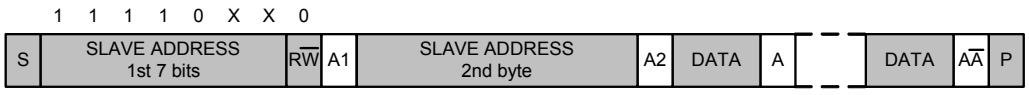
In Figure 22-14 on page 484 and Figure 22-15 on page 485, the grey boxes represent signals driven by the master, the white boxes are driven by the slave.

22.8.8.1 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 22-14. A Write Transfer with 10-bit Addressing



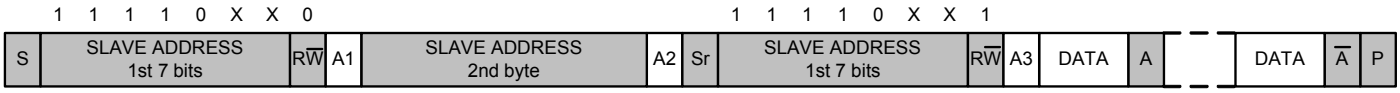
22.8.8.2 Master Receiver

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 22-15. A Read Transfer with 10-bit Addressing



22.8.9 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.

22.8.9.1 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.

22.8.9.2 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.

22.8.9.3 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.

22.8.10 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 22-5. 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.

Table 22-5. Bus Events

Event	Effect
SMBus timeout received	SR.SMBTOUT is set. SR.CCOMP not set. CMDR.VALID remains set. STOP automatically transmitted on bus.
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.

22.9 User Interface

Table 22-6. 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.

22.9.1 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.

22.9.2 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}.

22.9.3 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 $T_{LOW:SEXT}$.

22.9.4 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.

22.9.5 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.

22.9.6 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.

22.9.7 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.

22.9.8 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.

22.9.9 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

22.9.10 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

22.9.11 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.

22.9.12 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.

22.9.13 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
-	-	-	-	-	-	-	-

22.9.14 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.

22.10 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 22-7. Module Clock Name

Module Name	Clock Name
TWIM0	CLK_TWIM0
TWIM1	CLK_TWIM1

Table 22-8. Register Reset Values

Register	Reset Value
VERSION	0x0000 0101
PARAMETER	0x0000 0000

23. Two-Wire Slave Interface (TWIS)

Rev 1.1.2.1

23.1 Features

- **Compatible with I²C standard**
  - **100 and 400 kbit/s 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**

23.2 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 23-1 on page 506 lists the compatibility level of the Atmel Two-wire Slave Interface and a full I²C compatible device.

Table 23-1. Atmel TWIS Compatibility with I²C Standard

I²C Standard	Atmel TWIS
Standard Mode Speed (100 KHz)	Supported
Fast Mode Speed (400 KHz)	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 23-2 on page 507 lists the compatibility level of the Atmel Two-wire Slave Interface and a full SMBus compatible device.

Table 23-2. Atmel TWIS Compatibility with SMBus Standard

SMBus Standard	Atmel TWIS
Bus Timeouts	Supported
Address Resolution Protocol	Supported
Alert	Supported
Packet Error Checking	Supported

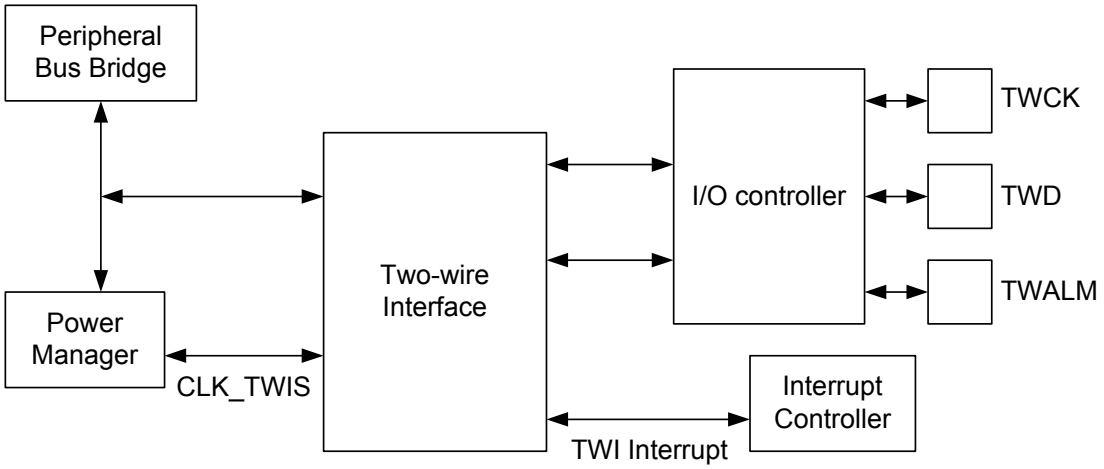
23.3 List of Abbreviations

Table 23-3. 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

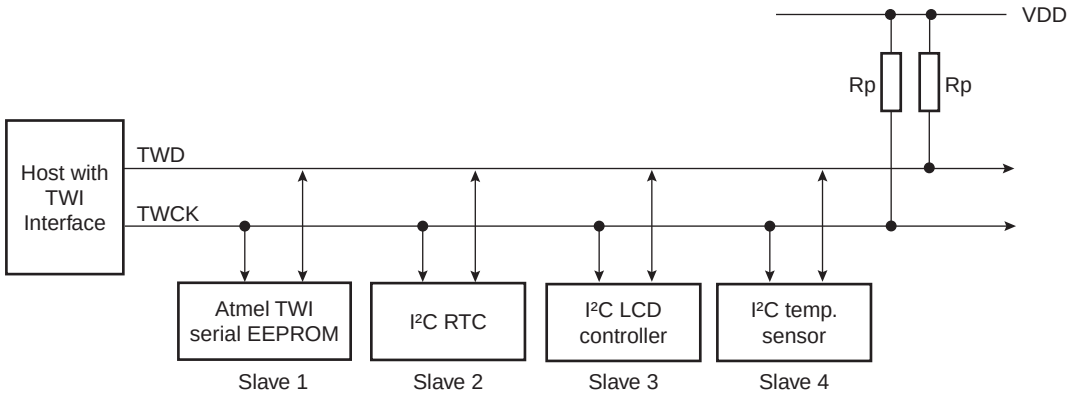
23.4 Block Diagram

Figure 23-1. Block Diagram



23.5 Application Block Diagram

Figure 23-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

23.6 I/O Lines Description

Table 23-4. 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

23.7 Product Dependencies

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

23.7.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 23-2 on page 508). 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.

23.7.2 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 23.8.7 on page 516.

23.7.3 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.

23.7.4 Interrupts

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

23.7.5 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.

23.8 Functional Description

23.8.1 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 23-4 on page 509).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 23-3 on page 509).

- 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 23-3. START and STOP Conditions

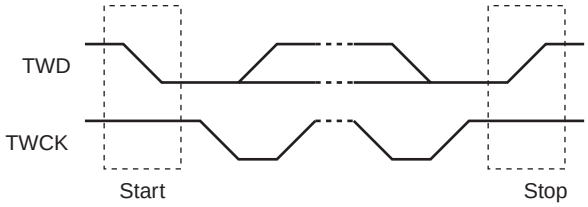
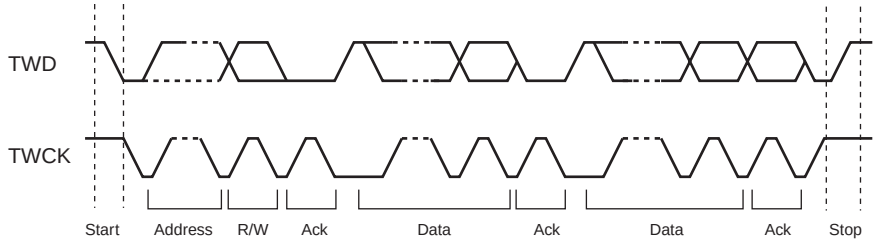


Figure 23-4. Transfer Format



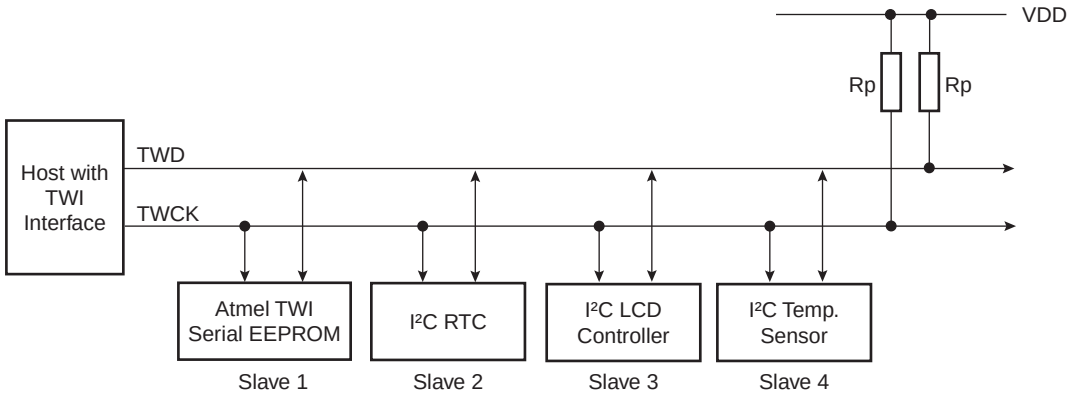
23.8.2 Operation

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 23-5. Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

23.8.2.1 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:

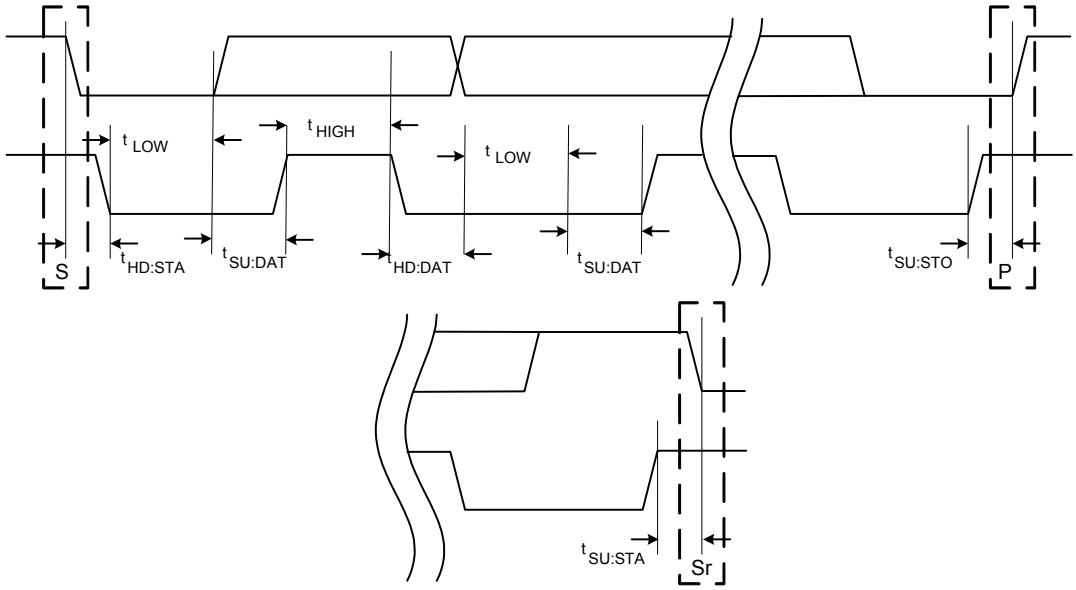
TLOWS: Prescaled clock cycles used to time SMBUS timeout $T_{LOW:SEXT}$.

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 23-6. Bus Timing Diagram



23.8.2.2 *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.

23.8.2.3 *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.

23.8.2.4 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.

23.8.2.5 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.

23.8.3 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 SR.TRA

After the address phase, the following is done:

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 low period of TWCK. SR.TXRDY indicates the state of THR.
   – 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. Transmit the correct data byte. Set SR.BTF when done.
4. Update NBYTES. If CR.CUP is set, NBYTES is incremented, otherwise NBYTES is decremented.
5. After each data byte has been transferred, the master transmits an ACK or NAK bit. If a NAK bit is received, transfer is finished, and TWIS will wait for a STOP or REPEATED START. If an ACK bit is received, more data should be transmitted, jump to step 2.
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 Not Acknowledge bit (NAK**)** 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 23-7 on page 513 and Figure 23-8 on page 513.

Figure 23-7. Slave Transmitter with One Data Byte

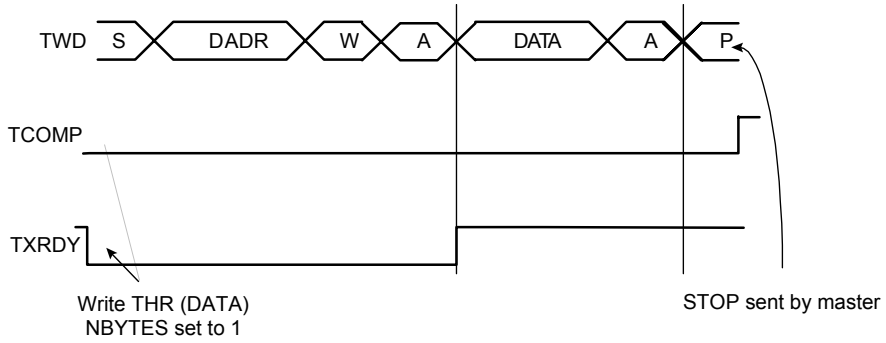
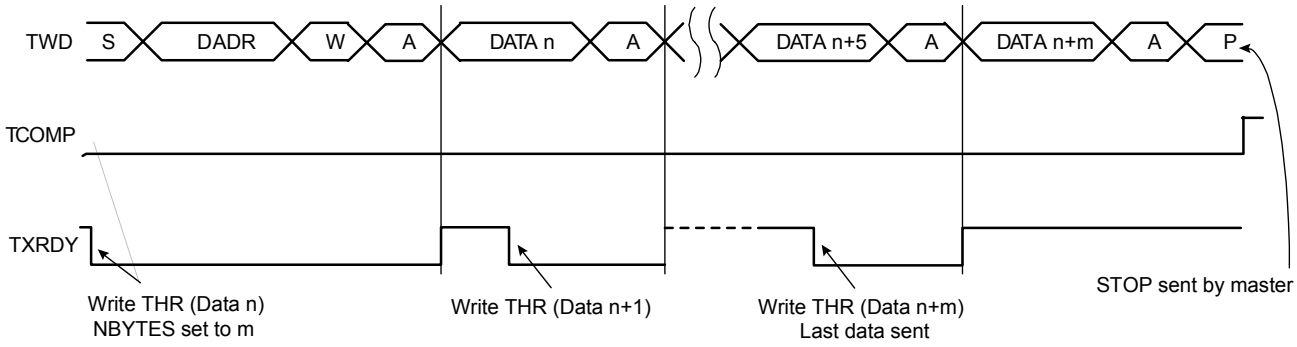


Figure 23-8. Slave Transmitter with Multiple Data Bytes



23.8.4 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.

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 23-9. Slave Receiver with One Data Byte

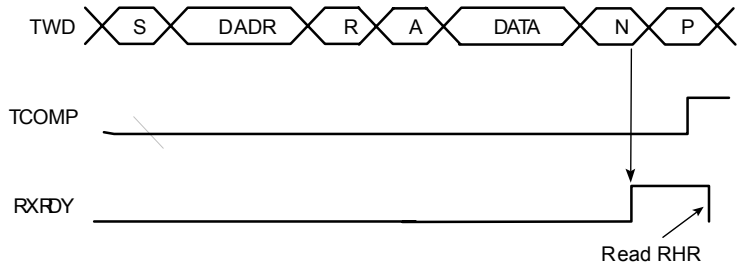
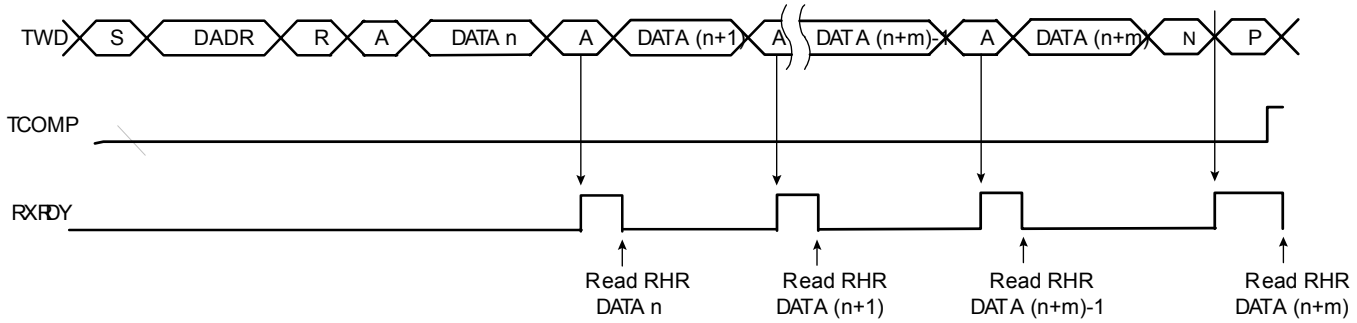


Figure 23-10. Slave Receiver with Multiple Data Bytes



23.8.5 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. 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:

23.8.5.1 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.

23.8.5.2 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.

23.8.6 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.

23.8.6.1 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 com-

pare 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.

23.8.6.2 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.

23.8.6.3 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).

23.8.7 Wakeup from Sleep Modes by TWI Address Match

The TWIS is able to wake the device up from sleep modes upon an address match, including modes where CLK_TWIS is stopped. If a TWI Start condition is received in a sleep mode where CLK_TWIS is stopped, TWIS will stretch TWCK until CLK_TWIS has started. The time required for restarting CLK_TWIS depends on which sleep mode the system was in.

When CLK_TWIS has been restarted, the TWCK stretching is released and the slave address will be received on the TWI bus. To save power, only a limited part of the device including TWIS receives a clock at this time. If the address phase causes a TWIS address match, the entire device will be wakened and normal TWIS address match actions performed. Normal TWI transfer will then follow. If the TWIS was not addressed by the transfer, CLK_TWIS will automatically be stopped and the system will go back to the original sleep mode.

23.8.8 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 23-5. 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.

Table 23-5. Bus Events

Event	Effect
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. 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. 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.
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.

23.9 User Interface

Table 23-6. 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.

23.9.1 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	TENBIT	ADR[9:8]	

23	22	21	20	19	18	17	16
ADR[7:0]							

15	14	13	12	11	10	9	8
	SOAM	CUP	ACK	PECEN	SMHH	SMDA	SMBALERT

7	6	5	4	3	2	1	0
SWRST	-	-	STREN	GCMATCH	SMATCH	SMEN	SEN

- **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.
- **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.

23.9.2 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.

23.9.3 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 $T_{LOW:SEXT}$.

23.9.4 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.

23.9.5 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.

23.9.6 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.

23.9.7 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.

23.9.8 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.

23.9.9 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.

23.9.10 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.

23.9.11 Status Clear Register

Name: SCR

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
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.

23.9.12 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
-	-	-	-	-	-	-	-

This register always reads as zero. No functionality associated.

23.9.13 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.

23.10 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 23-7. Module Clock Name

Module Name	Clock Name
TWIS0	CLK_TWIS0
TWIS1	CLK_TWIS1

Table 23-8. Register Reset Values

Register	Reset Value
VERSION	0x00000112
PARAMETER	0x00000000

24. Pulse Width Modulation Controller (PWMA)

Rev 1.0.1.0

24.1 Features

- **Left-aligned non-inverted 8-bit PWM**
- **Common 8-bit timebase counter**
  - **Asynchronous clock source supported**
  - **Spread-spectrum counter to allow a constantly varying duty cycle**
- **Separate 8-bit duty cycle register per channel**
- **Synchronized channel updates**
  - **No glitches when changing the duty cycles**
- **Interlinked operation supported**
  - **Multiple channels can be updated with the same duty cycle value at a time**
  - **Up to four channels can be updated with different duty cycle values at a time**
- **Interrupt on PWM timebase overflow**
- **Incoming peripheral events supported**
  - **Pre-defined channels support incoming (increase/decrease) peripheral events from the Peripheral Event System**
  - **Increase event will increase the duty cycle by one**
  - **Decrease event will decrease the duty cycle by one**
- **One output peripheral event supported**
  - **Connected to channel 0 and asserted when the common timebase counter is equal to the programmed duty cycle for channel 0**
- **Output PWM waveform for each channel**
- **Open drain driving on selected pins for 5V PWM operation**

24.2 Overview

The Pulse Width Modulation Controller (PWMA) controls several pulse width modulation (PWM) channels. The number of channels is specific to the device. Each channel controls one square output PWM waveform. Characteristics of the output PWM waveforms such as period and duty cycle are configured through the user interface. All user interface registers are mapped on the peripheral bus.

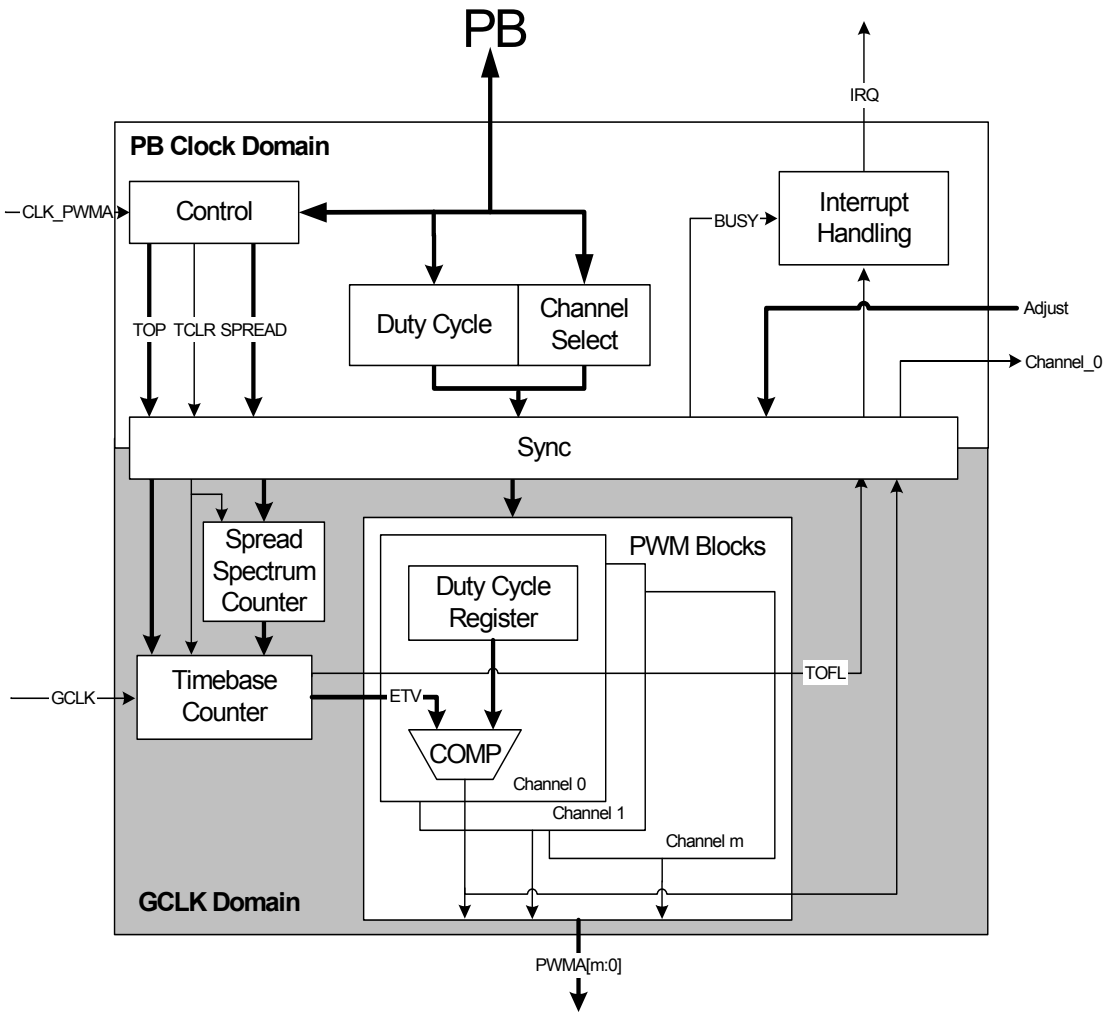
The duty cycle value for each channel can be set independently, while the period is determined by a common timebase counter (TC). The timebase for the counter is selected by using the allocated asynchronous Generic Clock (GCLK). The user interface for the PWMA contains handshake and synchronizing logic to ensure that no glitches occur on the output PWM waveforms while changing the duty cycle values.

PWMA duty cycle values can be changed using two approaches, either an interlinked single-value mode or an interlinked multi-value mode. In the interlinked single-value mode, any set of channels, up to 32 channels, can be updated simultaneously with the same value while the other channels remain unchanged. In the interlinked multi-value mode, up to 4 selected channels can be updated with 4 different values while the other channels remain unchanged.

Some pins can be driven in open drain mode, allowing the PWMA to generate a 5V waveform using an external pullup resistor.

24.3 Block Diagram

Figure 24-1. PWMA Block Diagram



24.4 I/O Lines Description

Each channel outputs one PWM waveform on one external I/O line.

Table 24-1. I/O Line Description

Pin Name	Pin Description	Type
PWMA[n]	Output PWM waveform for one channel n	Output
PWMMOD[n]	Output PWM waveform for one channel n, open drain mode	Output

24.5 Product Dependencies

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

24.5.1 I/O Lines

The pins used for interfacing the PWMA may be multiplexed with I/O Controller lines. The programmer must first program the I/O Controller to assign the desired PWMA pins to their peripheral function.

It is only required to enable the PWMA outputs actually in use.

24.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the PWMA, the PWMA will stop functioning and resume operation after the system wakes up from sleep mode.

24.5.3 Clocks

The clock for the PWMA bus interface (CLK_PWMA) 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 PWMA before disabling the clock, to avoid freezing the PWMA in an undefined state.

Additionally, the PWMA depends on a dedicated Generic Clock (GCLK). The GCLK can be set to a wide range of frequencies and clock sources and must be enabled in the System Control Interface (SCIF) before the PWMA can be used.

24.5.4 Interrupts

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

24.5.5 Peripheral Events

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

24.5.6 Debug Operation

When an external debugger forces the CPU into debug mode, the PWMA continues normal operation. If the PWMA is configured in a way that requires it to be periodically serviced by the CPU through interrupts, improper operation or data loss may result during debugging.

24.6 Functional Description

The PWMA embeds a number of PWM channel submodules, each providing an output PWM waveform. Each PWM channel contains a duty cycle register and a comparator. A common timebase counter for all channels determines the frequency and the period for all the PWM waveforms.

24.6.1 Enabling the PWMA

Once the GCLK has been enabled, the PWMA is enabled by writing a one to the EN bit in the Control Register (CR).

24.6.2 Timebase Counter

The top value of the timebase counter defines the period of the PWMA output waveform. The timebase counter starts at zero when the PWMA is enabled and counts upwards until it reaches its effective top value (ETV). The effective top value is defined by specifying the desired number of GCLK clock cycles in the TOP field of CR (CR.TOP) in normal operation (CR.SPREAD is

zero). When the timebase counter reaches its effective top value, it restarts counting from zero. The period of the PWMA output waveform is then:

$$T_{PWMA} = (ETV + 1) \cdot T_{GCLK}$$

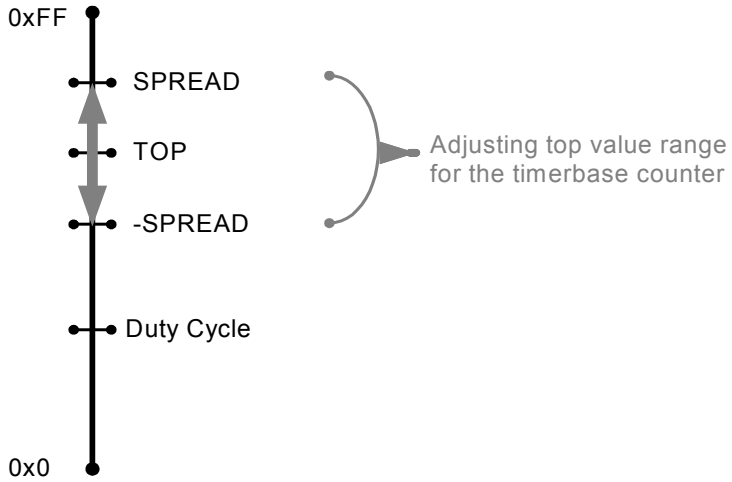
The timebase counter can be reset by writing a one to the Timebase Clear bit in CR (CR.TCLR). Note that this can cause a glitch to the output PWM waveforms in use.

24.6.3 Spread Spectrum Counter

The spread spectrum counter allows the generation of constantly varying duty cycles on the output PWM waveforms. This is achieved by varying the effective top value of the timebase counter in a range defined by the spread spectrum counter value.

When CR.SPREAD is not zero, the spread spectrum counter is enabled. Its range is defined by CR.SPREAD. It starts to count from -CR.SPREAD when the PWMA is enabled or after reset and counts upwards. When it reaches CR.SPREAD, it restarts to count from -CR.SPREAD again. The spread spectrum counter will cause the effective top value (ETV) to vary from TOP-SPREAD to TOP+SPREAD. Figure 24-2 on page 539 illustrates this. This leads to a constantly varying duty cycle on the PWM output waveforms though the duty cycle values stored are unchanged.

Figure 24-2. PWMA Adjusting Top Value for Timebase Counter



24.6.3.1 Special considerations

The maximum value of the timebase counter is 255. If SPREAD is written to a value that will cause the ETV to exceed this value, the spread spectrum counter's range will be limited to prevent the timebase counter to exceed its maximum value.

If SPREAD is written to a value causing (TOP-SPREAD) to be below zero, the spread specturm counter's range will be limited to prevent the timebase counter to count below zero.

In both cases, the SPREAD value read from the Control Register will be the same value as written to the SPREAD field.

When writing a one to CR.TCLR, the timebase counter and the spread spectrum counter are reset at their lower limit values and the effective top value of the timebase counter will also be reset.

24.6.4 Duty Cycle and Waveform Properties

Each PWM channel has its own duty cycle value (DCV) which is write-only and cannot be read out. The duty cycle value can be changed in two approaches as described in Section24.6.5.

When the duty cycle value is zero, the PWM output is zero. Otherwise, the PWM output is set when the timebase counter is zero, and cleared when the timebase counter reaches the duty cycle value. This is summarized as:

$$\text{PWM Waveform} = \begin{cases} low \text{ when } DCV = 0 \text{ or } TC > DCV \\ high \text{ when } TC \leq DCV \text{ and } DCV \neq 0 \end{cases}$$

Note that when increasing the duty cycle value for one channel from 0 to 1, the number of GCLK cycles when the PWM waveform is high will jump from 0 to 2. When incrementing the duty cycle value by one for any other values, the number of GCLK cycle when the waveform is high will increase by one. This is summarized in Table 24-2.

Table 24-2. PMW Waveform Duty Cycles

Duty Cycle Value	#Clock Cycles When Waveform is High	#Clock Cycles When Waveform is Low
0	0	ETV+1
1	2	ETV-1
2	3	ETV-2
...	...	...
ETV-1	ETV	1
ETV	ETV+1	0

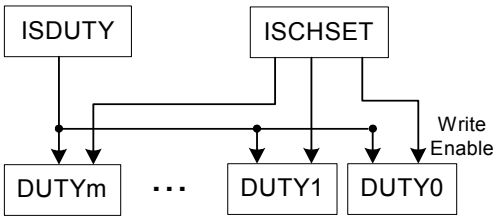
Every other output PWM waveform toggles on the negative edge of the GCLK instead of the positive edge. This is to avoid too many I/O toggling simultaneously on the output I/O lines.

24.6.5 Updating Duty Cycle Values

24.6.5.1 Interlinked Single Value PWM Operation

The PWM channels can be interlinked to allow multiple channels to be updated simultaneously with the same duty cycle value. This value must be written to the Interlinked Single Value Duty (ISDUTY) register. Each channel has a corresponding enabling bit in the Interlinked Single Value Channel Set (ISCHSETm) register. When a bit is written to one in the ISCHSETm register, the duty cycle register for the corresponding channel will be updated with the value stored in the ISDUTY register. It can only be updated when the READY bit in the Status Register (SR.READY) is one, indicating that the PWMA is ready for writing. Figure 24-3 on page 541 shows the writing procedure. It is thus possible to update the duty cycle values for up to 32 PWM channels within one ISCHSETm register at a time.

Figure 24-3. Interlinked Single Value PWM Operation Flow

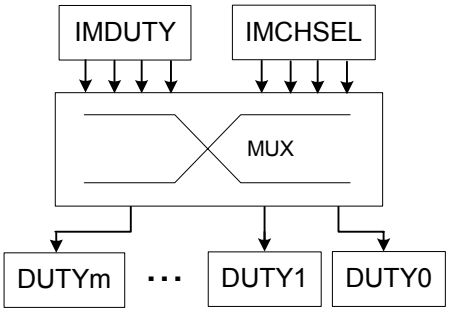


24.6.5.2 Interlinked Multiple Value PWM Operation

The interlinked multiple value PWM operation allows up to four channels to be updated simultaneously with different duty cycle values. These duty cycle values must be written to the IMDUTY register. The index number of the four channels to be updated is written to the four SEL fields in the Interlinked Multiple Value Channel Select (IMCHSEL) register (IMCHSEL.SEL). When the IMCHSEL register is written, the values stored in the IMDUTY register are synchronized to the duty cycle registers for the channels selected by the SEL fields. Figure 24-4 on page 541 shows the writing procedure.

Note that only writes to the implemented channels will be effective. If one of the IMCHSEL.SEL fields points to a non-existing channel, the corresponding value in the IMDUTY register will not be written. If the same channel is specified in multiple IMCHSEL.SEL fields, the channel will be updated with the value stored in the corresponding upper field of the IMDUTY register.

Figure 24-4. Interlinked Multiple Value PWM Operation Flow



24.6.6 Open Drain Mode

Some pins can be used in open drain mode, allowing the PWMA waveform to toggle between 0V and up to 5V on these pins. In this mode the PWMA will drive the pin to zero or leave the output open. An external pullup can be used to pull the pin up to the desired voltage.

To enable open drain mode on a pin the PWMAOD function must be selected instead of the PWMA function in the I/O Controller. Please refer to the Module Configuration chapter for information about which pins are available in open drain mode.

24.6.7 Synchronization

Both the timebase counter and the spread spectrum counter can be reset and the duty cycle registers can be written through the user interface of the module. This requires a synchronization between the PB and GCLK clock domains, which takes a few clock cycles of each clock domain. The BUSY bit in SR indicates when the synchronization is ongoing. Writing to the module while the BUSY bit is set will result in discarding the new value.

Note that the duty cycle registers will not be updated with the new values until the timebase counter reaches its top value, in order to avoid glitches. The BUSY bit in SR will always be set during this updating and synchronization period.

24.6.8 Interrupts

When the timebase counter overflows, the Timebase Overflow bit in the Status Register (SR.TOFL) is set. If the corresponding bit in the Interrupt Mask Register (IMR) is set, an interrupt request will be generated.

Since the user needs to wait until the user interface is available between each write due to synchronization, a READY bit is provided in SR, which can be used to generate an interrupt request.

The interrupt request will be generated if the corresponding bit in 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).

24.6.9 Peripheral Events

24.6.9.1 Input Peripheral Events

The pre-defined channels support input peripheral events from the Peripheral Event System. An increase event (event_incr) will increase the duty cycle value by one, and a decrease event (event_decr) will decrease the duty cycle value by one. If an increase event and a decrease event occur at the same time, the duty cycle value will not be changed.

The number of channels supporting input peripheral events is device specific. Please refer to the Module Configuration section at the end of this chapter for details.

Input peripheral events must be enabled by writing a one to the corresponding bit in the Chanel Event Enable Register (CHEERm) before peripheral events can be used to control the duty cycle value. Each bit in the register corresponds to one channel, where bit 0 corresponds to channel 0 and so on. Both the increase and decrease events are enabled for the corresponding channel when a bit in the CHEERm register is written to one.

24.6.9.2 Output Peripheral Event

The PWMA also supports one output peripheral event (event_ch0) to the Peripheral Event System. This output peripheral event is connected to channel 0 and will be asserted when the timebase counter reaches the duty cycle value for channel 0. This output event is always enabled.

24.7 User Interface

Table 24-3. PWMA Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Read/Write	0x00000000
0x04	Interlinked Single Value Duty Register	ISDUTY	Write-only	0x00000000
0x08	Interlinked Multiple Value Duty Register	IMDUTY	Write-only	0x00000000
0x0C	Interlinked Multiple Value Channel Select	IMCHSEL	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	Status Register	SR	Read-only	0x00000000
0x20	Status Clear Register	SCR	Write-only	0x00000000
0x24	Parameter Register	PARAMETER	Read-only	- (1)
0x28	Version Register	VERSION	Read-only	- (1)
0x30	Interlinked Single Value Channel Set 0	ISCHSET0	Write-only	0x00000000
0x38	Channel Event Enable Register 0	CHEER0	Write-only	0x00000000
0x40	Interlinked Single Value Channel Set 1	ISCHSET1	Write-only	0x00000000
0x48	Channel Event Enable Register 1	CHEER1	Write-only	0x00000000

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

24.7.1 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
-	-	-	SPREAD				

15	14	13	12	11	10	9	8
TOP							

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

- **SPREAD: Spread Spectrum Limit Value**

  The spread spectrum limit value, together with the TOP field, defines the range for the spread spectrum counter. It is introduced in order to achieve constant varying duty cycles on the output PWM waveforms. Refer to Section24.6.3 for more information.
- **TOP: Timebase Counter Top Value**

  The top value for the timebase counter. The effective top value of the timebase counter is defined by both the TOP and the SPREAD fields. Refer to Section24.6.2 for more information.
- **TCLR: Timebase Clear**

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the timebase counter.

  This bit is always read as zero.
- **EN: Module Enable**

  0: The PWMA is disabled

  1: The PWMA is enabled

24.7.2 Interlinked Single Value Duty Register

Name: ISDUTY

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
-	-	-	-	-	-	-	-

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

7	6	5	4	3	2	1	0
DUTY							

- **DUTY: Duty Cycle Value**

  The duty cycle value written to this field is written simultaneously to all channels selected in the ISCHSET registers.

  If the value zero is written to DUTY all affected channels will be disabled. In this state the output waveform will be zero all the time.

24.7.3 Interlinked Multiple Value Duty Register

Name: IMDUTY

Access Type: Write-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DUTY3							

23	22	21	20	19	18	17	16
DUTY2							

15	14	13	12	11	10	9	8
DUTY1							

7	6	5	4	3	2	1	0
DUTY0							

- **DUTYn: Duty Cycle**

  The value written to DUTY field *n* will be written to the PWMA channel selected by the corresponding SEL field in the IMCHSEL register.

  If the value zero is written to DUTY all affected channels will be disabled. In this state the output waveform will be zero all the time.

24.7.4 Interlinked Multiple Value Channel Select

Name: IMCHSEL

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SEL3							

23	22	21	20	19	18	17	16
SEL2							

15	14	13	12	11	10	9	8
SEL1							

7	6	5	4	3	2	1	0
SEL0							

- **SELn: Channel Select**

  The duty cycle of the PWMA channel SELn will be updated with the value in the DUTYn field of the IMDUTY register when IMCHSEL is written. If SELn points to a non-implemented channel, the write will be discarded.

Note: The duty registers will be updated with the value stored in the IMDUTY register when the IMCHSEL register is written. Synchronization takes place immeidately when an IMCHSEL register is written. The duty cycle registers will, however, not be updated until the synchronization is completed and the timebase counter reaches its top value in order to avoid glitches.

24.7.5 Interrupt Enable Register

Name: IER

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
-	-	-	-	-	READY	-	TOFL

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.

24.7.6 Interrupt Disable Register

Name: IDR

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
-	-	-	-	-	READY	-	TOFL

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.

24.7.7 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
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	READY	-	TOFL

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.

24.7.8 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
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	BUSY	READY	-	TOFL

- **BUSY: Interface Busy**
  This bit is automatically cleared when the interface is no longer busy.
  This bit is set when the user interface is busy and will not respond to new write operations.
- **READY: Interface Ready**
  This bit is cleared by writing a one to the corresponding bit in the SCR register.
  This bit is set when the BUSY bit has a 1-to-0 transition.
- **TOFL: Timebase Overflow**
  This bit is cleared by writing a one to corresponding bit in the SCR register.
  This bit is set when the timebase counter has wrapped at its top value.

24.7.9 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
-	-	-	-	-	-	-	-
15	14	13	12	1	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	READY	-	TOFL

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.
This register always reads as zero.

24.7.10 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: Channels Implemented**
  This field contains the number of channels implemented on the device.

24.7.11 Version Register

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.

24.7.12 Interlinked Single Value Channel Set

Name: ISCHSETm

Access Type: Write-only

Offset: 0x30+m*0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SET							

23	22	21	20	19	18	17	16
SET							

15	14	13	12	11	10	9	8
SET							

7	6	5	4	3	2	1	0
SET							

- **SET: Single Value Channel Set**
  If the bit *n* in SET is one, the duty cycle of PWMA channel *n* will be updated with the value written to ISDUTY.
  If more than one ISCHSET register is present, ISCHSET0 controls channels 31 to 0 and ISCHSET1 controls channels 63 to 32.

Note: The duty registers will be updated with the value stored in the ISDUTY register when any ISCHSETm register is written. Synchronization takes place immeidately when an ISCHSET register is written. The duty cycle registers will, however, not be updated until the synchronization is completed and the timebase counter reaches its top value in order to avoid glitches.

24.7.13 Channel Event Enable Register

Name: CHEERm

Access Type: Write-only

Offset: 0x38+m*0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CHEE							

23	22	21	20	19	18	17	16
CHEE							

15	14	13	12	11	10	9	8
CHEE							

7	6	5	4	3	2	1	0
CHEE							

- **CHEE: Channel Event Enable**

  0: The input peripheral event for the corresponding channel is disabled.

  1: The input peripheral event for the corresponding channel is enabled.

  Both increase and decrease events for channel n are enabled if bit n is one.

  If more than one CHEER register is present, CHEER0 controls channels 31-0 and CHEER1 controls channels 64-32 and so on.

24.8 Module Configuration

The specific configuration for each PWMA 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 24-4. PWMA Configuration

Feature	PWMA
Number of PWM channels	36
Channels supporting incoming peripheral events	0, 6, 8, 9, 11, 14, 19, and 20
PWMA channels with Open Drain mode	21, 27, and 28

Table 24-5. PWMA Clocks

Clock Name	Descripton
CLK_PWMA	Clock for the PWMA bus interface
GCLK	The generic clock used for the PWMA is GCLK3

Table 24-6. Register Reset Values

Register	Reset Value
VERSION	0x00000101
PARAMETER	0x00000024

25. Timer/Counter (TC)

Rev: 2.2.3.1.1

25.1 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 lines in capture mode**

25.2 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.

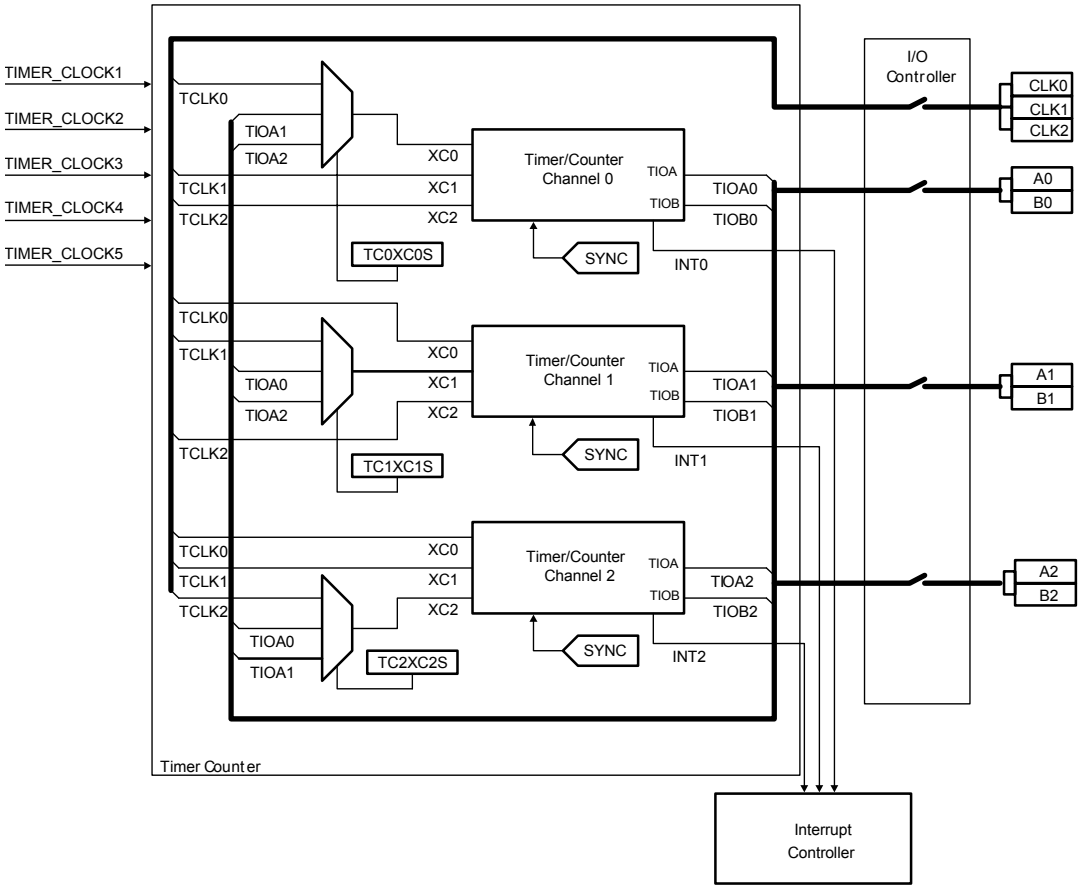
The TC block has two global registers which act upon all three TC channels.

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.

25.3 Block Diagram

Figure 25-1. TC Block Diagram



25.4 I/O Lines Description

Table 25-1. 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

25.5 Product Dependencies

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

25.5.1 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 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.

25.5.2 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.

25.5.3 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.

25.5.4 Interrupts

The TC interrupt request line is connected to the interrupt controller. Using the TC interrupt requires the interrupt controller to be programmed first.

25.5.5 Peripheral Events

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

25.5.6 Debug Operation

The Timer Counter clocks are frozen during debug operation, unless the OCD system keeps peripherals running in debug operation.

25.6 Functional Description

25.6.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Figure 25-3 on page 575.

25.6.1.1 Channel I/O Signals

As described in Figure 25-1 on page 559, each Channel has the following I/O signals.

Table 25-2. 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

25.6.1.2 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.

25.6.1.3 *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 25-2 on page 561.

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.

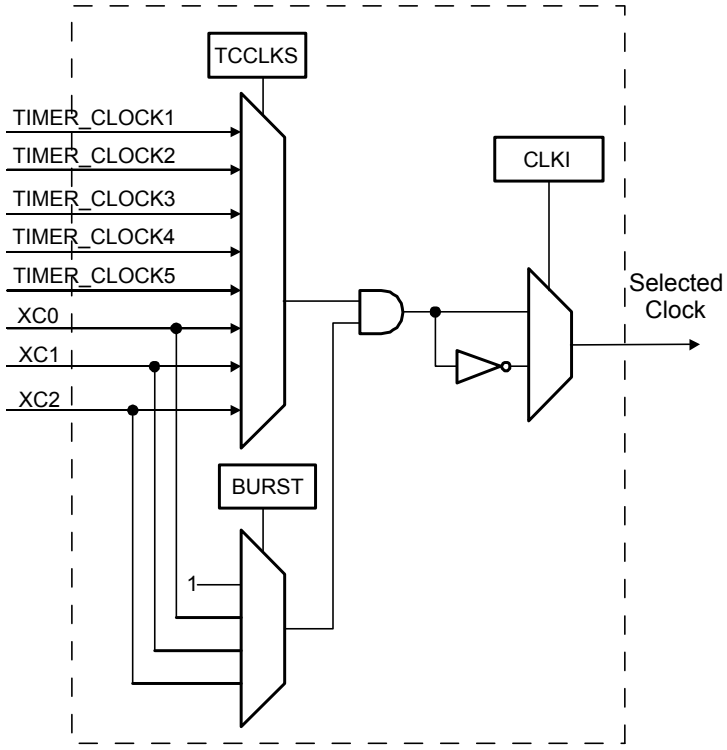
This selection is made by the Clock Selection field in the Channel n Mode Register (CMRn.TCCLKS).

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 25-2. Clock Selection

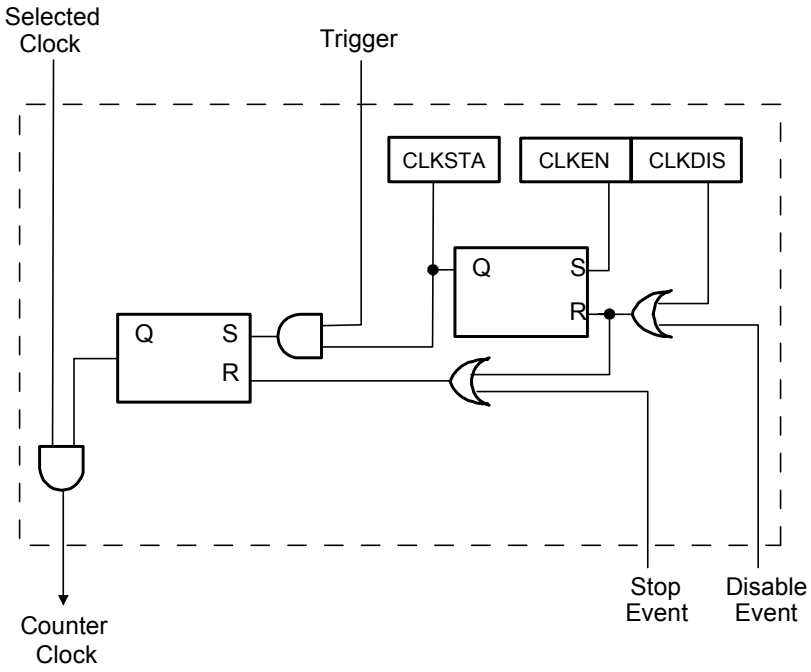


25.6.1.4 Clock control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 25-3 on page 562.

- 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 25-3. Clock Control



25.6.1.5 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.

25.6.1.6 *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.

25.6.1.7 *Peripheral events on TIOA inputs*

The TIOA 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 line are disabled. When capturing on the external TIOA pin the user must ensure that no peripheral events are generated on this pin.

25.6.2 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 25-4 on page 565 shows the configuration of the TC channel when programmed in Capture mode.

25.6.2.1 *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.

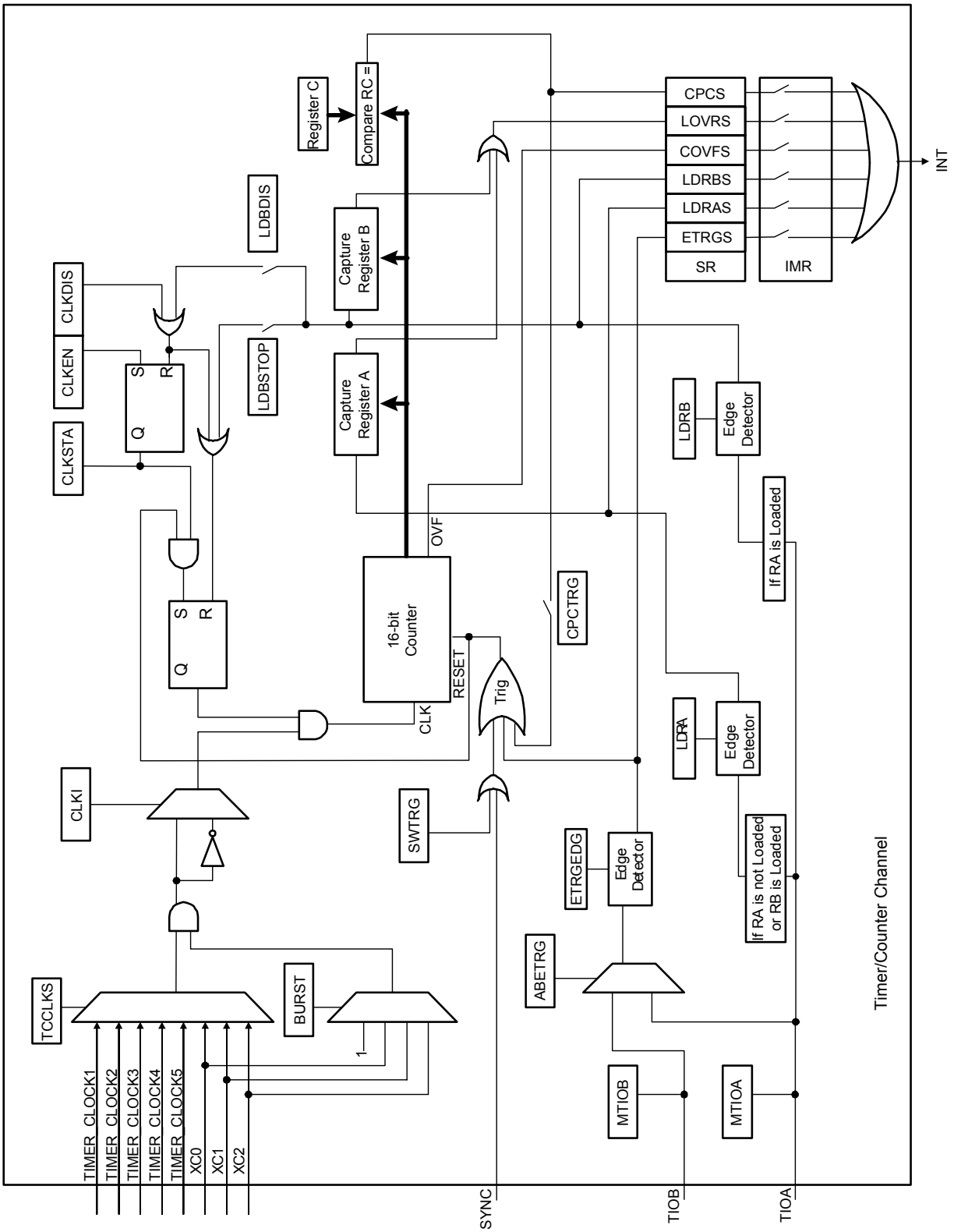
Loading RA or RB before the read of the last value loaded sets the Load Overrun Status bit in SRn (SRn.LOVRS). In this case, the old value is overwritten.

25.6.2.2 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 25-4. Capture Mode



25.6.3 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 25-5 on page 567 shows the configuration of the TC channel when programmed in Waveform operating mode.

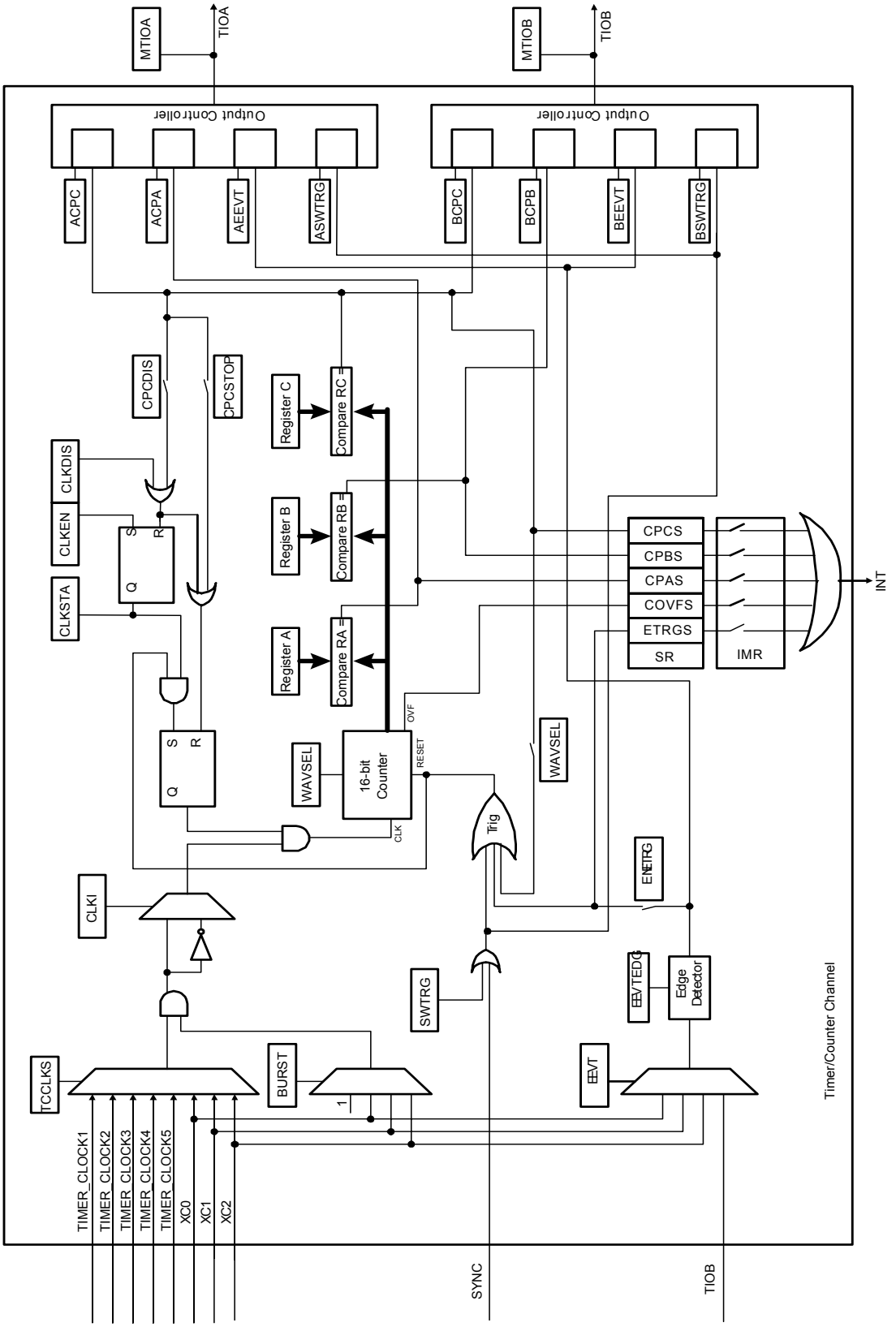
25.6.3.1 Waveform selection

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 25-5. Waveform Mode



25.6.3.2 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 25-6 on page 568.

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 25-7 on page 569.

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 25-6. WAVSEL= 0 Without Trigger

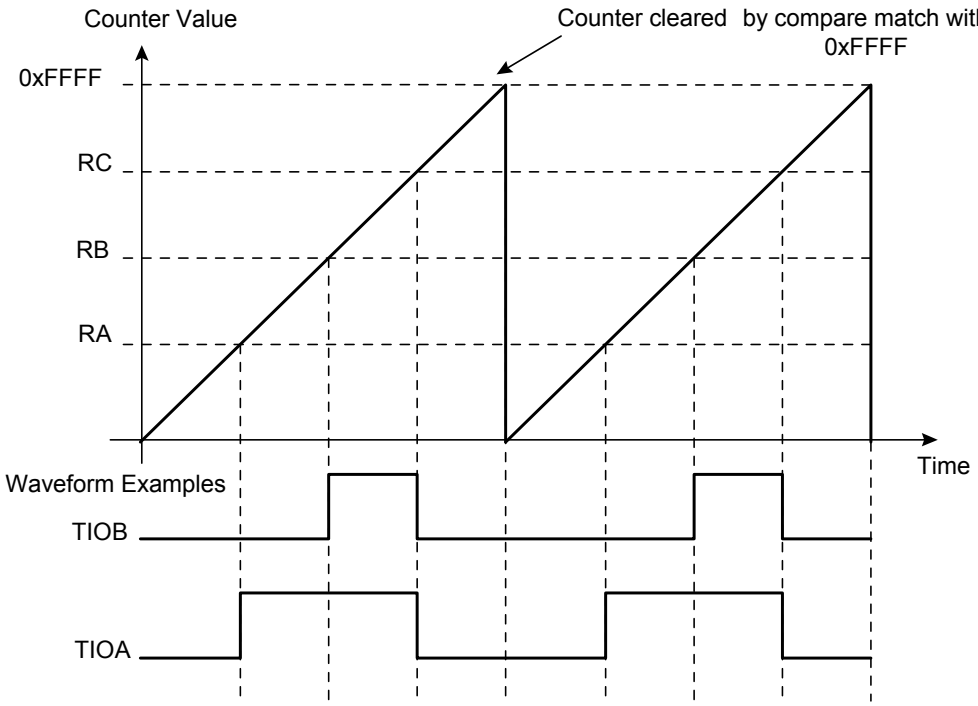
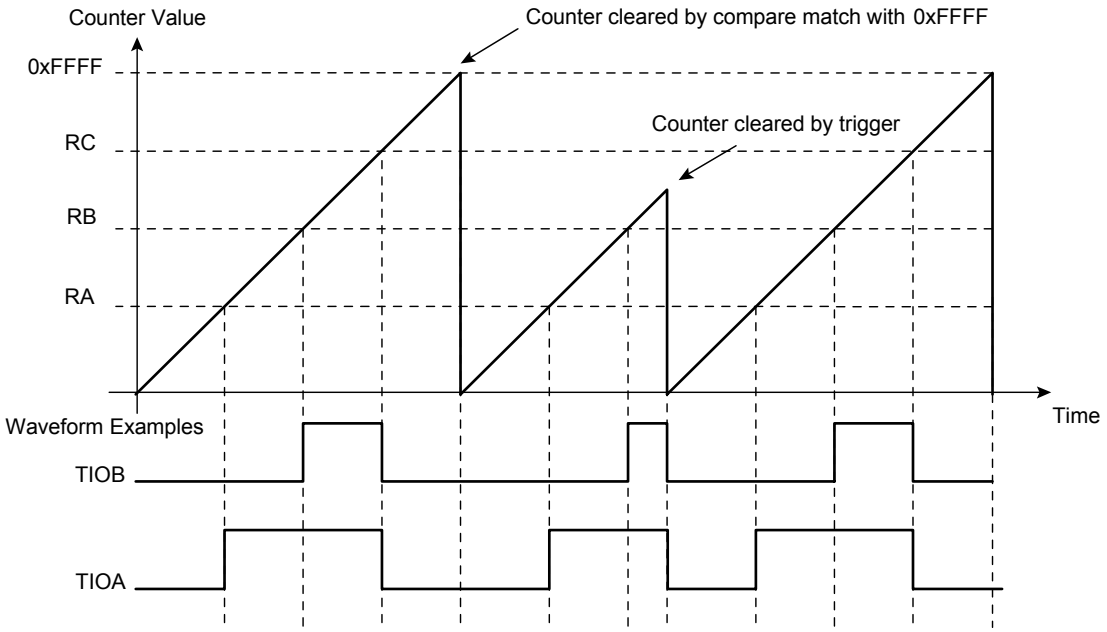


Figure 25-7. WAVSEL= 0 With Trigger



25.6.3.3 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 25-8 on page 570.

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 25-9 on page 570.

In addition, RC Compare can stop the counter clock (CMRn.CPCSTOP) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 25-8. WAVSEL = 2 Without Trigger

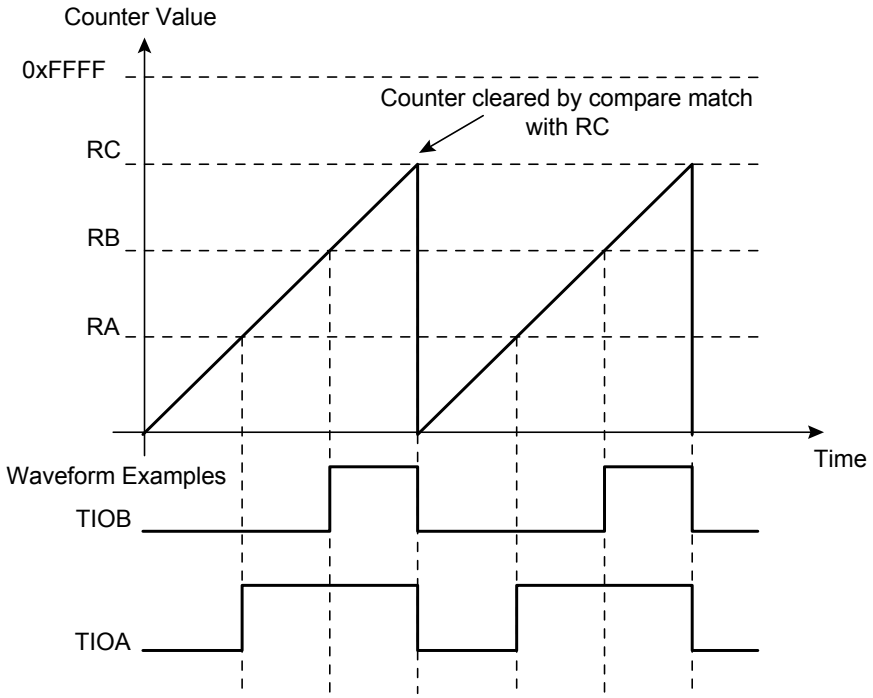
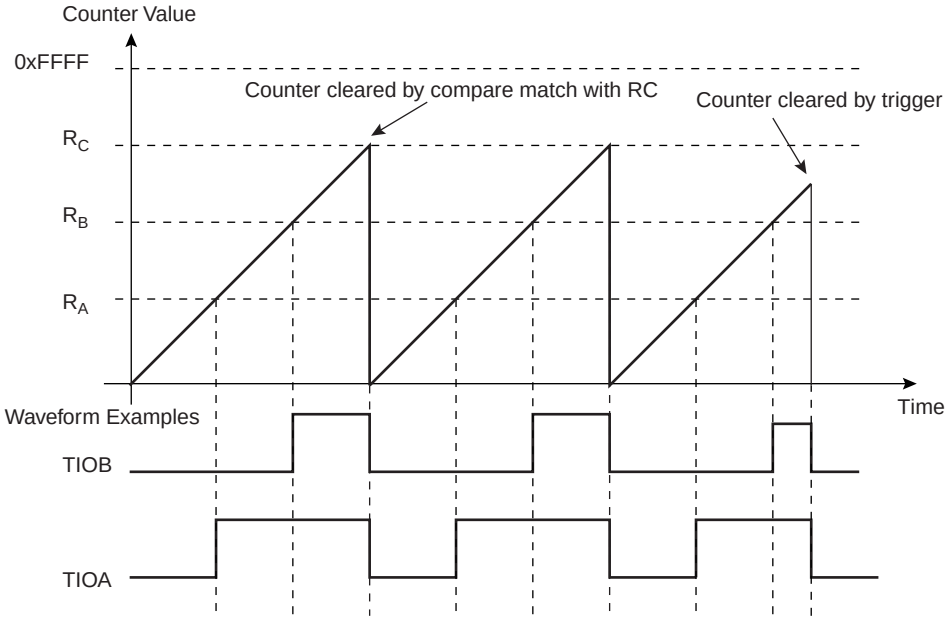


Figure 25-9. WAVSEL = 2 With Trigger



25.6.3.4 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 25-10 on page 571.

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 25-11 on page 571.

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 25-10. WAVSEL = 1 Without Trigger

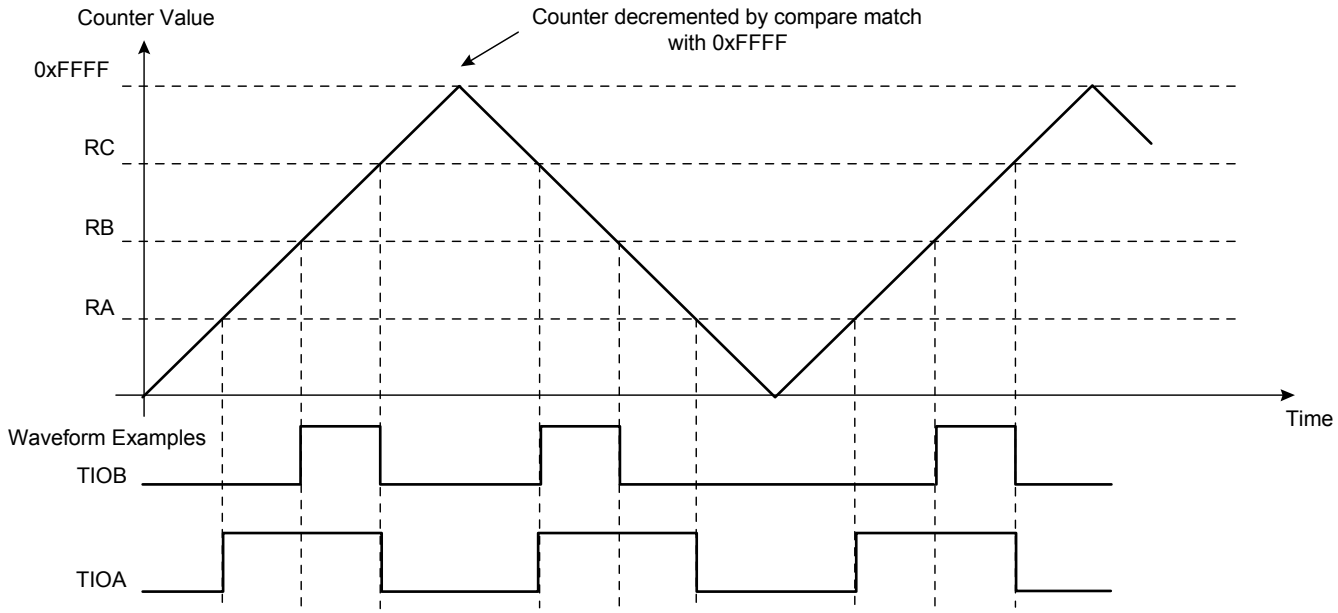
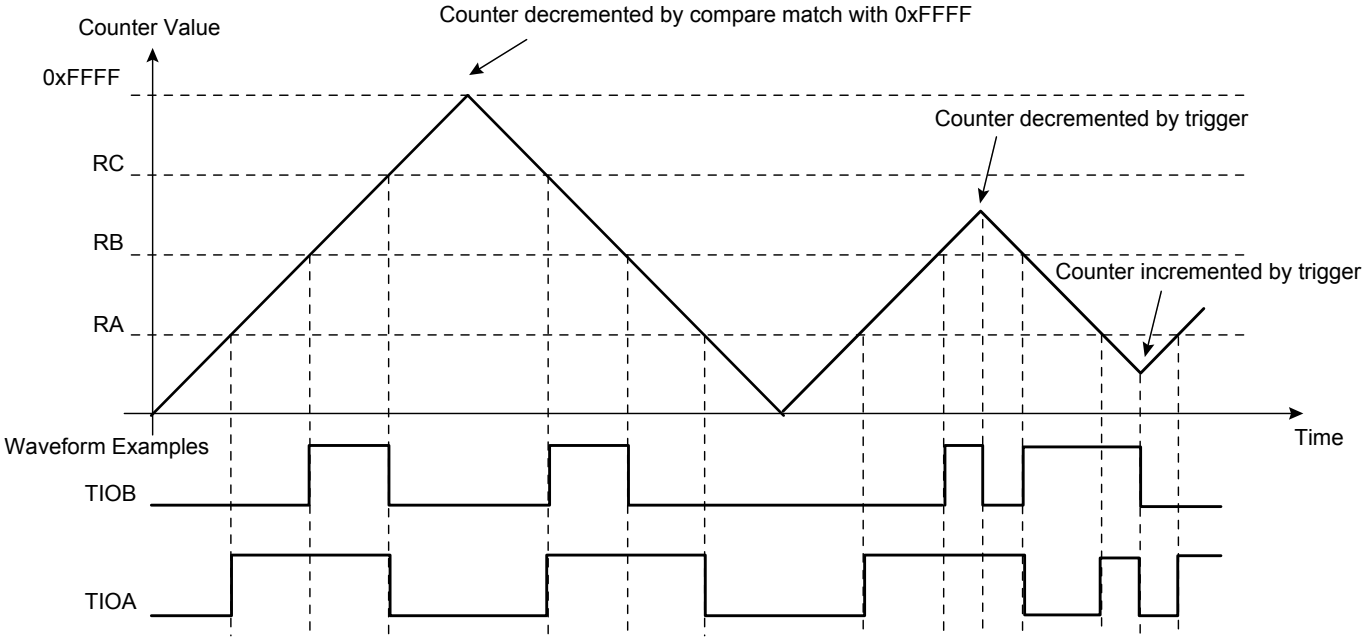


Figure 25-11. WAVSEL = 1 With Trigger



25.6.3.5 *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 25-12 on page 572.

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 25-13 on page 573.

RC Compare can stop the counter clock (CMRn.CPCSTOP = 1) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 25-12. WAVSEL = 3 Without Trigger

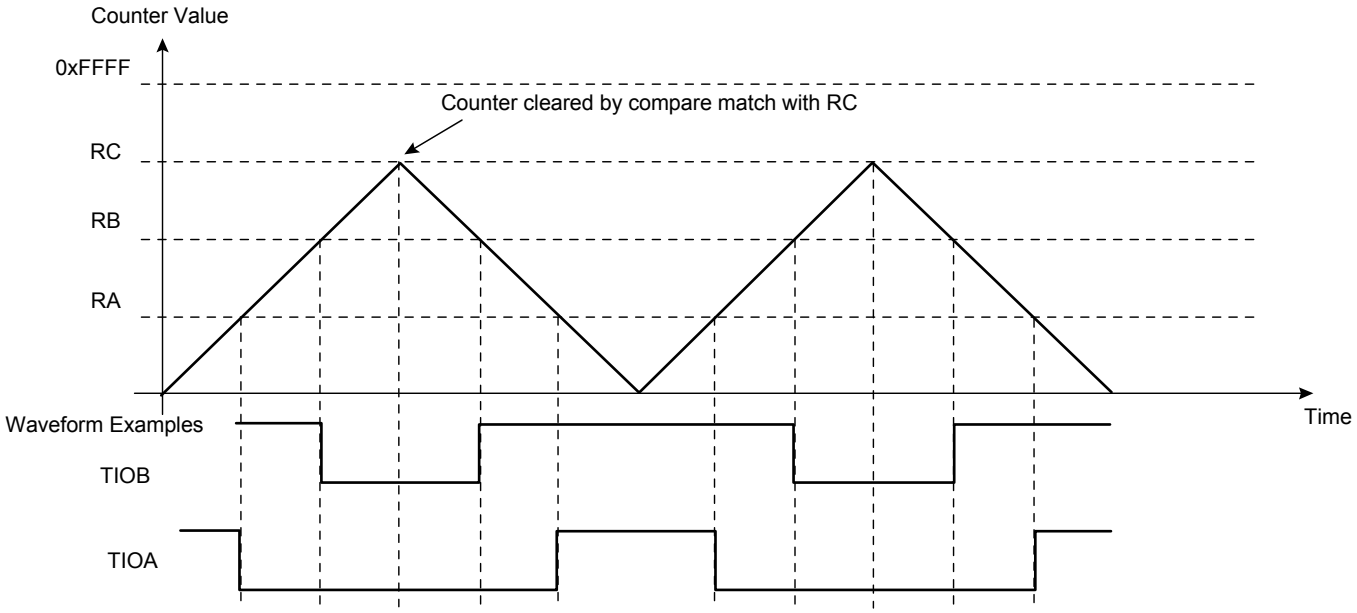
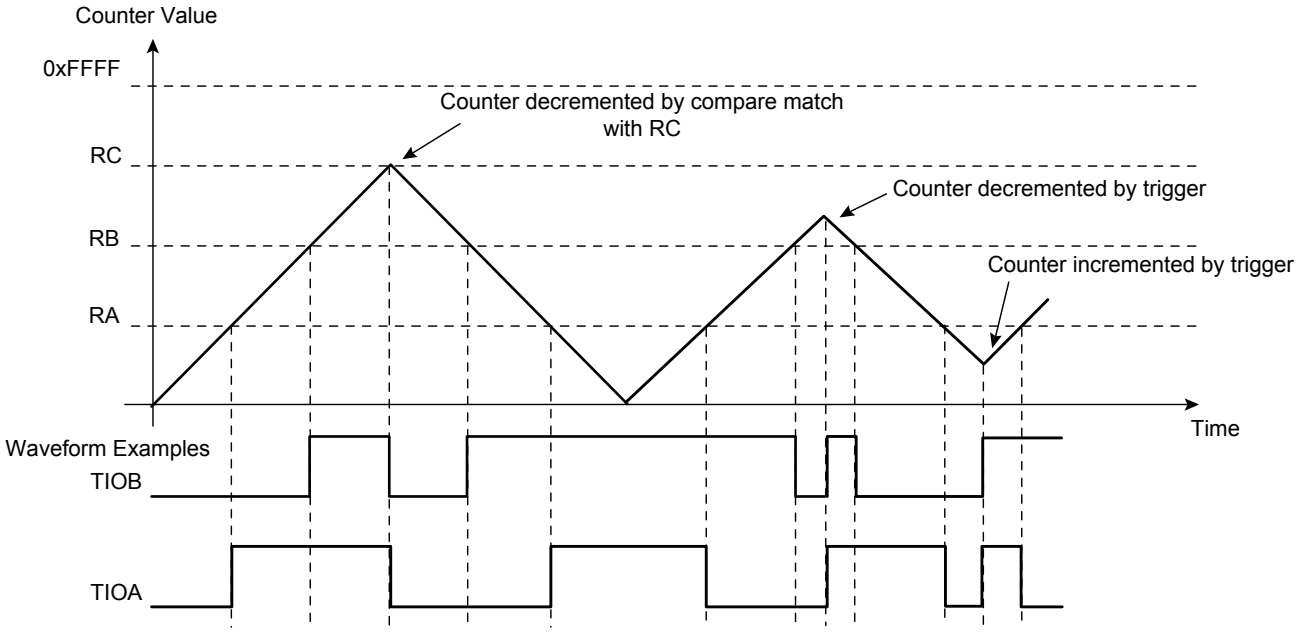


Figure 25-13. WAVSEL = 3 With Trigger



25.6.3.6 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.

25.6.3.7 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)

25.7 User Interface

Table 25-3. 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	00x00000000
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

Notes: 1. Read-only if CMRn.WAVE is zero

25.7.1 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.

25.7.2 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

25.7.3 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

25.7.4 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.

25.7.5 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.

25.7.6 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.

25.7.7 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.

25.7.8 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.

25.7.9 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.

25.7.10 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.

25.7.11 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.

25.7.12 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.

25.7.13 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

25.8 Module Configuration

25.8.1 Clock Connections

Each Timer/Counter channel can independently select an internal or external clock source for its counter:

Table 25-4. Timer/Counter Clock Connections

Module	Source	Name	Connection
TC0	Internal	TIMER_CLOCK1	32 KHz oscillator clock (CLK_32K)
		TIMER_CLOCK2	PBA Clock / 2
		TIMER_CLOCK3	PBA Clock / 8
		TIMER_CLOCK4	PBA Clock / 32
		TIMER_CLOCK5	PBA Clock / 128
	External	XC0	See Section 3.2 on page 9
		XC1	
		XC2	
TC1	Internal	TIMER_CLOCK1	32 KHz oscillator clock (CLK_32K)
		TIMER_CLOCK2	PBA Clock / 2
		TIMER_CLOCK3	PBA Clock / 8
		TIMER_CLOCK4	PBA Clock / 32
		TIMER_CLOCK5	PBA Clock / 128
	External	XC0	See Section 3.2 on page 9
		XC1	
		XC2	
TC2	Internal	TIMER_CLOCK1	32 KHz oscillator clock (CLK_32K)
		TIMER_CLOCK2	PBA Clock / 2
		TIMER_CLOCK3	PBA Clock / 8
		TIMER_CLOCK4	PBA Clock / 32
		TIMER_CLOCK5	PBA Clock / 128
	External	XC0	See Section 3.2 on page 9
		XC1	
		XC2	

26. ADC Interface (ADCIFB)

Rev.:1.0.1.1

26.1 Features

- **Multi-channel Analog-to-Digital Converter with up to 12-bit resolution**
- **Enhanced Resolution Mode**
  - **11-bit resolution obtained by interpolating 4 samples**
  - **12-bit resolution obtained by interpolating 16 samples**
- **Glueless interface with resistive touch screen panel, allowing**
  - **Touch Screen position measurement**
  - **Pen detection and pen loss detection**
- **Integrated enhanced sequencer**
  - **ADC Mode**
  - **Touch Screen Mode**
- **Numerous trigger sources**
  - **Software**
  - **Embedded 16-bit timer for periodic trigger**
  - **Pen detect trigger**
  - **Continuous trigger**
  - **External trigger, rising, falling, or any-edge trigger**
  - **Peripheral event trigger**
- **ADC Sleep Mode for low power ADC applications**
- **Programmable ADC timings**
  - **Programmable ADC clock**
  - **Programmable startup time**

26.2 Overview

The ADC Interface (ADCIFB) converts analog input voltages to digital values. The ADCIFB is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). The conversions extend from 0V to ADVREFP.

The ADCIFB supports 8-bit and 10-bit resolution mode, in addition to enhanced resolution mode with 11-bit and 12-bit resolution. Conversion results are reported in a common register for all channels.

The 11-bit and 12-bit resolution modes are obtained by interpolating multiple samples to acquire better accuracy. For 11-bit mode 4 samples are used, which gives an effective sample rate of 1/4 of the actual sample frequency. For 12-bit mode 16 samples are used, giving a effective sample rate of 1/16 of actual. This arrangement allows conversion speed to be traded for better accuracy.

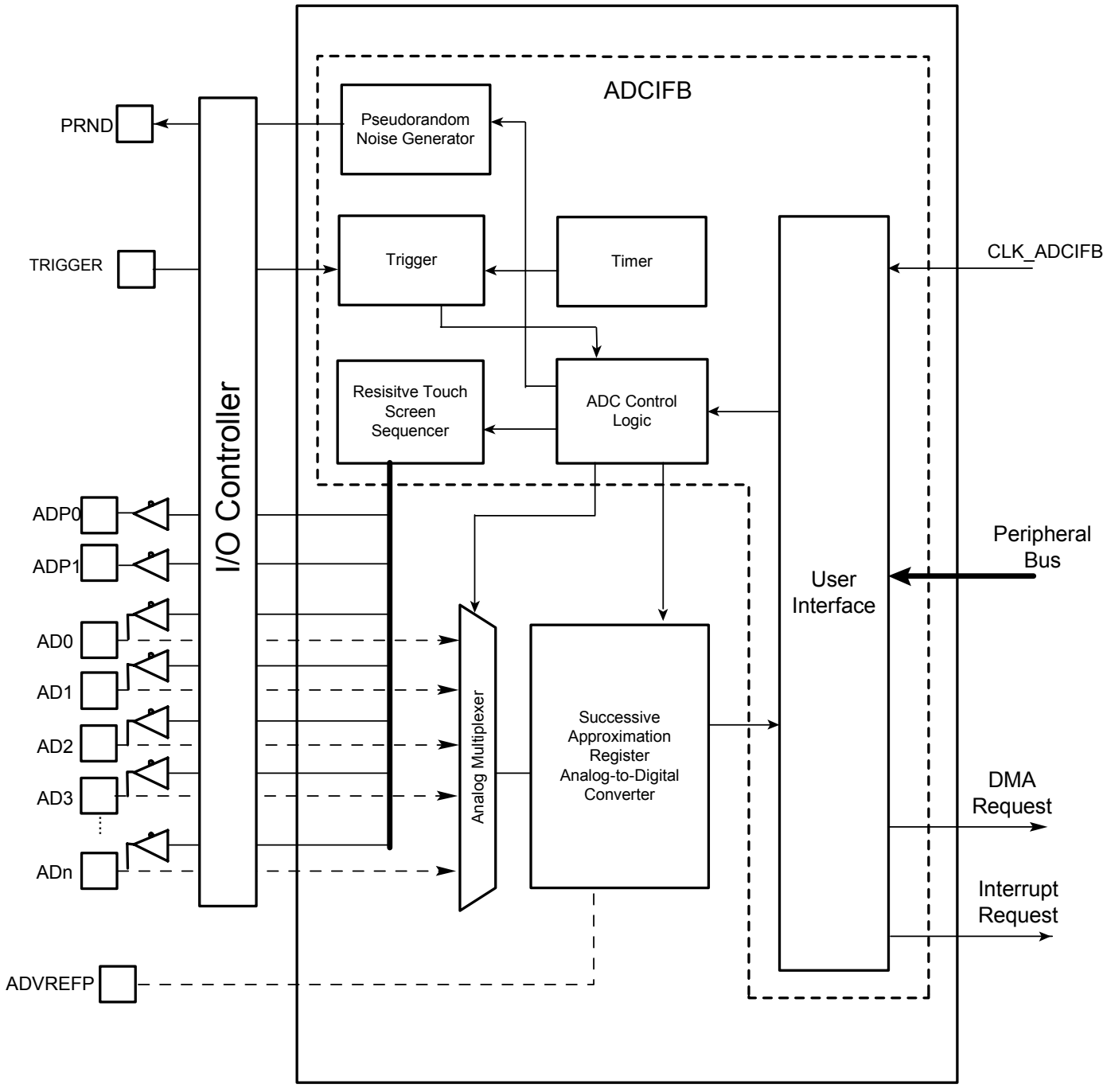
Conversions can be started for all enabled channels, either by a software trigger, by detection of a level change on the external trigger pin (TRIGGER), or by an integrated programmable timer.

When the Touch Screen ADC Mode is enabled, an integrated sequencer automatically configures the pad control signals and performs touch screen conversions.

The ADCIFB also integrates an ADC Sleep Mode, a Pen-Detect Mode, and an Analog Compare Mode, and connects with one Peripheral DMA Controller channel. These features reduce both power consumption and processor intervention.

26.3 Block Diagram

Figure 26-1. ADCIFB Block Diagram



26.4 I/O Lines Description

Table 26-1. I/O Lines Description

Pin Name	Description	Type
ADVREFP	Reference voltage	Analog
TRIGGER	External trigger	Digital
ADP0	Drive Pin 0 for Touch Screen top channel (Xp)	Digital
ADP1	Drive Pin 1 for Touch Screen right channel (Yp)	Digital
AD0-ADn	Analog input channels 0 to n	Analog

26.5 Product Dependencies

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

26.5.1 I/O Lines

The analog input pins can be multiplexed with I/O Controller lines. The user must make sure the I/O Controller is configured correctly to allow the ADCIFB access to the AD pins before the ADCIFB is instructed to start converting data. If the user fails to do this the converted data may be wrong.

The number of analog inputs is device dependent, please refer to the ADCIFB Module Configuration chapter for the number of available AD inputs on the current device.

The ADVREFP pin must be connected correctly prior to using the ADCIFB. Failing to do so will result in invalid ADC operation. See the Electrical Characteristics chapter for details.

If the TRIGGER, ADP0, and ADP1 pins are to be used in the application, the user must configure the I/O Controller to assign the needed pins to the ADCIFB function.

26.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the ADCIFB, the ADCIFB will stop functioning and resume operation after the system wakes up from sleep mode.

If the Peripheral Event System is configured to send asynchronous peripheral events to the ADCIFB and the clock used by the ADCIFB is stopped, a local and temporary clock will automatically be requested so the event can be processed. Refer to Section 26.6.13, Section 26.6.12, and the Peripheral Event System chapter for details.

Before entering a sleep mode where the clock to the ADCIFB is stopped, make sure the Analog-to-Digital Converter cell is put in an inactive state. Refer to Section 26.6.13 for more information.

26.5.3 Clocks

The clock for the ADCIFB bus interface (CLK_ADCIFB) 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 ADCIFB before disabling the clock, to avoid freezing the ADCIFB in an undefined state.

26.5.4 Interrupts

The ADCIFB interrupt request line is connected to the interrupt controller. Using the ADCIFB interrupt request functionality requires the interrupt controller to be programmed first.

26.5.5 Peripheral Events

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

26.5.6 Debug Operation

When an external debugger forces the CPU into debug mode, this module 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.

26.6 Functional Description

The ADCIFB embeds a Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC). The ADC supports 8-bit or 10-bit resolution, which can be extended to 11 or 12 bits by the Enhanced Resolution Mode.

The conversion is performed on a full range between 0V and the reference voltage pin ADVREFP. Analog inputs between these voltages converts to digital values (codes) based on a linear conversion. This linear conversion is described in the expression below where M is the number of bits used to represent the analog value, Vin is the voltage of the analog value to convert, Vref is the maximum voltage, and Code is the converted digital value.

$$Code = \frac{2^M \cdot V_{in}}{V_{ref}}$$

26.6.1 Initializing the ADCIFB

The ADC Interface is enabled by writing a one to the Enable bit in the Control Register (CR.EN). After the ADC Interface is enabled, the ADC timings needs to be configured by writing the correct values to the RES, PRESCAL, and STARTUP fields in the ADC Configuration Register (ACR). See Section 26.6.5, and Section 26.6.7 for details. Before the ADCIFB can be used, the I/O Controller must be configured correctly and the Reference Voltage (ADVREFP) signal must be connected. Refer to Section 26.5.1 for details.

26.6.2 Basic Operation

To convert analog values to digital values the user must first initialize the ADCIFB as described in Section 26.6.1. When the ADCIFB is initialized the channels to convert must be enabled by writing a one the corresponding bits in the Channel Enable Register (CHER). Enabling channel N instructs the ADCIFB to convert the analog voltage applied to AD pin N at each conversion sequence. Multiple channels can be enabled resulting in multiple AD pins being converted at each conversion sequence.

To start converting data the user can either manually start a conversion sequence by writing a one to the START bit in the Control Register (CR.START) or configure an automatic trigger to initiate the conversions. The automatic trigger can be configured to trig on many different conditions. Refer to Section 26.8.1 for details.

The result of the conversion is stored in the Last Converted Data Register (LCDR) as they become available, overwriting the result from the previous conversion. To avoid data loss if more than one channel is enabled, the user must read the conversion results as they become available either by using an interrupt handler or by using a Peripheral DMA channel to copy the results to memory. Failing to do so will result in an Overrun Error condition, indicated by the OVRE bit in the Status Register (SR).

To use an interrupt handler the user must enable the Data Ready (DRDY) interrupt request by writing a one to the corresponding bit in the Interrupt Enable Register (IER). To clear the interrupt after the conversion result is read, the user must write a one to the corresponding bit in the Interrupt Clear Register (ICR). See Section 26.6.11 for details.

To use a Peripheral DMA Controller channel the user must configure the Peripheral DMA Controller appropriately. The Peripheral DMA Controller will, when configured, automatically read converted data as they become available. There is no need to manually clear any bits in the Interrupt Status Register as this is performed by the hardware. If an Overrun Error condition happens during DMA operation, the OVRE bit in the SR will be set.

26.6.3 ADC Resolution

The Analog-to-Digital Converter cell supports 8-bit or 10-bit resolution, which can be extended to 11-bit and 12-bit with the Enhanced Resolution Mode. The resolution is selected by writing the selected resolution value to the RES field in the ADC Configuration Register (ACR). See Section 26.9.3.

By writing a zero to the RES field, the ADC switches to the lowest resolution and the conversion results can be read in the eight lowest significant bits of the Last Converted Data Register (LCDR). The four highest bits of the Last Converted Data (LDATA) field in the LCDR register reads as zero. Writing a one to the RES field enables 10-bit resolution, the optimal resolution for both sampling speed and accuracy. Writing two or three automatically enables Enhanced Resolution Mode with 11-bit or 12-bit resolution, see Section 26.6.4 for details.

When a Peripheral DMA Controller channel is connected to the ADCIFB in 10-bit, 11-bit, or 12-bit resolution mode, a transfer size of 16 bits must be used. By writing a zero to the RES field, the destination buffers can be optimized for 8-bit transfers.

26.6.4 Enhanced Resolution Mode

The Enhanced Resolution Mode is automatically enabled when 11-bit or 12-bit mode is selected in the ADC Configuration Register (ACR). In this mode the ADCIFB will trade conversion performance for accuracy by averaging multiple samples.

To be able to increase the accuracy by averaging multiple samples it is important that some noise is present in the input signal. The noise level should be between one and two LSB peak-to-peak to get good averaging performance.

The performance cost of enabling 11-bit mode is 4 ADC samples, which reduces the effective ADC performance by a factor 4. For 12-bit mode this factor is 16. For 12-bit mode the effective sample rate is maximum ADC sample rate divided by 16.

26.6.5 ADC Clock

The ADCIFB generates an internal clock named CLK_ADC that is used by the Analog-to-Digital Converter cell to perform conversions. The CLK_ADC frequency is selected by writing to the PRESCAL field in the ADC Configuration Register (ACR). The CLK_ADC range is between CLK_ADCIFB/2, if PRESCAL is 0, and CLK_ADCIFB/128, if PRESCAL is 63 (0x3F).

A sensible PRESCAL value must be used in order to provide an ADC clock frequency according to the maximum sampling rate parameter given in the Electrical Characteristics section. Failing to do so may result in incorrect Analog-to-Digital Converter operation.

26.6.6 ADC Sleep Mode

The ADC Sleep Mode maximizes power saving by automatically deactivating the Analog-to-Digital Converter cell when it is not being used for conversions. The ADC Sleep Mode is enabled by writing a one to the SLEEP bit in the ADC Configuration Register (ACR).

When a trigger occurs while the ADC Sleep Mode is enabled, the Analog-to-Digital Converter cell is automatically activated. As the analog cell requires a startup time, the logic waits during this time and then starts the conversion of the enabled channels. When conversions of all enabled channels are complete, the ADC is deactivated until the next trigger.

26.6.7 Startup Time

The Analog-to-Digital Converter cell has a minimal startup time when the cell is activated. This startup time is given in the Electrical Characteristics chapter and must be written to the STARTUP field in the ADC Configuration Register (ACR) to get correct conversion results.

The STARTUP field expects the startup time to be represented as the number of CLK_ADC cycles between 8 and 1024 and in steps of 8 that is needed to cover the ADC startup time as specified in the Electrical Characteristics chapter.

The Analog-to-Digital Converter cell is activated at the first conversion after reset and remains active if ACR.SLEEP is zero. If ACR.SLEEP is one, the Analog-to-Digital Converter cell is automatically deactivated when idle and thus each conversion sequence will have a initial startup time delay.

26.6.8 Sample and Hold Time

A minimal Sample and Hold Time is necessary for the ADCIFB to guarantee the best converted final value when switching between ADC channels. This time depends on the input impedance of the analog input, but also on the output impedance of the driver providing the signal to the analog input, as there is no input buffer amplifier.

The Sample and Hold time has to be programmed through the SHTIM field in the ADC Configuration Register (ACR). This field can define a Sample and Hold time between 1 and 16 CLK_ADC cycles.

26.6.9 ADC Conversion

ADC conversions are performed on all enabled channels when a trigger condition is detected. For details regarding trigger conditions see Section 26.8.1. The term channel is used to identify a specific analog input pin so it can be included or excluded in an Analog-to-Digital conversion sequence and to identify which AD pin was used to convert the current value in the Last Converted Data Register (LCDR). Channel number N corresponding to AD pin number N.

Channels are enabled by writing a one to the corresponding bit in the Channel Enable Register (CHER), and disabled by writing a one to the corresponding bit in the Channel Disable Register (CHDR). Active channels are listed in the Channel Status Register (CHSR).

When a conversion sequence is started, all enabled channels will be converted in one sequence and the result will be placed in the Last Converted Data Register (LCDR) with the channel number used to produce the result. It is important to read out the results while the conversion

sequence is ongoing, as new values will automatically overwrite any old value and the old value will be lost if not previously read by the user.

If the Analog-to-Digital Converter cell is inactive when starting a conversion sequence, the conversion logic will wait a configurable number of CLK_ADC cycles as defined in the startup time field in the ADC Configuration Register (ACR). After the cell is activated all enabled channels is converted one by one until no more enabled channels exist. The conversion sequence converts each enabled channel in order starting with the channel with the lowest channel number. If the ACR.SLEEP bit is one, the Analog-to-Digital Converter cell is deactivated after the conversion sequence has finished.

For each channel converted, the ADCIFB waits a Sample and Hold number of CLK_ADC cycles as defined in the SHTIM field in ACR, and then instructs the Analog-to-Digital Converter cell to start converting the analog voltage. The ADC cell requires 10 CLK_ADC cycles to actually convert the value, so the total time to convert a channel i Sample and Hold + 10 CLK_ADC cycles.

26.6.10 Analog Compare Mode

The ADCIFB can test if the converted values, as they become available, are below, above, or inside a specified range and generate interrupt requests based on this information. This is useful for applications where the user wants to monitor some external analog signal and only initiate actions if the value is above, below, or inside some specified range.

The Analog Compare mode is enabled by writing a one to the Analog Compare Enable (ACE) bit in the Mode Register (MR). The values to compare must be written to the Low Value (LV) field and the High Value (HV) field in the Compare Value Register (CVR). The Analog Compare mode will, when enabled, check all enabled channels against the pre-programmed high and low values and set status bits.

To generate an interrupt request if a converted value is below a limit, write the limit to the CVR.LV field and enable interrupt request on the Compare Lesser Than (CLT) bit by writing a one to the corresponding bit in the Interrupt Enable Register (IER). To generate an interrupt request if a converted value is above a limit, write the limit to the CVR.HV field and enable interrupt for Compare Greater Than (CGT) bit. To generate an interrupt request if a converted value is inside a range, write the low and high limit to the LV and HV fields and enable the Compare Else (CELSE) interrupt. To generate an interrupt request if a value is outside a range, write the LV and HV fields to the low and high limits of the range and enable CGT and CLT interrupts.

Note that the values written to LV and HV must match the resolution selected in the ADC Configuration Register (ACR).

26.6.11 Interrupt Operation

Interrupt requests are 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). Enabled interrupts can be read from the Interrupt Mask Register (IMR). Active interrupt requests, but potentially masked, are visible in the Interrupt Status Register (ISR). To clear an active interrupt request, write a one to the corresponding bit in the Interrupt Clear Register (ICR).

The source for the interrupt requests are the status bits in the Status Register (SR). The SR shows the ADCIFB status at the time the register is read. The Interrupt Status Register (ISR) shows the status since the last write to the Interrupt Clear Register. The combination of ISR and SR allows the user to react to status change conditions but also allows the user to read the current status at any time.

26.6.12 Peripheral Events

The Peripheral Event System can be used together with the ADCIFB to allow any peripheral event generator to be used as a trigger source. To enable peripheral events to trigger a conversion sequence the user must write the Peripheral Event Trigger value (0x7) to the Trigger Mode (TRGMOD) field in the Trigger Register (TRGR). Refer to Table 26-4 on page 615. The user must also configure a peripheral event generator to emit peripheral events for the ADCIFB to trigger on. Refer to the Peripheral Event System chapter for details.

26.6.13 Sleep Mode

Before entering sleep modes the user must make sure the ADCIFB is idle and that the Analog-to-Digital Converter cell is inactive. To deactivate the Analog-to-Digital Converter cell the SLEEP bit in the ADC Configuration Register (ACR) must be written to one and the ADCIFB must be idle. To make sure the ADCIFB is idle, write a zero the Trigger Mode (TRGMOD) field in the Trigger Register (TRGR) and wait for the READY bit in the Status Register (SR) to be set.

Note that by deactivating the Analog-to-Digital Converter cell, a startup time penalty as defined in the STARTUP field in the ADC Configuration Register (ACR) will apply on the next conversion.

26.6.14 Conversion Performances

For performance and electrical characteristics of the ADCIFB, refer to the Electrical Characteristics chapter.

26.7 Resistive Touch Screen

The ADCIFB embeds an integrated resistive touch screen sequencer that can be used to calculate contact coordinates on a resistive film touch screen. When instructed to start, the integrated touch screen sequencer automatically applies a sequence of voltage patterns to the touch screen films and the Analog-to-Digital Conversion cell is used to measure the effects. The resulting measurements can be used to calculate the horizontal and vertical contact coordinates. It is recommended to use a high resistance touch screen for optimal resolution.

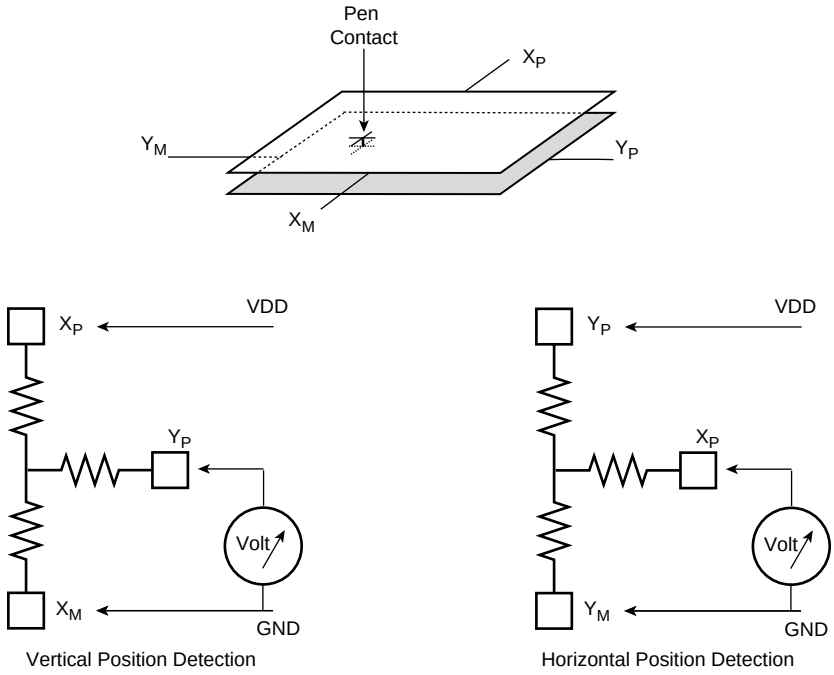
The touch screen film is connected to the ADCIFB using the AD and ADP pins. See Section 26.7.3 for details.

Touch Screen ADC Mode is enabled by writing a one to the TSAMOD field in the Mode Register (MR). In this mode, channels TSPO+0 though TSPO+3 are automatically enabled where TSPO refers to the Touch Screen Pin Offset field in the Mode Register. For each conversion sequence, all enabled channels before TSPO+0 and after TSPO+3 are converted as ordinary ADC channels, producing 1 conversion result each. When the sequencer enters the TSPO+0 channel the touch screen sequencer will take over control and convert the next 4 channels as described in Section 26.7.4.

26.7.1 Resistive Touch Screen Principles

A resistive touch screen is based on two resistive films, each one fitted with a pair of electrodes, placed at the top and bottom on one film, and on the right and left on the other. Between the two, there is a layer that acts as an insulator, but makes a connection when pressure is applied to the screen. This is illustrated in Figure 26-2 on page 604.

Figure 26-2. Touch Screen Position Measurement



26.7.2 Position Measurement Method

As shown in Figure 26-2 on page 604, to detect the position of a contact, voltage is first applied to X_P (top) and X_m (bottom) leaving Y_p and Y_m tristated. Due to the linear resistance of the film,

there is a voltage gradient from top to bottom on the first film. When a contact is performed on the screen, the voltage at the contact point propagates to the second film. If the input impedance on the Y_P (right) and Y_m (left) electrodes are high enough, no current will flow, allowing the voltage at the contact point to be measured at Y_p. The value measured represents the vertical position component of the contact point.

For the horizontal direction, the same method is used, but by applying voltage from Y_P (right) to Y_m (left) and measuring at X_P.

In an ideal world (linear, with no loss), the vertical position is equal to:

VY_P / VDD

To compensate for some of the real world imperfections, VX_P and VX_m can be measured and used to improve accuracy at the cost of two more conversions per axes. The new expression for the vertical position then becomes:

(VY_P - VX_M) / (VX_P - VX_M)

26.7.3 Touch Screen Pin Connections

Table 26-2. Touch Screen Pin Connections

ADCIFB Pin	TS Signal, APOE == 0	TS Signal, APOE == 1
ADP0	Xp through a resistor	No Connect
ADP1	Yp through a resistor	No Connect
ADtspo+0	Xp	Xp
ADtspo+1	Xm	Xm
ADtspo+2	Yp	Yp
ADtspo+3	Ym	Ym

The touch screen film signals connects to the ADCIFB using the AD and ADP pins. The X_P (top) and X_M (bottom) film signals are connected to ADtspo+0 and ADtspo+1 pins, and the Y_P (right) and Y_M (left) signals are connected to ADtspo+2 and ADtspo+3 pins. The tspo index is configurable through the Touch Screen Pin Offset (TSPO) field in the Mode Register (MR) and allows the user to configure which AD pins to use for touch screen applications. Writing a zero to the TSPO field instructs the ADCIFB to use AD0 through AD3, where AD0 is connected to X_P, AD1 is connected to X_M and so on. Writing a one to the TSPO field instructs the ADCIFB to use AD1 through AD4 for touch screen sequencing, where AD1 is connected to X_P and AD0 is free to be used as an ordinary ADC channel.

When the Analog Pin Output Enable (APOE) bit in the Mode Register (MR) is zero, the AD pins are used to measure input voltage and drive the GND sequences, while the ADP pins are used to drive the VDD sequences. This arrangement allows the user to reduce the voltage seen at the AD input pins by inserting external resistors between ADP0 and X_P and ADP1 and Y_P signals which are again directly connected to the AD pins. It is important that the voltages observed at the AD pins are not higher than the maximum allowed ADC input voltage. See Figure 26-3 on page 607 for details regarding how to connect the touch screen films to the AD and ADP pins.

By adding a resistor between ADP0 and X_P, and ADP1 and Y_P, the maximum voltage observed at the AD pins can be controlled by the following voltage divider expressions:

$$V(AD_{tspo+0}) = \frac{R_{filmx}}{R_{filmx} + R_{resistorx}} \cdot V(DP_0)$$

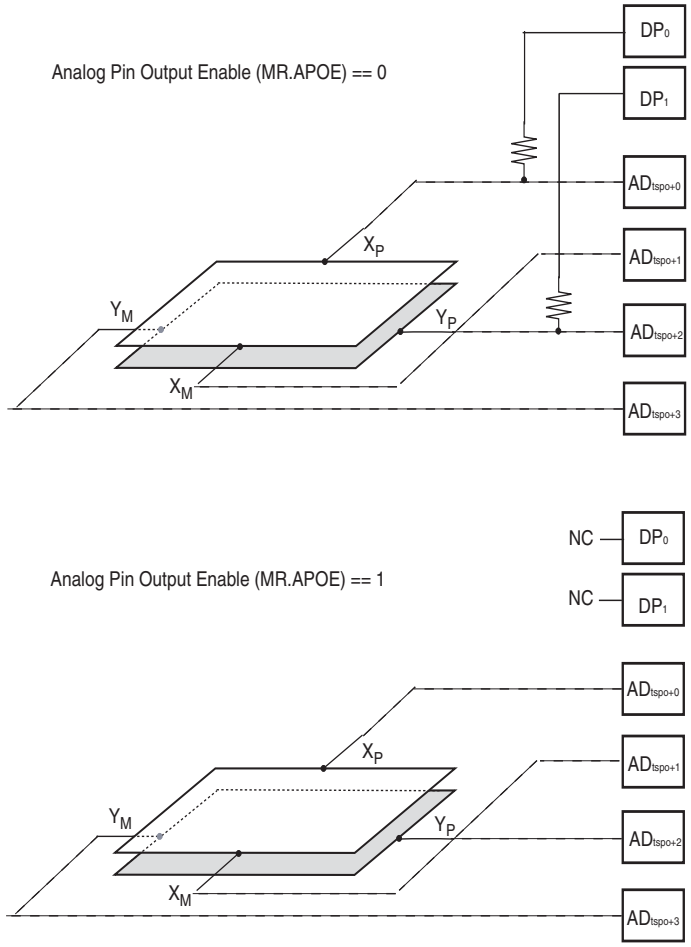
$$V(AD_{tspo+2}) = \frac{R_{filmY}}{R_{filmY} + R_{resistorY}} \cdot V(DP_1)$$

The Rfilmx parameter is the film resistance observed when measuring between X_P and X_M. The Rresistorx parameter is the resistor size inserted between ADP0 and X_P. The definition of Rfilmy and Rresistory is the same but for ADP1, Y_P, and Y_M instead.

The ADP pins are used by default, as the APOE bit is zero after reset. Writing a one to the APOE bit instructs the ADCIFB touch screen sequencer to use the already connected ADtspo+0 and ADtspo+2 pins to drive VDD to X_P and Y_P signals directly. In this mode the ADP pins can be used as general purpose I/O pins.

Before writing a one to the APOE bit the user must make sure that the I/O voltage is compatible with the ADC input voltage. If the I/O voltage is higher than the maximum input voltage of the ADC, permanent damage may occur. Refer to the Electrical Characteristics chapter for details.

Figure 26-3. Touch Screen Pin Connections



26.7.4 Touch Screen Sequencer

The touch screen sequencer is responsible for applying voltage to the resistive touch screen films as described in Section 26.7.2. This is done by controlling the output enable and the output value of the ADP and AD pins. This allows the touch screen sequencer to add a voltage gradient on one film while keeping the other film floating so a touch can be measured.

The Touch Screen Sequencer will when measuring the vertical position, apply VDD and GND to the pins connected to X_P and X_M. The Y_P and Y_M pins are put in tristate mode so the measurement of Y_P can proceed without interference. To compensate for ADC offset errors and non ideal pad drivers, the actual voltage of X_P and X_M is measured as well, so the real values for VDD and GND can be used in the contact point calculation to increase accuracy. See second formula in Section 26.7.2.

When the vertical values are converted the same setup is applies for the second axes, by setting X_P and X_M in tristate mode and applying VDD and GND to Y_P and Y_M. Refer to Section 26.8.3 for details.

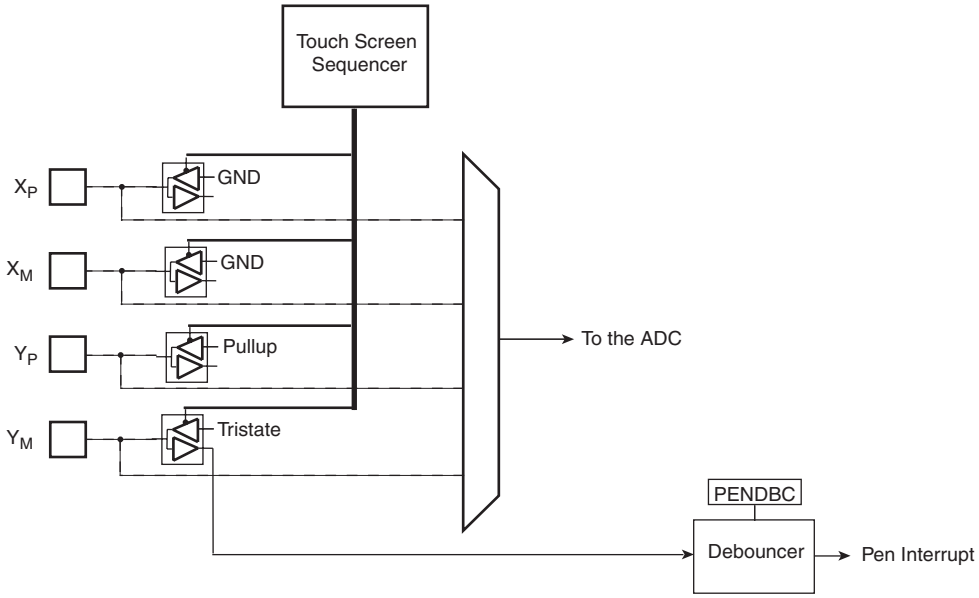
26.7.5 Pen Detect

If no contact is applied to the touch screen films, any touch screen conversion result will be undefined as the film being measured is floating. This can be avoided by enabling Pen Detect and only trigger touch screen conversions when the Pen Contact (PENCNT) status bit in the Status Register (SR) is one. Pen Detect is enabled by writing a one to the Pen Detect (PENDET) bit in the Mode Register (MR).

When Pen Detect is enabled, the ADCIFB grounds the vertical panel by applying GND to X_P and X_M and polarizes the horizontal panel by enabling pull-up on the pin connected to Y_P. The Y_M pin will in this mode be tristated. Since there is no contact, no current is flowing and there is no related power consumption. As soon as a contact occurs, GND will propagate to Y_M by pulling down Y_P, allowing the contact to be registered by the ADCIFB.

A programmable debouncing filter can be used to filter out false pen detects because of noise. The debouncing filter is programmable from one CLK_ADC period and up to 2^{15} CLK_ADC periods. The debouncer length is set by writing to the PENDBC field in MR.

Figure 26-4. Touch Screen Pen Detect



The Touch Screen Pen Detect can be used to generate an ADCIFB interrupt request or it can be used to trig a conversion, so that a position can be measured as soon as a contact is detected.

The Pen Detect Mode generates two types of status signals, reported in the Status Register (SR):

- The bit PENCNT is set when current flows and remains set until current stops.
- The bit NOCNT is set when no current flows and remains set until current flows.

Before a current change is reflected in the SR, the new status must be stable for the duration of the debouncing time.

Both status conditions can generate an interrupt request if the corresponding bit in the Interrupt Mask Register (IMR) is one. Refer to Section 26.6.11 on page 602.

26.8 Operating Modes

The ADCIFB features two operating modes, each defining a separate conversion sequence:

- ADC Mode: At each trigger, all the enabled channels are converted.
- Touch Screen ADC Mode: At each trigger, all enabled channels plus the touch screen channels are converted as described in Section 26.8.3. If channels except the dedicated touch screen channels are enabled, they are converted normally before and after the touch screen channels are converted.

The operating mode is selected by the TSAMOD field in the Mode Register (MR).

26.8.1 Conversion Triggers

A conversion sequence is started either by a software or by a hardware trigger. When a conversion sequence is started, all enabled channels will be converted and made available in the shared Last Converted Register (LCDR).

The software trigger is asserted by writing a one to the START field in the Control Register (CR).

The hardware trigger can be selected by the TRGMOD field in the Trigger Register (TRGR). Different hardware triggers exist:

- External trigger, either rising or falling or any, detected on the external trigger pin TRIGGER
- Pen detect trigger, depending the PENDET bit in the Mode Register (MR)
- Continuous trigger, meaning the ADCIFB restarts the next sequence as soon as it finishes the current one
- Periodic trigger, which is defined by the TRGR.TRGPER field
- Peripheral event trigger, allowing the Peripheral Event System to synchronize conversion with some configured peripheral event source.

Enabling a hardware trigger does not disable the software trigger functionality. Thus, if a hardware trigger is selected, the start of a conversion can still be initiated by the software trigger.

26.8.2 ADC Mode

In the ADC Mode, the active channels are defined by the Channel Status Register (CHSR). A channel is enabled by writing a one to the corresponding bit in the Channel Enable Register (CHER), and disabled by writing a one to the corresponding bit in the Channel Disable Register (CHDR). The conversion results are stored in the Last Converted Data Register (LCDR) as they become available, overwriting old conversions.

At each trigger, the following sequence is performed:

1. If ACR.SLEEP is one, wake up the ADC and wait for the startup time.
2. If Channel 0 is enabled, convert Channel 0 and store result in LCDR.
3. If Channel 1 is enabled, convert Channel 1 and store result in LCDR.
4. If Channel N is enabled, convert Channel N and store result in LCDR.
5. If ACR.SLEEP is one, place the ADC cell in a low-power state.

If the Peripheral DMA Controller is enabled, all converted values are transferred continuously into the memory buffer.

26.8.3 Touch Screen ADC Mode

Writing a one to the TSAMOD field in the Mode Register (MR) enables Touch Screen ADC Mode. In this mode the channels TSPO+0 to TSPO+3, corresponding to the touch screen

inputs, are automatically activated. In addition, if any other channels are enabled, they will be converted before and after the touch screen conversion.

At each trigger, the following sequence is performed:

1. If ACR.SLEEP is one, wake up the ADC cell and wait for the startup time.
2. Convert all enabled channels before TSPO and store the results in the LCDR.
3. Apply supply on the inputs X_P and X_M during the Sample and Hold Time.
4. Convert Channel X_M and store the result in TMP.
5. Apply supply on the inputs X_P and X_M during the Sample and Hold Time.
6. Convert Channel X_P, subtract TMP from the result and store the subtracted result in LCDR.
7. Apply supply on the inputs X_P and X_M during the Sample and Hold Time.
8. Convert Channel Y_P, subtract TMP from the result and store the subtracted result in LCDR.
9. Apply supply on the inputs Y_P and Y_M during the Sample and Hold Time.
10. Convert Channel Y_M and store the result in TMP.
11. Apply supply on the inputs Y_P and Y_M during the Sample and Hold Time.
12. Convert Channel Y_P, subtract TMP from the result and store the subtracted result in LCDR.
13. Apply supply on the inputs Y_P and Y_M during the Sample and Hold Time.
14. Convert Channel X_P, subtract TMP from the result and store the subtracted result in LCDR.
15. Convert all enabled channels after TSPO + 3 and store results in the LCDR.
16. If ACR.SLEEP is one, place the ADC cell in a low-power state.

The resulting buffer structure stored in memory is:

1. X_P - X_M
2. Y_P - X_M
3. Y_P - Y_M
4. X_P - Y_M.

The vertical position can be easily calculated by dividing the data at offset 1(X_P - X_M) by the data at offset 2(Y_P - X_M).

The horizontal position can be easily calculated by dividing the data at offset 3(Y_P - Y_M) by the data at offset 4(X_P - Y_M).

26.9 User Interface

Table 26-3. ADCIFB Register Memory Map

Offset	Register	Name	Access	Reset
0x00	Control Register	CR	Write-only	-
0x04	Mode Register	MR	Read/Write	0x00000000
0x08	ADC Configuration Register	ACR	Read/Write	0x00000000
0x0C	Trigger Register	TRGR	Read/Write	0x00000000
0x10	Compare Value Register	CVR	Read/Write	0x00000000
0x14	Status Register	SR	Read-only	0x00000000
0x18	Interrupt Status Register	ISR	Read-only	0x00000000
0x1C	Interrupt Clear Register	ICR	Write-only	-
0x20	Interrupt Enable Register	IER	Write-only	-
0x24	Interrupt Disable Register	IDR	Write-only	-
0x28	Interrupt Mask Register	IMR	Read-only	0x00000000
0x2C	Last Converted Data Register	LCDR	Read-only	0x00000000
0x30	Parameter Register	PARAMETER	Read-only	-(1)
0x34	Version Register	VERSION	Read-only	-(1)
0x40	Channel Enable Register	CHER	Write-only	-
0x44	Channel Disable Register	CHDR	Write-only	-
0x48	Channel Status Register	CHSR	Read-only	0x00000000

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

26.9.1 Control Register

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
–	–	–	–	–	–	DIS	EN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	START	SWRST

- **DIS: ADCDIFB Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the ADCIFB.
  Note: Disabling the ADCIFB effectively stops all clocks in the module so the user must make sure the ADCIFB is idle before disabling the ADCIFB.
- **EN: ADCIFB Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the ADCIFB.
  Note: The ADCIFB must be enabled before use.
- **START: Start Conversion**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts an Analog-to-Digital conversion.
- **SWRST: Software Reset**
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets the ADCIFB, simulating a hardware reset.

26.9.2 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
PENDBC				–	–	–	–

23	22	21	20	19	18	17	16
TSPO							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	APOE	ACE	PENDET	–	–	_	TSAMOD

- **PENDBC: Pen Detect Debouncing Period**

  Period = $2^{PENDBC} * T_{CLK_ADC}$
- **TSPO: Touch Screen Pin Offset**

  The Touch Screen Pin Offset field is used to indicate which AD pins are connected to the resistive touch screen film edges. Only an offset is specified and it is assumed that the touch screen films are connected sequentially from the specified offset pin and up to and including offset + 3 (4 pins).
- **APOE: Analog Pin Output Enable**

  0: AD pins are not used to drive VDD in Touch Screen sequence.

  1: AD pins are used to drive VDD in Touch Screen sequence.

  Note: If the selected I/O voltage configuration is incompatible with the Analog-to-Digital converter cell voltage specification, this bit must stay cleared to avoid damaging the ADC. In this case the ADP pins must be used to drive VDD instead, as described in Section 26.7.3. If the I/O and ADC voltages are compatible, the AD pins can be used directly by writing a one to this bit. In this case the ADP pins can be ignored.
- **ACE: Analog Compare Enable**

  0: The analog compare functionality is disabled.

  1: The analog compare functionality is enabled.
- **PENDET: Pen Detect**

  0: The pen detect functionality is disabled.

  1: The pen detect functionality is enabled.
- **TSAMOD: Touch Screen ADC Mode**

  0: Touch Screen Mode is disabled.

  1: Touch Screen Mode is enabled.

26.9.3 ADC Configuration Register

Name: ACR

Access Type: Read/Write

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	SHTIM			

23	22	21	20	19	18	17	16
–	STARTUP						

15	14	13	12	11	10	9	8
–	–	PRESCAL					

7	6	5	4	3	2	1	0
-	–	RES		–	–	–	SLEEP

- **SHTIM: Sample & Hold Time for ADC Channels**

  $T_{SAMPLE\&HOLD} = (SHTIM + 1) \cdot T_{CLK_ADC}$
- **STARTUP: Startup Time**

  $_{TARTUP} = (STARTUP + 1) \cdot 8 \cdot T_{CLK_A}$
- **PRESCAL: Prescaler Rate Selection**

  $T_{CLK_ADC} = (PRESCAL + 1) \cdot 2 \cdot T_{CLK_ADCIFB}$
- **RES: Resolution Selection**

  0: 8-bit resolution.

  1: 10-bit resolution.

  2: 11-bit resolution, interpolated.

  3: 12-bit resolution, interpolated.
- **SLEEP: ADC Sleep Mode**

  0: ADC Sleep Mode is disabled.

  1: ADC Sleep Mode is enabled.

26.9.4 Trigger Register

Name: TRGR

Access Type: Read/Write

Offset: 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
TRGPER[15:8]							

23	22	21	20	19	18	17	16
TRGPER[7:0]							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	TRGMOD		

- **TRGPER: Trigger Period**

  Effective only if TRGMOD defines a Periodic Trigger.

  Defines the periodic trigger period, with the following equations:

  Trigger Period = TRGPER *T_{CLK_ADC}

- **TRGMOD: Trigger Mode**

Table 26-4. Trigger Modes

TRGMOD			Selected Trigger Mode
0	0	0	No trigger, only software trigger can start conversions
0	0	1	External Trigger Rising Edge
0	1	0	External Trigger Falling Edge
0	1	1	External Trigger Any Edge
1	0	0	Pen Detect Trigger (shall be selected only if PENDET is set and TSAMOD = Touch Screen mode)
1	0	1	Periodic Trigger (TRGPER shall be initiated appropriately)
1	1	0	Continuous Mode
1	1	1	Peripheral Event Trigger

26.9.5 Compare Value Register

Name: CVR

Access Type: Read/Write

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	HV[11:8]			

23	22	21	20	19	18	17	16
HV[7:0]							

15	14	13	12	11	10	9	8
–	–	–	–	LV[11:8]			

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

- **HV: High Value**
  Defines the high value used when comparing analog input.
- **LV: Low Value**
  Defines the low value used when comparing analog input.

26.9.6 Status Register

Name: SR

Access Type: Read-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	EN

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

15	14	13	12	11	10	9	8
	CELSE	CGT	CLT	–	–	BUSY	READY

7	6	5	4	3	2	1	0
–	–	NOCNT	PENCNT	–	–	OVRE	DRDY

- **EN: Enable Status**
  - 0: The ADCIFB is disabled.
  - 1: The ADCIFB is enabled.
- **CELSE: Compare Else Status**
  - 0: Either CLT or CGT detected or compare is disabled.
  - 1: No CLT or CGT detected on last converted data.
- **CGT: Compare Greater Than Status**
  - 0: No compare greater than CVR.HV detected on last converted data or compare is disabled.
  - 1: Compare greater than CVR.HV detected on last converted data or compare is disabled.
- **CLT: Compare Lesser Than Status**
  - 0: No compare lesser than CVR.LV detected on last converted data or compare is disabled.
  - 1: Compare lesser than CVR.LV detected on last converted data or compare is disabled.
- **BUSY: Busy Status**
  - 0: The ADCIFB is ready to perform a conversion sequence.
  - 1: The ADCIFB is busy performing a convention sequence.
- **READY: Ready Status**
  - 0: The ADCIFB is busy performing a conversion sequence.
  - 1: The ADCIFB is ready to perform a conversion sequence.
- **NOCNT: No Contact Status**
  - 0: No contact loss is detected or pen detect disabled.
  - 1: Contact loss is detected.
- **PENCNT: Pen Contact Status**
  - 0: No contact is detected or pen detect disabled.
  - 1: Pen contact is detected.
- **OVRE: Overrun Error Status**
  - 0: No Overrun Error has occurred since the start of conversion sequence.
  - 1: One or more Overrun Error has occurred since the start of conversion sequence.
- **DRDY: Data Ready Status**
  - 0: No data has been converted since the start of conversion sequence.
  - 1: One or more data has been converted since the start of conversion and is available in LCDR.

26.9.7 Interrupt Status Register

Name: ISR

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
–	CELSE	CGT	CLT	–	–	BUSY	READY

7	6	5	4	3	2	1	0
–	–	NOCNT	PENCNT	–	–	OVRE	DRDY

A bit in this register is cleared by writing a one to the corresponding bit in the Interrupt Clear Register (ISR).
A bit in this register is set when the corresponding bit in the Status Register (SR) is set.

26.9.8 Interrupt Clear Register

Name: ICR

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
–	CELSE	CGT	CLT	–	–	BUSY	READY

7	6	5	4	3	2	1	0
–	–	NOCNT	PENCNT	–	–	OVRE	DRDY

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.

26.9.9 Interrupt Enable Register

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
–	CELSE	CGT	CLT	–	–	BUSY	READY

7	6	5	4	3	2	1	0
–	–	NOCNT	PENCNT	–	–	OVRE	DRDY

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.

26.9.10 Interrupt Disable Register

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
–	CELSE	CGT	CLT	–	–	BUSY	READY

7	6						
–	–	NOCNT	PENCNT	–	–	OVRE	DRDY

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.

26.9.11 Interrupt Mask Register

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
–	CELSE	CGT	CLT	–	–	BUSY	READY

7	6						
–	–	NOCNT	PENCNT	–	–	OVRE	DRDY

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared by writing a one to the corresponding bit in Interrupt Disable Register (IDR).

A bit in this register is set by writing a one to the corresponding bit in Interrupt Enable Register (IER).

26.9.12 Last Converted Data Register

Name: LCDR

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
LCCH							

15	14	13	12	11	10	9	8
–	–	–	–	LDATA[11:8]			

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

- **LCCH: Last Converted Channel**

  This field indicates what channel was last converted, i.e. what channel the LDATA represents.

- **LDATA: Last Data Converted**

  The analog-to-digital conversion data is placed in this register at the end of a conversion on any analog channel and remains until a new conversion on any analog channel is completed.

26.9.13 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x30

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHn: Channel n Implemented**

  0: The corresponding channel is not implemented.

  1: The corresponding channel is implemented.

26.9.14 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x34

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.

26.9.15 Channel Enable Register

Name: CHER

Access Type: Write-only

Offset: 0x40

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHn: Channel n Enable**

  Writing a zero to a bit in this register has no effect

  Writing a one to a bit in this register enables the corresponding channel

  The number of available channels is device dependent. Please refer to the Module Configuration section at the end of this chapter for information regarding which channels are implemented.

26.9.16 Channel Disable Register

Name: CHDR

Access Type: Write-only

Offset: 0x44

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHn: Channel N Disable**

  Writing a zero to a bit in this register has no effect.

  Writing a one to a bit in this register disables the corresponding channel.

  Warning: If the corresponding channel is disabled during a conversion, or if it is disabled and then re-enabled during a conversion, its associated data and its corresponding DRDY and OVRE bits in SR are unpredictable.

  The number of available channels is device dependent. Please refer to the Module Configuration section at the end of this chapter for information regarding how many channels are implemented.

26.9.17 Channel Status Register

Name: CHSR

Access Type: Read-only

Offset: 0x48

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHn: Channel N Status**

  0: The corresponding channel is disabled.

  1: The corresponding channel is enabled.

  A bit in this register is cleared by writing a one to the corresponding bit in Channel Disable Register (CHDR).

  A bit in this register is set by writing a one to the corresponding bit in Channel Enable Register (CHER).

  The number of available channels is device dependent. Please refer to the Module Configuration section at the end of this chapter for information regarding how many channels are implemented.

26.10 Module Configuration

The specific configuration for each ADCIFB 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 26-5. Module Configuration

Feature	ADCIFB
Number of ADC channels	9 (8 + 1 internal temperature sensor channel)

Table 26-6. Module Clock Name

Module Name	Clock Name
ADCIFB	**CLK_ADCIFB**

Table 26-7. Register Reset Values

Register	Reset Value
VERSION	0x00000101
PARAMETER	0x000003FF

Table 26-8. ADC Input Channels

Channel	Input
CH0	AD0
CH1	AD1
CH2	AD2
CH4	AD4
CH5	AD5
CH6	AD6
CH7	AD7
CH8	AD8
CH9	Temperature sensor

27. Analog Comparator Interface (ACIFB)

Rev: 2.0.2.2

27.1 Features

- **Controls an array of Analog Comparators**
- **Low power option**
  - **Single shot mode support**
- **Selectable settings for filter option**
  - **Filter length and hysteresis**
- **Window Mode**
  - **Detect inside/outside window**
  - **Detect above/below window**
- **Interrupt**
  - **On comparator result rising edge, falling edge, toggle**
  - **Inside window, outside window, toggle**
  - **When startup time has passed**
- **Can generate events to the peripheral event system**

27.2 Overview

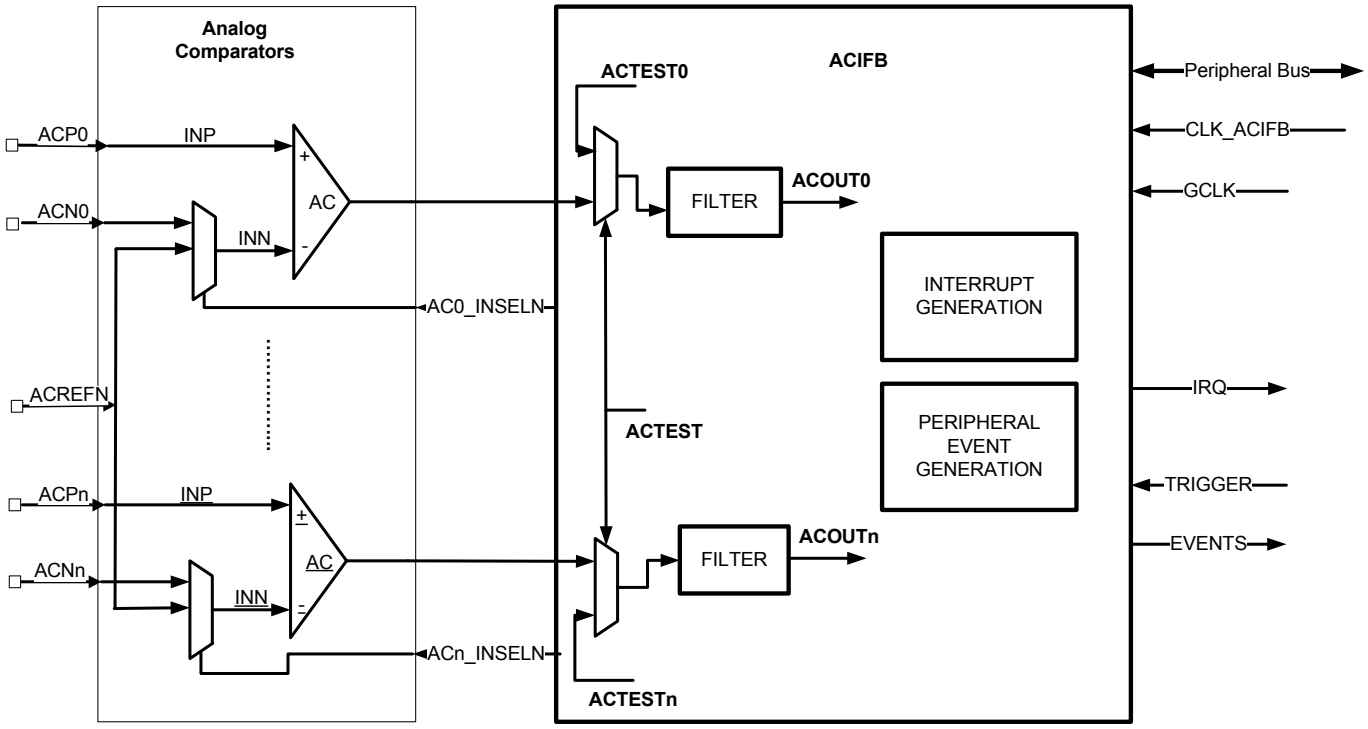
The Analog Comparator Interface (ACIFB) is able to control a number of Analog Comparators (AC) with identical behavior. An Analog Comparator compares two voltages and gives a compare output depending on this comparison.

The ACIFB can be configured in normal mode using each comparator independently or in window mode using defined comparator pairs to observe a window.

The number of channels implemented is device specific. Refer to the Module Configuration section at the end of this chapter for details.

27.3 Block Diagram

Figure 27-1. ACIFB Block Diagram



27.4 I/O Lines Description

Table 27-1. I/O Line Description

Pin Name	Pin Description	Type
ACPn	Positive reference pin for Analog Comparator n	Analog
ACNn	Negative reference pin for Analog Comparator n	Analog
ACREFN	Reference Voltage for all comparators selectable for INN	Analog

27.5 Product Dependencies

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

27.5.1 I/O Lines

The ACIFB pins are multiplexed with other peripherals. The user must first program the I/O Controller to give control of the pins to the ACIFB.

27.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the ACIFB, the ACIFB will stop functioning and resume operation after the system wakes up from sleep mode.

27.5.3 Clocks

The clock for the ACIFB bus interface (CLK_ACIFB) 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 ACIFB before disabling the clock, to avoid freezing the ACIFB in an undefined state.

The ACIFB uses a GCLK as clock source for the Analog Comparators. The user must set up this GCLK at the right frequency. The CLK_ACIFB clock of the interface must be at least 4x the GCLK frequency used in the comparators. The GCLK is used both for measuring the startup time of a comparator, and to give a frequency for the comparations done in Continuous Measurement Mode, see Section 27.6.

Refer to the Electrical Characteristics chapter for GCLK frequency limitations.

27.5.4 Interrupts

The ACIFB interrupt request line is connected to the interrupt controller. Using the ACIFB interrupt requires the interrupt controller to be programmed first.

27.5.5 Peripheral Events

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

27.5.6 Debug Operation

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

27.6 Functional Description

The ACIFB is enabled by writing a one to the Control Register Enable bit (CTRL.EN). Additionally, the comparators must be individually enabled by programming the MODE field in the AC Configuration Register (CONFn.MODE).

The results from the individual comparators can either be used directly (normal mode), or the results from two comparators can be grouped to generate a comparison window (window mode). All comparators need not be in the same mode, some comparators may be in normal mode, while others are in window mode. There are restrictions on which AC channels that can be grouped together in a window pair, see Section 27.6.5.

27.6.1 Analog Comparator Operation

Each AC channel can be in one of four different modes, determined by CONFn.MODE:

- Off
- Continuous Measurement Mode (CM)
- User Triggered Single Measurement Mode (UT)
- Event Triggered Single Measurement Mode (ET)

After being enabled, a startup time defined in CTRL.SUT is required before the result of the comparison is ready. The GCLK is used for measuring the startup time of a comparator,

During the startup time the AC output is not available. When the ACn Ready bit in the Status Register (SR.ACRDYn) is one, the output of ACn is ready. In window mode the result is available when both the comparator outputs are ready (SR.ACRDYn=1 and SR.ACRDYn+1=1).

27.6.1.1 *Continuous Measurement Mode*

In CM, the Analog Comparator is continuously enabled and performing comparisons. This ensures that the result of the latest comparison is always available in the ACn Current Comparison Status bit in the Status Register (SR.ACCSn). Comparisons are done on every positive edge of GCLK.

CM is enabled by writing CONFn.MODE to 1. After the startup time has passed, a comparison is done and SR is updated. Appropriate peripheral events and interrupts are also generated. New comparisons are performed continuously until the CONFn.MODE field is written to 0.

27.6.1.2 *User Triggered Single Measurement Mode*

In the UT mode, the user starts a single comparison by writing a one to the User Start Single Comparison bit (CTRL.USTART). This mode is enabled by writing CONFn.MODE to 2. After the startup time has passed, a single comparison is done and SR is updated. Appropriate peripheral events and interrupts are also generated. No new comparisons will be performed. CTRL.USTART is cleared automatically by hardware when the single comparison has been done.

27.6.1.3 *Event Triggered Single Measurement Mode*

This mode is enabled by writing CONFn.MODE to 3 and Peripheral Event Trigger Enable (CTRL.EVENTEN) to one. The ET mode is similar to the UT mode, the difference is that a peripheral event from another hardware module causes the hardware to automatically set the Peripheral Event Start Single Comparison bit (CTRL.ESTART). After the startup time has passed, a single comparison is done and SR is updated. Appropriate peripheral events and interrupts are also generated. No new comparisons will be performed. CTRL.ESTART is cleared automatically by hardware when the single comparison has been done.

27.6.1.4 *Selecting Comparator Inputs*

Each Analog Comparator has one positive (INP) and one negative (INN) input. The positive input is fed from an external input pin (ACPn). The negative input can either be fed from an external input pin (ACNn) or from a reference voltage common to all ACs (ACREFN).

The user selects the input source as follows:

- In normal mode with the Negative Input Select and Positive Input Select fields (CONFn.INSELN and CONFn.INSELP).
- In window mode with CONFn.INSELN, CONFn.INSELP and CONFn+1.INSELN, CONFn+1,INSELP. The user must configure CONFn.INSELN and CONFn+1.INSELP to the same source.

27.6.2 Interrupt Generation

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 ISR is cleared by writing a one to the corresponding bit in the Interrupt Status Clear Register (ICR).

27.6.3 Peripheral Event Generation

The ACIFB can be set up so that certain comparison results notify other parts of the device via the Peripheral Event system. Refer to Section 27.6.4.3 and Section 27.6.5.3 for information on which comparison results can generate events, and how to configure the ACIFB to achieve this.

Zero or one event will be generated per comparison.

27.6.4 Normal Mode

In normal mode all Analog Comparators are operating independently.

27.6.4.1 Normal Mode Output

Each Analog Comparator generates one output ACOUT according to the input voltages on INP (AC positive input) and INN (AC negative input):

- ACOUT = 1 if $V_{INP} > V_{INN}$
- ACOUT = 0 if $V_{INP} < V_{INN}$
- ACOUT = 0 if the AC output is not available (SR.ACRDY = 0)

The output can optionally be filtered, as described in Section 27.6.6.

27.6.4.2 Normal Mode Interrupt

The AC channels can generate interrupts. The Interrupt Settings field in the Configuration Register (CONFn.IS) can be configured to select when the AC will generate an interrupt:

- When $V_{INP} > V_{INN}$
- When $V_{INP} < V_{INN}$
- On toggle of the AC output (ACOUT)
- When comparison has been done

27.6.4.3 Normal Mode Peripheral Events

The ACIFB can generate peripheral events according to the configuration of CONFn.EVENN and CONFn.EVENP.

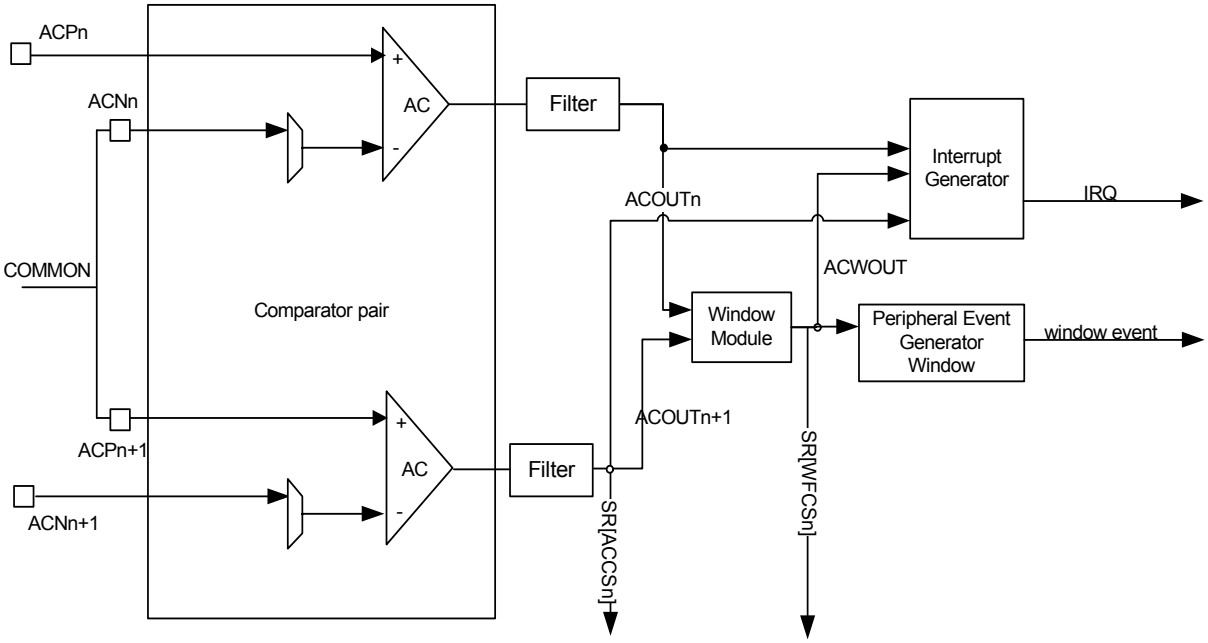
- When $V_{INP} > V_{INN}$ or
- When $V_{INP} < V_{INN}$ or
- On toggle of the AC output (ACOUT)

27.6.5 Window Mode

In window mode, two ACs (an even and the following odd build up a pair) are grouped.

The negative input of ACn (even) and the positive input of ACn+1 (odd) has to be connected together externally to the device and are controlled by the Input Select fields in the AC Configuration Registers (CONFn.INSELN and CONFn+1.INSELP). The positive input of ACn (even) and the negative input of ACn+1 (odd) can still be configured independently by CONFn.INSELP and CONFn+1.INSELN, respectively.

Figure 27-2. Analog Comparator Interface in Window Mode



27.6.5.1 *Window Mode Output*

When operating in window mode, each channel generates the same ACOUT outputs as in normal mode, see Section 27.6.4.1.

Additionally, the ACIFB generates a window mode signal (acwout) according to the common input voltage to be compared:

- ACWOUT = 1 if the common input voltage is inside the window, $V_{ACN(N+1)} < V_{common} < V_{ACP(N)}$
- ACWOUT = 0 if the common input voltage is outside the window, $V_{common} < V_{ACN(N+1)}$ or $V_{common} > V_{ACP(N)}$
- ACWOUT = 0 if the window mode output is not available (SR.ACRDY*n*=0 or SR.ACRDY*n+1=0)*

27.6.5.2 *Window Mode Interrupts*

When operating in window mode, each channel can generate the same interrupts as in normal mode, see Section 27.6.4.2.

Additionally, when channels operate in window mode, programming Window Mode Interrupt Settings in the Window Mode Configuration Register (CONFWn.WIS) can cause interrupts to be generated when:

- 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).
- When the comparison in both channels in the window pair is ready.

27.6.5.3 Window Mode Peripheral Events

When operating in window mode, each channel can generate the same peripheral events as in normal mode, see Section 27.6.4.3.

Additionally, when channels operate in window mode, programming Window Mode Event Selection Source (CONFWn.WEVSRC) can cause peripheral events to be generated when:

- 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)
- Whenever a comparison is ready and the common input voltage is inside the window.
- Whenever a comparison is ready and the common input voltage is outside the window.
- When the comparison in both channels in the window pair is ready.

27.6.6 Filtering

The output of the comparator can be filtered to reduce noise. The filter length is determined by the Filter Length field in the CONFn register (CONFn.FLEN). The filter samples the Analog Comparator output at the GCLK frequency for $2^{CONFn.FLEN}$ samples. A separate counter (CNT) counts the number of cycles the AC output was one. This filter is deactivated if CONFn.FLEN equals 0.

If the filter is enabled, the Hysteresis Value field HYS in the CONFn register (CONFn.HYS) can be used to define a hysteresis value. The hysteresis value should be chosen so that:

$$\frac{2^{FLEN}}{2} \geq HYS$$

The filter function is defined by:

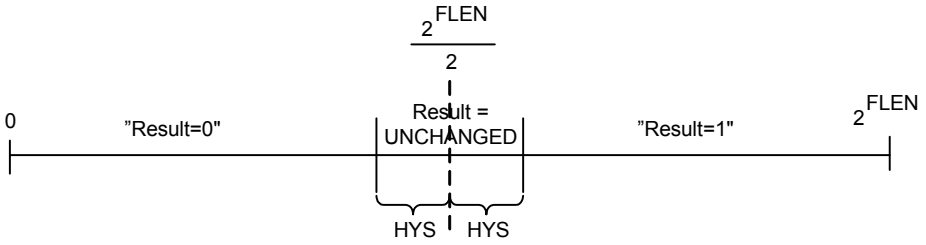
$$CNT \geq \left(\frac{2^{FLEN}}{2} + HYS\right) \Rightarrow comp = 1$$

$$\left(\frac{2^{FLEN}}{2} + HYS\right) > CNT \geq \left(\frac{2^{FLEN}}{2} - HYS\right) \Rightarrow comp \text{ unchanged}$$

$$CNT < \left(\frac{2^{FLEN}}{2} - HYS\right) \Rightarrow comp = 0$$

The filtering algorithm is explained in Figure 27-3. 2^{FLEN} measurements are sampled. If lthe number of measurements that are zero is less than (2^{FLEN}/2 - HYS), the filtered result is zero. If lthe number of measurements that are one is more than (2^{FLEN}/2 + HYS), the filtered result is one. Otherwise, the result is unchanged.

Figure 27-3. The Filtering Algorithm



27.7 Peripheral Event Triggers

Peripheral events from other modules can trigger comparisons in the ACIFB. All channels that are set up in Event Triggered Single Measurement Mode will be started simultaneously when a peripheral event is received. Channels that are operating in Continuous Measurement Mode or User Triggered Single Measurement Mode will be unaffected by the received event. The software can still operate these channels independently of channels in Event Triggered Single Measurement Mode.

A peripheral event will trigger one or more comparisons, in normal or window mode.

27.8 AC Test mode

By writing the Analog Comparator Test Mode (CR.ACTEST) bit to one, the outputs from the ACs are overridden by the value in the Test Register (TR), see Figure 27-1. This is useful for software development.

27.9 User Interface

Table 27-2. ACIFB Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CTRL	Read/Write	0x00000000
0x04	Status Register	SR	Read-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	Interrupt Status Register	ISR	Read-only	0x00000000
0x20	Interrupt Status Clear Register	ICR	Write-only	0x00000000
0x24	Test Register	TR	Read/Write	0x00000000
0x30	Parameter Register	PARAMETER	Read-only	-(1)
0x34	Version Register	VERSION	Read-only	-(1)
0x80	Window0 Configuration Register	CONFW0	Read/Write	0x00000000
0x84	Window1 Configuration Register	CONFW1	Read/Write	0x00000000
0x88	Window2 Configuration Register	CONFW2	Read/Write	0x00000000
0x8C	Window3 Configuration Register	CONFW3	Read/Write	0x00000000
0xD0	AC0 Configuration Register	CONF0	Read/Write	0x00000000
0xD4	AC1 Configuration Register	CONF1	Read/Write	0x00000000
0xD8	AC2 Configuration Register	CONF2	Read/Write	0x00000000
0xDC	AC3 Configuration Register	CONF3	Read/Write	0x00000000
0xE0	AC4 Configuration Register	CONF4	Read/Write	0x00000000
0xE4	AC5 Configuration Register	CONF5	Read/Write	0x00000000
0xE8	AC6 Configuration Register	CONF6	Read/Write	0x00000000
0xEC	AC7 Configuration Register	CONF7	Read/Write	0x00000000

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

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
-	-	-	-	-	-	SUT[9:8]	

15	14	13	12	11	10	9	8
SUT[7:0]							

7	6	5	4	3	2	1	0
ACTEST	-	ESTART	USTART	-	-	EVENTEN	EN

- **SUT: Startup Time**
  Analog Comparator startup time = $\frac{SUT}{F_{GCLK}}$.

  Each time an AC is enabled, the AC comparison will be enabled after the startup time of the AC.
- **ACTEST: Analog Comparator Test Mode**
  0: The Analog Comparator outputs feeds the channel logic in ACIFB.
  1: The Analog Comparator outputs are bypassed with the AC Test Register.
- **ESTART: Peripheral Event Start Single Comparison**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts a comparison and can be used for test purposes.
  This bit is cleared when comparison is done.
  This bit is set when an enabled peripheral event is received.
- **USTART: User Start Single Comparison**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts a Single Measurement Mode comparison.
  This bit is cleared when comparison is done.
- **EVENTEN: Peripheral Event Trigger Enable**
  0: A peripheral event will not trigger a comparison.
  1: Enable comparison triggered by a peripheral event.
- **EN: ACIFB Enable**
  0: The ACIFB is disabled.
  1: The ACIFB is enabled.

27.9.2 Status Register

Name: SR

Access Type: Read-only

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	WFCS3	WFCS2	WFCS1	WFCS0
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
ACRDY7	ACCS7	ACRDY6	ACCS6	ACRDY5	ACCS5	ACRDY4	ACCS4
7	6	5	4	3	2	1	0
ACRDY3	ACCS3	ACRDY2	ACCS2	ACRDY1	ACCS1	ACRDY0	ACCS0

- **WFCSn: Window Mode Current Status**
  This bit is cleared when the common input voltage is outside the window.
  This bit is set when the common input voltage is inside the window.
- **ACRDYn: ACn Ready**
  This bit is cleared when the AC output (ACOUT) is not ready.
  This bit is set when the AC output (ACOUT) is ready, AC is enabled and its startup time is over.
- **ACCSn: ACn Current Comparison Status**
  This bit is cleared when V_{INP} is currently lower than V_{INN}
  This bit is set when V_{INP} is currently greater than V_{INN}.

27.9.3 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	WFINT3	WFINT2	WFINT1	WFINT0
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
SUTINT7	ACINT7	SUTINT6	ACINT6	SUTINT5	ACINT5	SUTINT4	ACINT4
7	6	5	4	3	2	1	0
SUTINT3	ACINT3	SUTINT2	ACINT2	SUTINT1	ACINT1	SUTINT0	ACINT0

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.

27.9.4 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	WFINT3	WFINT2	WFINT1	WFINT0
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
SUTINT7	ACINT7	SUTINT6	ACINT6	SUTINT5	ACINT5	SUTINT4	ACINT4
7	6	5	4	3	2	1	0
SUTINT3	ACINT3	SUTINT2	ACINT2	SUTINT1	ACINT1	SUTINT0	ACINT0

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.

27.9.5 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	WFINT3	WFINT2	WFINT1	WFINT0
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
SUTINT7	ACINT7	SUTINT6	ACINT6	SUTINT5	ACINT5	SUTINT4	ACINT4
7	6	5	4	3	2	1	0
SUTINT3	ACINT3	SUTINT2	ACINT2	SUTINT1	ACINT1	SUTINT0	ACINT0

- **WFINT*n*: Window Mode Interrupt Mask**
  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.
- **SUTINT*n*: AC*n* Startup Time Interrupt Mask**
  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.
- **ACINT*n*: AC*n* Interrupt Mask**
  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.

27.9.6 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	WFINT3	WFINT2	WFINT1	WFINT0
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
SUTINT7	ACINT7	SUTINT6	ACINT6	SUTINT5	ACINT5	SUTINT4	ACINT4
7	6	5	4	3	2	1	0
SUTINT3	ACINT3	SUTINT2	ACINT2	SUTINT1	ACINT1	SUTINT0	ACINT0

- **WFINTn: Window Mode Interrupt Status**
  - 0: No Window Mode Interrupt is pending.
  - 1: Window Mode Interrupt is pending.
  - This bit is cleared when the corresponding bit in ICR is written to one.
  - This bit is set when the corresponding channel pair operating in window mode generated an interrupt.
- **SUTINTn: ACn Startup Time Interrupt Status**
  - 0: No Startup Time Interrupt is pending.
  - 1: Startup Time Interrupt is pending.
  - This bit is cleared when the corresponding bit in ICR is written to one.
  - This bit is set when the startup time of the corresponding AC has passed.
- **ACINTn: ACn Interrupt Status**
  - 0: No Normal Mode Interrupt is pending.
  - 1: Normal Mode Interrupt is pending.
  - This bit is cleared when the corresponding bit in ICR is written to one.
  - This bit is set when the corresponding channel generated an interrupt.

27.9.7 Interrupt Status Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x20

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	WFINT3	WFINT2	WFINT1	WFINT0
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
SUTINT7	ACINT7	SUTINT6	ACINT6	SUTINT5	ACINT5	SUTINT4	ACINT4
7	6	5	4	3	2	1	0
SUTINT3	ACINT3	SUTINT2	ACINT2	SUTINT1	ACINT1	SUTINT0	ACINT0

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

27.9.8 Test Register

Name: TR

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
ACTEST7	ACTEST6	ACTEST5	ACTEST4	ACTEST3	ACTEST2	ACTEST1	ACTEST0

- **ACTESTn: AC Output Override Value**
  If CTRL.ACTEST is set, the ACn output is overridden with the value of ACTESTn.

27.9.9 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
-	-	-	-	WIMPL3	WIMPL2	WIMPL1	WIMPL0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
ACIMPL7	ACIMPL6	ACIMPL5	ACIMPL4	ACIMPL3	ACIMPL2	ACIMPL1	ACIMPL0

- **WIMPLn: Window Pair n Implemented**
  0: Window Pair not implemented.
  1: Window Pair implemented.
- **ACIMPLn: Analog Comparator n Implemented**
  0: Analog Comparator not implemented.
  1: Analog Comparator implemented.

27.9.10 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.

27.9.11 Window Configuration Register

Name: CONFWn

Access Type: Read/Write

Offset: 0x80,0x84,0x88,0x8C

Reset Value: 0x00000000

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

- **WFEN: Window Mode Enable**
  - 0: The window mode is disabled.
  - 1: The window mode is enabled.
- **WEVEN: Window Event Enable**
  - 0: Event from awout is disabled.
  - 1: Event from awout is enabled.
- **WEVSRC: Event Source Selection for Window Mode**
  - 000: Event on acwout rising edge.
  - 001: Event on acwout falling edge.
  - 010: Event on awout rising or falling edge.
  - 011: Inside window.
  - 100: Outside window.
  - 101: Measure done.
  - 110-111: Reserved.
- **WIS: Window Mode Interrupt Settings**
  - 00: Window interrupt as soon as the input voltage is inside the window.
  - 01: Window interrupt as soon as the input voltage is outside the window.
  - 10: Window interrupt on toggle of window compare output.
  - 11: Window interrupt when evaluation of input voltage is done.

27.9.12 AC Configuration Register

Name: CONFn

Access Type: Read/Write

Offset: 0xD0,0xD4,0xD8,0xDC,0xE0,0xE4,0xE8,0xEC

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	FLEN			HYS			

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

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

7	6	5	4	3	2	1	0
-	-	MODE		-	-	IS	

- **FLEN: Filter Length**
  000: Filter off.
  n: Number of samples to be averaged =2^n.
- **HYS: Hysteresis Value**
  0000: No hysteresis.
  1111: Max hysteresis.
- **EVENN: Event Enable Negative**
  0: Do not output event when ACOUT is zero.
  1: Output event when ACOUT is zero.
- **EVENP: Event Enable Positive**
  0: Do not output event when ACOUT is one.
  1: Output event when ACOUT is one.
- **INSELP: Positive Input Select**
  00: ACPn pin selected.
  01: Reserved.
  10: Reserved.
  11: Reserved.
- **INSELN: Negative Input Select**
  00: ACNn pin selected.
  01: ACREFN pin selected.
  10: Reserved.
  11: Reserved.
- **MODE: Mode**
  00: Off.
  01: Continuous Measurement Mode.
  10: User Triggered Single Measurement Mode.
  11: Event Triggered Single Measurement Mode.

- **IS: Interrupt Settings**
  - 00: Comparator interrupt when as $V_{INP} > V_{INN}$.
  - 01: Comparator interrupt when as $V_{INP} < V_{INN}$.
  - 10: Comparator interrupt on toggle of Analog Comparator output.
  - 11: Comparator interrupt when comparison of V_{INP} and V_{INN} is done.

27.10 Module Configuration

The specific configuration for each ACIFB 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 27-3. ACIFB Configuration

Feature	ACIFB
Number of channels	8

Table 27-4. ACIFB Clocks

Clock Name	Description
CLK_ACIFB	Clock for the ACIFB bus interface
GCLK	The generic clock used for the ACIFB is GCLK4

Table 27-5. Register Reset Values

Register	Reset Value
VERSION	0x00000202
PARAMETER	0x000F00FF

28. Capacitive Touch Module (CAT)

Rev: 2.0.0.1

28.1 Features

- **QTouch™ method allows N touch sensors to be implemented using 2N physical pins**
- **QMatrix™ method allows X by Y matrix of sensors to be implemented using (X+2Y) physical pins**
- **One autonomous QTouch™ sensor operates without CPU intervention**
- **Interfaced with peripheral DMA controller to reduce processor overhead**
- **External synchronization to reduce 50 or 60 Hz mains interference**
- **Spread spectrum sensor drive capability**

28.2 Overview

The Capacitive Touch Module (CAT) senses touch on external capacitive touch sensors. Capacitive touch sensors use no external mechanical components, and therefore demand less maintenance in the user application.

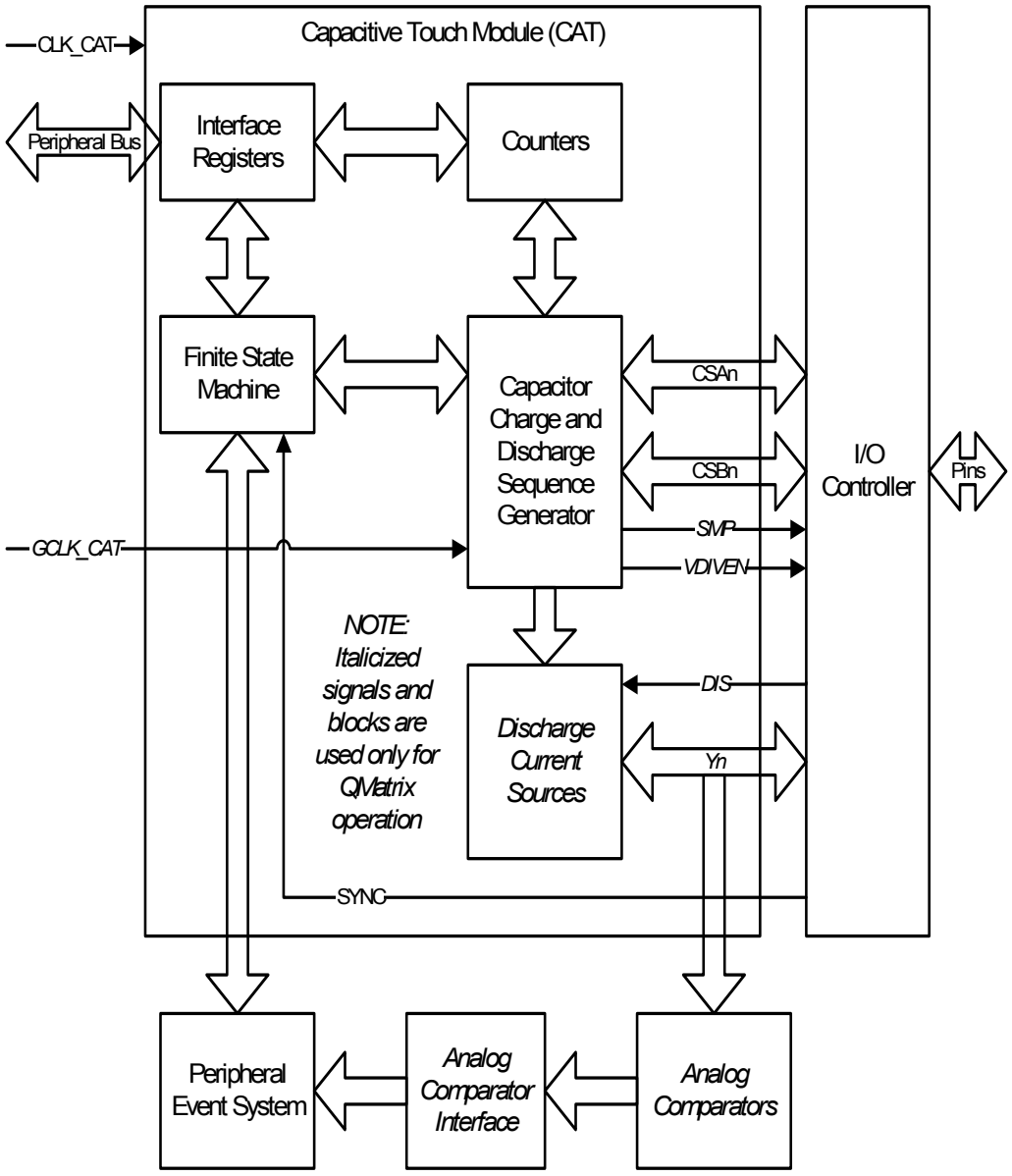
The module implements the QTouch method of capturing signals from capacitive touch sensors. The QTouch method is generally suitable for small numbers of sensors since it requires 2 physical pins per sensor. The module also implements the QMatrix method, which is more appropriate for large numbers of sensors since it allows an X by Y matrix of sensors to be implemented using only (X+2Y) physical pins. The module allows methods to function together, so N touch sensors and an X by Y matrix of sensors can be implemented using (2N+X+2Y) physical pins.

In addition, the module allows sensors using the QTouch method to be divided into two groups. Each QTouch group can be configured with different properties. This eases the implementation of multiple kinds of controls such as push buttons, wheels, and sliders.

The module also implements one autonomous QTouch sensor that is capable of detecting touch or proximity without CPU intervention. This allows proximity or activation detection in low-power sleep modes.

28.3 Block Diagram

Figure 28-1. CAT Block Diagram



28.4 I/O Lines Description

Table 28-1. I/O Lines Description

Name	Description	Type
CSAn	Capacitive sense A line n	I/O
CSBn	Capacitive sense B line n	I/O
DIS	Discharge current control (only used for QMatrix)	Analog
SMP	SMP line (only used for QMatrix)	Output
SYNC	Synchronize signal	Input
VDIVEN	Voltage divider enable (only used for QMatrix)	Output

28.5 Product Dependencies

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

28.5.1 I/O Lines

The CAT pins may be multiplexed with other peripherals. The user must first program the I/O Controller to give control of the pins to the CAT module. In QMatrix mode, the Y lines must be driven by the CAT and analog comparators sense the voltage on the Y lines. Thus, the CAT (not the Analog Comparator Interface) must be the selected function for the Y lines in the I/O Controller.

By writing ones and zeros to bits in the Pin Mode Registers (PINMODEx), most of the CAT pins can be individually selected to implement the QTouch method or the QMatrix method. Each pin has a different name and function depending on whether it is implementing the QTouch method or the QMatrix method. The following table shows the pin names for each method and the bits in the PINMODEx registers which control the selection of the QTouch or QMatrix method.

Table 28-2. Pin Selection Guide

CAT Module Pin Name	QTouch Method Pin Name	QMatrix Method Pin Name	Selection Bit in PINMODEx Register
CSA0	SNS0	X0	SP0
CSB0	SNSK0	X1	SP0
CSA1	SNS1	Y0	SP1
CSB1	SNSK1	YK0	SP1
CSA2	SNS2	X2	SP2
CSB2	SNSK2	X3	SP2
CSA3	SNS3	Y1	SP3
CSB3	SNSK3	YK1	SP3
CSA4	SNS4	X4	SP4
CSB4	SNSK4	X5	SP4
CSA5	SNS5	Y2	SP5

Table 28-2. Pin Selection Guide

CAT Module Pin Name	QTouch Method Pin Name	QMatrix Method Pin Name	Selection Bit in PINMODEx Register
CSB5	SNSK5	YK2	SP5
CSA6	SNS6	X6	SP6
CSB6	SNSK6	X7	SP6
CSA7	SNS7	Y3	SP7
CSB7	SNSK7	YK3	SP7
CSA8	SNS8	X8	SP8
CSB8	SNSK8	X9	SP8
CSA9	SNS9	Y4	SP9
CSB9	SNSK9	YK4	SP9
CSA10	SNS10	X10	SP10
CSB10	SNSK10	X11	SP10
CSA11	SNS11	Y5	SP11
CSB11	SNSK11	YK5	SP11
CSA12	SNS12	X12	SP12
CSB12	SNSK12	X13	SP12
CSA13	SNS13	Y6	SP13
CSB13	SNSK13	YK6	SP13
CSA14	SNS14	X14	SP14
CSB14	SNSK14	X15	SP14
CSA15	SNS15	Y7	SP15
CSB15	SNSK15	YK7	SP15
CSA16	SNS16	X16	SP16
CSB16	SNSK16	X17	SP16

28.5.2 Clocks

The clock for the CAT module, CLK_CAT, is generated by the Power Manager (PM). This clock is turned on by default, and can be enabled and disabled in the PM. The user must ensure that CLK_CAT is enabled before using the CAT module.

QMatrix operations also require the CAT generic clock, GCLK_CAT. This generic clock is generated by the System Control Interface (SCIF), and is shared between the CAT and the Analog Comparator Interface. The user must ensure that the GCLK_CAT is enabled in the SCIF before using QMatrix functionality in the CAT module. For proper QMatrix operation, the frequency of GCLK_CAT must be less than half the frequency of CLK_CAT. If only QTouch functionality is used, then GCLK_CAT is unnecessary.

28.5.3 Interrupts

The CAT interrupt request line is connected to the interrupt controller. Using CAT interrupts requires the interrupt controller to be programmed first.

28.5.4 Peripheral Events

The CAT module receives nine peripheral events, one from the AST and the remaining eight from on-chip analog comparators. These peripheral events are connected via the Peripheral Event System. Refer to the Peripheral Event System chapter for additional details.

28.5.5 Peripheral Direct Memory Access

The CAT module provides handshake capability for a Peripheral DMA Controller. One handshake controls transfers from the Acquired Count Register (ACOUNT) to memory. A second handshake requests burst lengths for each (X,Y) pair to the Matrix Burst Length Register (MBLEN) when using the QMatrix acquisition method. The Peripheral DMA Controller must be configured properly and enabled in order to perform direct memory access transfers to/from the CAT module.

28.5.6 Analog Comparators

When the CAT module is performing QMatrix acquisition, it requires that on-chip analog comparators be used as part of the process. These analog comparators are not controlled directly by the CAT module, but by a separate Analog Comparator (AC) Interface. This interface must be configured properly and enabled before the CAT module is used. This includes configuring the generic clock input for the analog comparators to the proper sampling frequency.

28.5.7 Debug Operation

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

28.6 Functional Description

28.6.1 Acquisition Types

The CAT module can perform three types of QTouch acquisition from capacitive touch sensors: autonomous QTouch (one sensor only), QTouch group A, and QTouch group B. The CAT module can also perform QMatrix acquisition. Each type of acquisition has an associated set of pin selection and configuration registers that allow a large degree of flexibility.

The following schematic diagrams show typical hardware connections for QTouch and QMatrix sensors, respectively:

Figure 28-2. CAT Touch Connections

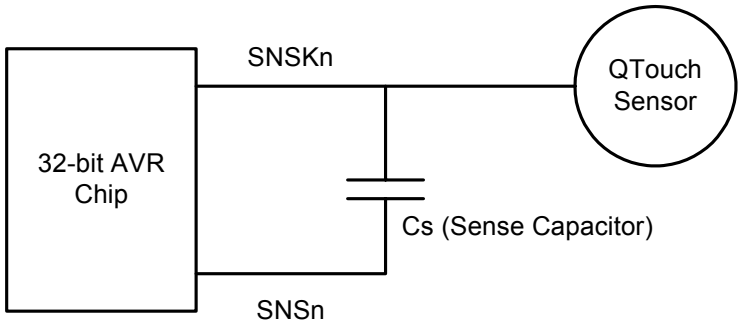
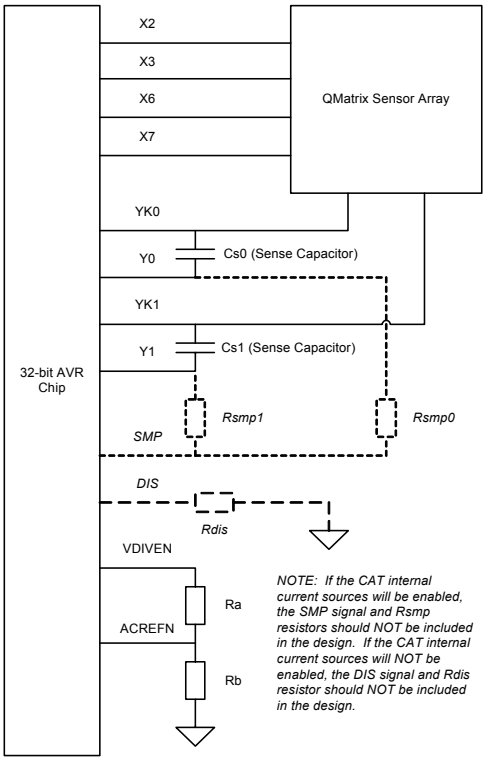


Figure 28-3. CAT Matrix Connections



In order to use the autonomous QTouch detection capability, the user must first set up the Autonomous Touch Pin Select Register (ATPINS) and Autonomous Touch Configuration Registers (ATCFG0 through 3) with appropriate values. The module can then be enabled using the Control Register (CTRL). After the module is enabled, the module will acquire data from the autonomous QTouch sensor and use it to determine whether the sensor is activated. The active/inactive status of the autonomous QTouch sensor is reported in the Status Register (SR), and it is also possible to configure the CAT to generate an interrupt whenever the status changes. The module will continue acquiring autonomous QTouch sensor data and updating autonomous QTouch status until the module is disabled or reset.

In order to use the QMatrix, QTouch group A, or QTouch group B acquisition capabilities, it is first necessary to set up the appropriate pin mode registers (PINMODE0 and PINMODE1) and configuration registers (MGCFG0, MGCFG1, TGACFG0, TGACFG1, TGBCFG0, and TGBCFG1). The module must then be enabled using the CTRL register. In order to initiate acquisition, it is necessary to perform a write to the Acquisition Initiation and Selection Register (AISR). The specific value written to AISR determines which type of acquisition will be performed: QMatrix, QTouch group A, or QTouch group B. The CPU can initiate acquisition by writing to the AISR.

While QMatrix, QTouch group A, or QTouch group B acquisition is in progress, the module collects count values from the sensors and buffers them. Availability of acquired count data is indicated by the Acquisition Ready (ACREADY) bit in the Status Register (SR). The CPU or the Peripheral DMA Controller can then read the acquired counts from the ACOUNT register.

Because the CAT module is configured with Peripheral DMA Controller capability that can transfer data from memory to MBLEN and from ACOUNT to memory, the Peripheral DMA Controller can perform long acquisition sequences and store results in memory without CPU intervention.

28.6.2 Prescaler and Charge Length

Each QTouch acquisition type (autonomous QTouch, QTouch group A, and QTouch group B) has its own prescaler. Each QTouch prescaler divides down the CLK_CAT clock to an appropriate sampling frequency for its particular acquisition type. Typical frequencies are 1 MHz for QTouch acquisition and 4 MHz for QMatrix burst timing control.

Each QTouch prescaler is controlled by the DIV field in the appropriate Configuration Register 0 (ATCFG0, TGACFG0, or TGBCFG0). The QMatrix burst timing prescaler is controlled by the DIV field in MGCFG0. Each prescaler uses the following formula to generate the sampling clock:

Sampling clock = CLK_CAT / (2(DIV+1))

The capacitive sensor charge length, discharge length, and settle length can be determined for each acquisition type using the CHLEN, DILEN, and SELEN fields in Configuration Registers 0 and 1. The lengths are specified in terms of prescaler clocks. In addition, the QMatrix Cx discharge length can be determined using the CXDILEN field in MGCFG2.

For QMatrix acquisition, the duration of CHLEN should not be set to the same value as the period of any periodic signal on any other pin. If the duration of CHLEN is the same as the period of a signal on another pin, it is likely that the other signal will significantly affect measurements due to stray capacitive coupling. For example, if a 1 MHz signal is generated on another pin of the chip, then CHLEN should not be 1 microsecond.

For the QMatrix method, burst and capture lengths are set for each (X,Y) pair by writing the desired length values to the MBLEN register. The write must be done before each X line can start its acquisition and is indicated by the status bit MBLREQ in the Status Register (SR). A DMA handshake interface is also connected to this status bit to reduce CPU overhead during QMatrix acquisitions.

Four burst lengths (BURST0..3) can be written at one time into the MBLEN register. If the current configuration uses Y lines larger than Y3 the register has to be written a second time. The first write to MBLEN specifies the burst length for Y lines 0 to 3 in the BURST0 to BURST3 fields, respectively. The second write specifies the burst length for Y lines 4 to 7 in fields BURST0 to BURST3, respectively, and so on.

The Y and YK pins remain clamped to ground apart from the specified number of burst pulses, when charge is transferred and captured into the sampling capacitor.

28.6.3 Capacitive Count Acquisition

For the QMatrix, QTouch group A, and QTouch group B types of acquisition, the module acquires count values from the sensors, buffers them, and makes them available for reading in the ACOUNT register. Further processing of the count values must be performed by the CPU.

When the module performs QMatrix acquisition using multiple Y lines, it starts the capture for each Y line at the appropriate time in the burst sequence so that all captures finish simultaneously. For example, suppose that an acquisition is performed on Y0 and Y1 with BURST0=53 and BURST1=60. The module will first toggle the X line 7 times while capturing on Y1 while Y0 and YK0 are clamped to ground. The module will then toggle the X line 53 times while capturing on both Y1 and Y0.

28.6.4 Autonomous QTouch

For autonomous QTouch, a complete detection algorithm is implemented within the CAT module. The additional parameters needed to control the autonomous QTouch detection algorithm must be specified by the user in the ATCFG2 and ATCFG3 registers.

Autonomous QTouch sensitivity and out-of-touch sensitivity can be adjusted with the SENSE and OUTSENS fields, respectively, in ATCFG2. Each field accepts values from one to 255 where 255 is the least sensitive setting. The value in the OUTSENS field should be smaller than the value in the SENSE field.

To avoid false positives a detect integration filtering technique can be used. The number of successive detects required is specified in the FILTER field of the ATCFG2 register.

To compensate for changes in capacitance the CAT can recalibrate the autonomous QTouch sensor periodically. The timing of this calibration is done with the NDRIFT and PDRIFT fields in the Configuration register, ATCFG3. It is recommended that the PDRIFT value is smaller than the NDRIFT value.

The autonomous QTouch sensor will also recalibrate if the count value goes too far positive beyond a threshold. This positive recalibration threshold is specified by the PTHR field in the ATCFG3 register.

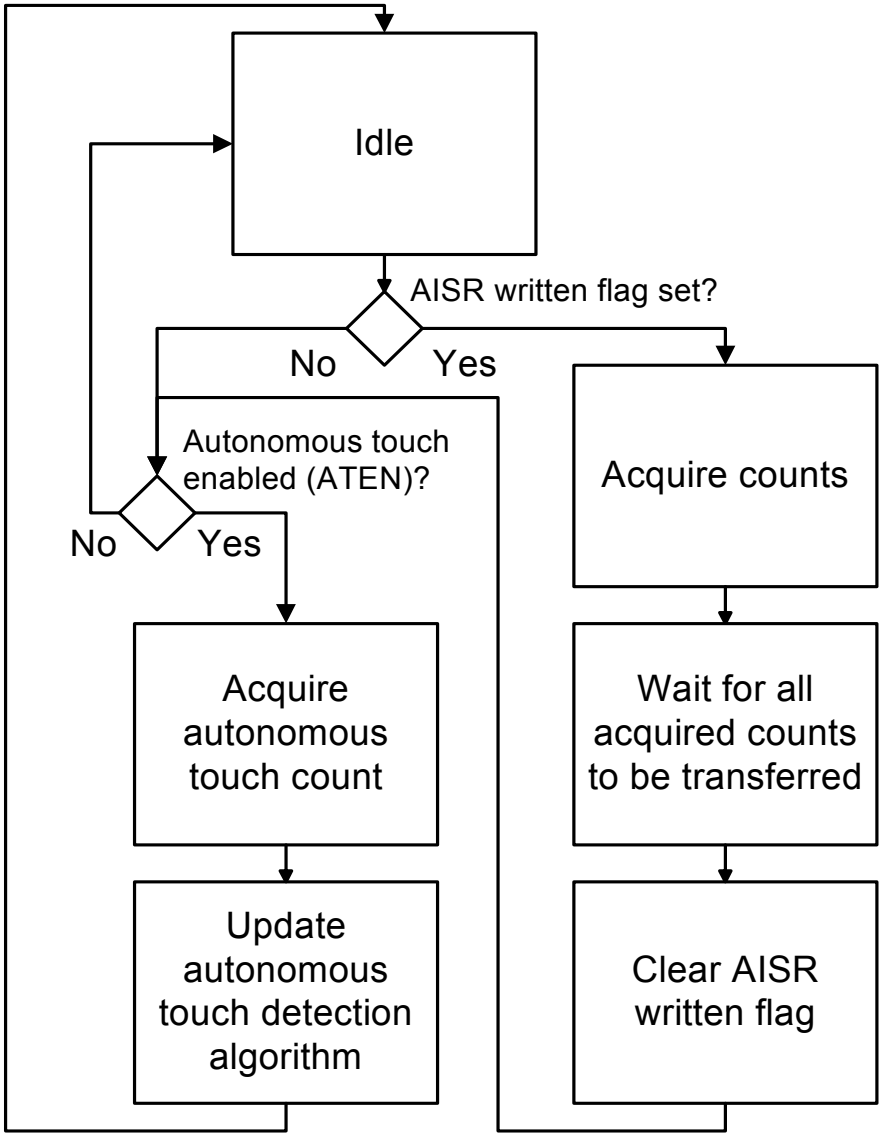
The following block diagram shows the sequence of acquisition and processing operations used by the CAT module. The AISR written bit is internal and not visible in the user interface.

28.6.5 Peripheral Events

The peripheral event from the AST is used to trigger one iteration of autonomous touch detection when the chip is in a sleep mode that has disabled CLK_CAT. When CLK_CAT is disabled and the peripheral event from the AST becomes active, a request will automatically be made to enable CLK_CAT. When CLK_CAT is enabled, the CAT will perform one iteration of the autonomous touch detection algorithm. After this, the CLK_CAT will be disabled, and the CAT module will remain in a frozen state until the next AST peripheral event.

The eight peripheral events from the analog comparators are automatically used by the CAT when performing QMatrix acquisition. The CAT will automatically use the negative peripheral events from the AC Interface on every Y pin in QMatrix mode. When QMatrix acquisition is used the analog comparator corresponding to the selected Y pins must be enabled and converting continuously, using the Y pin as the positive reference and the ACREFN as negative reference.

Figure 28-4. CAT Acquisition and Processing Sequence



28.6.6 Spread Spectrum Sensor Drive

To reduce electromagnetic compatibility issues, the capacitive sensors can be driven with a spread spectrum signal. To enable spread spectrum drive for a specific acquisition type, the user must write a one to the SPREAD bit in the appropriate Configuration Register 1 (MGCFG1, ATCFG1, TGACFG1, or TGBCFG1).

During spread spectrum operation, the length of each pulse within a burst is varied in a deterministic pattern, so that the exact same burst pattern is used for a specific burst length. The maximum spread is determined by the MAXDEV field in the Spread Spectrum Configuration Register (SSCFG) register. The prescaler divisor is varied in a sawtooth pattern from (2(DIV+1))-MAXDEV to (2(DIV+1))+MAXDEV and then back to (2(DIV+1))-MAXDEV. For example, if DIV is 2 and MAXDEV is 3, the prescaler divisor will have the following sequence: 6, 7, 8,

9, 3, 4, 5, 6, 7, 8, 9, 3, 4, etc. MAXDEV must not exceed the value of (2(DIV+1)), or undefined behavior will occur.

28.6.7 Synchronization

To prevent interference from the 50 or 60 Hz mains line the CAT can trigger acquisition on the SYNC signal. The SYNC signal should be derived from the mains line. The acquisition will trigger on a falling edge of this signal. To enable synchronization for a specific acquisition type, the user must write a one to the SYNC bit in the appropriate Configuration Register 1 (MGCFG1, ATCFG1, TGACFG1, or TGBCFG1).

For QMatrix acquisition, all X lines must be sampled at a specific phase of the noise signal for the synchronization to be effective. This can be accomplished by the synchronization timer, which is enabled by writing a non-zero value to the SYNCTIM field in the MGCFG2 register. This ensures that the start of the acquisition of each X line is spaced at regular intervals, defined by the SYNCTIM field.

28.6.8 Resistive Drive

By default, the CAT pins are driven with normal I/O drive properties. Some of the CSA and CSB pins can optionally drive with a 1k output resistance for improved EMC. The pins that have this capability are listed in the Module Configuration section.

28.6.9 Discharge Current Source

The device integrates a discharge current source, which can be used to discharge the sampling capacitors during the QMatrix measurement phase. The discharge current source is enabled by writing the EN bits in the Discharge Current Source (DICS) register to one. This enables an internal reference voltage, which can be the internal 1.1V band gap voltage or VDDIO/3, as selected by the INTVREFSEL bit in the DICS register. If the DICS.INTREFSEL bit is one, the reference voltage is applied across an internal resistor. Otherwise the voltage is applied to the DIS pin, and an external resistor must be connected between DIS and ground. This allows the discharge current to be programmed between 2 and 20 µA.

The reference current is mirrored to each Y-pin if the corresponding bit is written to one in the DICS.SOURCES field.

The reference current can be fine-tuned by writing the trim value to the DICS.TRIM field, allowing the user to compensate e.g. for temperature gradients in the resistance value.

28.7 User Interface

Table 28-3. CAT Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CTRL	Read/Write	0x00000000
0x04	Autonomous Touch Pin Selection Register	ATPINS	Read/Write	0x00000000
0x08	Pin Mode Register 0	PINMODE0	Read/Write	0x00000000
0x0C	Pin Mode Register 1	PINMODE1	Read/Write	0x00000000
0x10	Autonomous Touch Configuration Register 0	ATCFG0	Read/Write	0x00000000
0x14	Autonomous Touch Configuration Register 1	ATCFG1	Read/Write	0x00000000
0x18	Autonomous Touch Configuration Register 2	ATCFG2	Read/Write	0x00000000
0x1C	Autonomous Touch Configuration Register 3	ATCFG3	Read/Write	0x00000000
0x20	Touch Group A Configuration Register 0	TGACFG0	Read/Write	0x00000000
0x24	Touch Group A Configuration Register 1	TGACFG1	Read/Write	0x00000000
0x28	Touch Group B Configuration Register 0	TGBCFG0	Read/Write	0x00000000
0x2C	Touch Group B Configuration Register 1	TGBCFG1	Read/Write	0x00000000
0x30	Matrix Group Configuration Register 0	MGCFG0	Read/Write	0x00000000
0x34	Matrix Group Configuration Register 1	MGCFG1	Read/Write	0x00000000
0x38	Matrix Group Configuration Register 2	MGCFG2	Read/Write	0x00000000
0x3C	Status Register	SR	Read-only	0x00000000
0x40	Status Clear Register	SCR	Write-only	-
0x44	Interrupt Enable Register	IER	Write-only	-
0x48	Interrupt Disable Register	IDR	Write-only	-
0x4C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x50	Acquisition Initiation and Selection Register	AISR	Read/Write	0x00000000
0x54	Acquired Count Register	ACOUNT	Read-only	0x00000000
0x58	Matrix Burst Length Register	MBLEN	Write-only	-
0x5C	Discharge Current Source Register	DICS	Read/Write	0x00000000
0x60	Spread Spectrum Configuration Register	SSCFG	Read/Write	0x00000000
0x64	CSA Resistor Control Register	CSARES	Read/Write	0x00000000
0x68	CSB Resistor Control Register	CSBRES	Read/Write	0x00000000
0x6C	Autonomous Touch Base Count Register	ATBASE	Read-only	0x00000000
0x70	Autonomous Touch Current Count Register	ATCURR	Read-only	0x00000000
0x80	Analog Comparator Shift Offset Register 0	ACSHI0	Read/Write	0x00000000
0x84	Analog Comparator Shift Offset Register 1	ACSHI1	Read/Write	0x00000000
0x88	Analog Comparator Shift Offset Register 2	ACSHI2	Read/Write	0x00000000
0x8C	Analog Comparator Shift Offset Register 3	ACSHI3	Read/Write	0x00000000
0x90	Analog Comparator Shift Offset Register 4	ACSHI4	Read/Write	0x00000000
0x94	Analog Comparator Shift Offset Register 5	ACSHI5	Read/Write	0x00000000

Table 28-3. CAT Register Memory Map

Offset	Register	Register Name	Access	Reset
0x98	Analog Comparator Shift Offset Register 6	ACSHI6	Read/Write	0x00000000
0x9C	Analog Comparator Shift Offset Register 7	ACSHI7	Read/Write	0x00000000
0xF8	Parameter Register	PARAMETER	Read-only	
0xFC	Version Register	VERSION	Read-only	

28.7.1 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
SWRST	-	-	-	-	-	-	EN

- **SWRST: Software reset**
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets the module. The module will be disabled after the reset.
  This bit always reads as zero.
- **EN: Module enable**
  0: Module is disabled.
  1: Module is enabled.

28.7.2 Autonomous Touch Pin Selection Register

Name: ATPINS

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
-	-	-	-	-	-	-	ATEN

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

- **ATEN: Autonomous Touch Enable**
  0: Autonomous QTouch acquisition and detection is disabled.
  1: Autonomous QTouch acquisition and detection is enabled using the sense pair specified in ATSP.
- **ATSP: Autonomous Touch Sense Pair**
  Selects the sense pair that will be used by the autonomous QTouch sensor. A value of n will select sense pair n (CSAn and CSBn pins).

28.7.3 Pin Mode Registers 0 and 1

Name: PINMODE0 and PINMODE1

Access Type: Read/Write

Offset: 0x08, 0x0C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
-							SP[16]

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

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

- **SP: Sense Pair Mode Selection**

  Each SP[n] bit determines the operation mode of sense pair n (CSAn and CSBn pins). The (PINMODE1.SP[n] PINMODE0.SP[n]) bits have the following definitions:

  00: Sense pair n disabled.

  01: Sense pair n is assigned to QTouch Group A.

  10: Sense pair n is assigned to QTouch Group B.

  11: Sense pair n is assigned to the QMatrix Group.

28.7.4 Autonomous Touch Configuration Register 0

Name: ATCFG0

Access Type: Read/Write

Offset: 0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
DIV[15:8]							

23	22	21	20	19	18	17	16
DIV[7:0]							

15	14	13	12	11	10	9	8
CHLEN							

7	6	5	4	3	2	1	0
SELEN							

- **DIV: Clock Divider**
  The prescaler is used to ensure that the CLK_CAT clock is divided to around 1 MHz to produce the sampling clock.The prescaler uses the following formula to generate the sampling clock:
  Sampling clock = CLK_CAT / (2(DIV+1))
- **CHLEN: Charge Length**
  For the autonomous QTouch sensor, specifies how many sample clock cycles should be used for transferring charge to the sense capacitor.
- **SELEN: Settle Length**
  For the autonomous QTouch sensor, specifies how many sample clock cycles should be used for settling after charge transfer.

28.7.5 Autonomous Touch Configuration Register 1

Name: ATCFG1

Access Type: Read/Write

Offset: 0x14

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-		DISHIFT		-		SYNC	SPREAD

23	22	21	20	19	18	17	16
DILEN							

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

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

- **DISHIFT: Discharge Shift**
  For the autonomous QTouch sensor, specifies how many bits the DILEN field should be shifted before using it to determine the discharge time.
- **SYNC: Sync Pin**
  For the autonomous QTouch sensor, specifies that acquisition shall begin when a falling edge is received on the SYNC line.
- **SPREAD: Spread Spectrum Sensor Drive**
  For the autonomous QTouch sensor, specifies that spread spectrum sensor drive shall be used.
- **DILEN: Discharge Length**
  For the autonomous QTouch sensor, specifies how many sample clock cycles the CAT should use to discharge the capacitors before charging them.
- **MAX: Maximum Count**
  For the autonomous QTouch sensor, specifies how many counts the maximum acquisition should be.

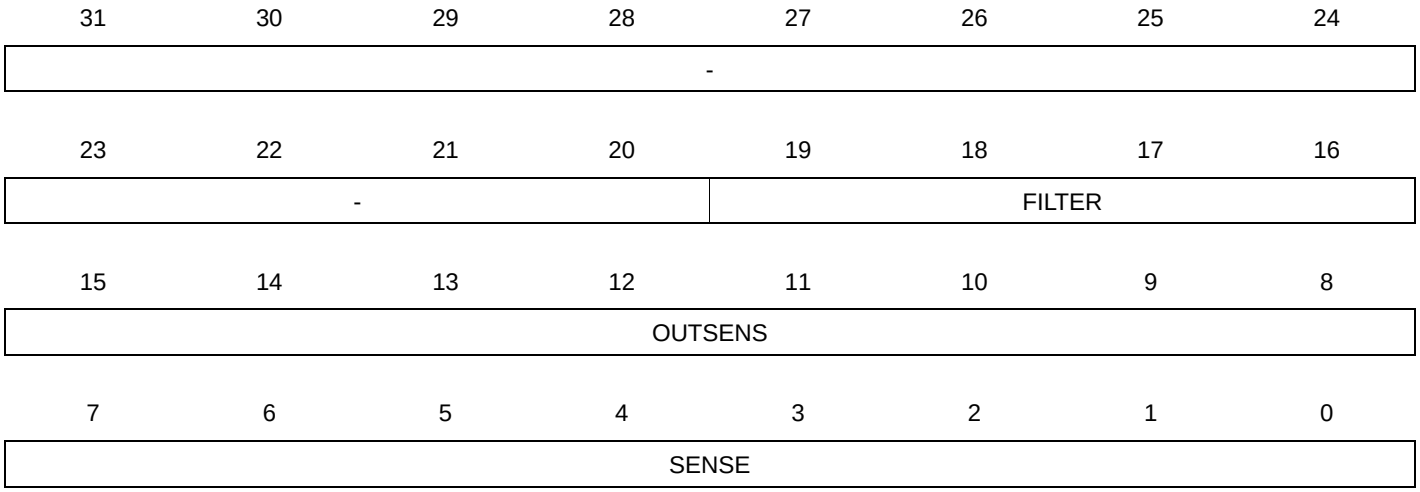
28.7.6 Autonomous Touch Configuration Register 2

Name: ATCFG2

Access Type: Read/Write

Offset: 0x18

Reset Value: 0x00000000



- **FILTER: Autonomous Touch Filter Setting**
  For the autonomous QTouch sensor, specifies how many positive detects in a row the CAT needs to have on the autonomous QTouch sensor before reporting it as a touch. A FILTER value of 0 is not allowed and will result in undefined behavior.
- **OUTSENS: Out-of-Touch Sensitivity**
  For the autonomous QTouch sensor, specifies how sensitive the out-of-touch detector should be.
- **SENSE: Sensitivity**
  For the autonomous QTouch sensor, specifies how sensitive the touch detector should be.

28.7.7 Autonomous Touch Configuration Register 3

Name: ATCFG3

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
PTHR							

15	14	13	12	11	10	9	8
PDRIFT							

7	6	5	4	3	2	1	0
NDRIFT							

- **PTHR: Positive Recalibration Threshold**
  For the autonomous QTouch sensor, specifies how far a sensor's signal must move in a positive direction from the reference in order to cause a recalibration.
- **PDRIFT: Positive Drift Compensation**
  For the autonomous QTouch sensor, specifies how often a positive drift compensation should be performed. When this field is zero, positive drift compensation will never be performed. When this field is non-zero, the positive drift compensation time interval is given by the following formula:
  Tpdrift = PDRIFT * 65536 * (sample clock period)
- **NDRIFT: Negative Drift Compensation**
  For the autonomous QTouch sensor, specifies how often a negative drift compensation should be performed. When this field is zero, negative drift compensation will never be performed. When this field is non-zero, the negative drift compensation time interval is given by the following formula:
  Tndrift = NDRIFT * 65536 * (sample clock period)

With the typical sample clock frequency of 1 MHz, PDRIFT and NDRIFT can be set from 0.066 seconds to 16.7 seconds with 0.066 second resolution.

28.7.8 Touch Group x Configuration Register 0

Name: TGxCFG0

Access Type: Read/Write

Offset: 0x20, 0x28

Reset Value: 0x00000000

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

- **DIV: Clock Divider**
  The prescaler is used to ensure that the CLK_CAT clock is divided to around 1 MHz to produce the sampling clock.The prescaler uses the following formula to generate the sampling clock:
  Sampling clock = CLK_CAT / (2(DIV+1))
- **CHLEN: Charge Length**
  For the QTouch method, specifies how many sample clock cycles should be used for transferring charge to the sense capacitor.
- **SELEN: Settle Length**
  For the QTouch method, specifies how many sample clock cycles should be used for settling after charge transfer.

28.7.9 Touch Group x Configuration Register 1

Name: TGxCFG1

Access Type: Read/Write

Offset: 0x24, 0x2C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	DISHIFT		-	-	SYNC	SPREAD

23	22	21	20	19	18	17	16
DILEN							

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

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

- **DISHIFT: Discharge Shift**
  For the sensors in QTouch group x, specifies how many bits the DILEN field should be shifted before using it to determine the discharge time.
- **SYNC: Sync Pin**
  For sensors in QTouch group x, specifies that acquisition shall begin when a falling edge is received on the SYNC line.
- **SPREAD: Spread Spectrum Sensor Drive**
  For sensors in QTouch group x, specifies that spread spectrum sensor drive shall be used.
- **DILEN: Discharge Length**
  For sensors in QTouch group x, specifies how many clock cycles the CAT should use to discharge the capacitors before charging them.
- **MAX: Touch Maximum Count**
  For sensors in QTouch group x, specifies how many counts the maximum acquisition should be.

28.7.10 Matrix Group Configuration Register 0

Name: MGCFG0

Access Type: Read/Write

Offset: 0x30

Reset Value: 0x00000000

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

- **DIV: Clock Divider**
  The prescaler is used to ensure that the CLK_CAT clock is divided to around 4 MHz to produce the burst timing clock.The prescaler uses the following formula to generate the burst timing clock:
  Burst timing clock = CLK_CAT / (2(DIV+1))
- **CHLEN: Charge Length**
  For QMatrix sensors, specifies how many burst prescaler clock cycles should be used for transferring charge to the sense capacitor.
- **SELEN: Settle Length**
  For QMatrix sensors, specifies how many burst prescaler clock cycles should be used for settling after charge transfer.

28.7.11 Matrix Group Configuration Register 1

Name: MGCFG1

Access Type: Read/Write

Offset: 0x34

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-		DISHIFT		-		SYNC	SPREAD

23	22	21	20	19	18	17	16
DILEN							

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

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

- **DISHIFT: Discharge Shift**
  For QMatrix sensors, specifies how many bits the DILEN field should be shifted before using it to determine the discharge time.
- **SYNC: Sync Pin**
  For QMatrix sensors, specifies that acquisition shall begin when a falling edge is received on the SYNC line.
- **SPREAD: Spread Spectrum Sensor Drive**
  For QMatrix sensors, specifies that spread spectrum sensor drive shall be used.
- **DILEN: Discharge Length**
  For QMatrix sensors, specifies how many burst prescaler clock cycles the CAT should use to discharge the capacitors at the beginning of a burst sequence.
- **MAX: Maximum Count**
  For QMatrix sensors, specifies how many counts the maximum acquisition should be.

28.7.12 Matrix Group Configuration Register 2

Name: MGCFG2

Access Type: Read/Write

Offset: 0x38

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
ACCTRL	CONSEN			-			

23	22	21	20	19	18	17	16
CXDILEN							

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

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

- **ACCTRL: Analog Comparator Control**
  When written to one, allows the CAT to disable the analog comparators when they are not needed. When written to zero, the analog comparators are always enabled.
- **CONSEN: Consensus Filter Length**
  For QMatrix sensors, specifies that discharge will be terminated when CONSEN out of the most recent 5 comparator samples are positive. For example, a value of 3 in the CONSEN field will terminate discharge when 3 out of the most recent 5 comparator samples are positive. When CONSEN has the default value of 0, discharge will be terminated immediately when the comparator output goes positive.
- **CXDILEN: Cx Capacitor Discharge Length**
  For QMatrix sensors, specifies how many burst prescaler clock cycles the CAT should use to discharge the Cx capacitor at the end of each burst cycle.
- **SYNCTIM: Sync Time Interval**
  When non-zero, determines the number of prescaled clock cycles between the start of the acquisition on each X line for QMatrix acquisition.

28.7.13 Status Register

Name: SR

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
-	-	-	-	-	-	ACQDONE	ACREADY
7	6	5	4	3	2	1	0
-	-	-	MBLREQ	ATSTATE	ATSC	ATCAL	ENABLED

- **ACQDONE: Acquisition Done**
  0: Acquisition is not done (still in progress).
  1: Acquisition is complete.
- **ACREADY: Acquired Count Data is Ready**
  0: Acquired count data is not available in the ACOUNT register.
  1: Acquired count data is available in the ACOUNT register.
- **MBLREQ: Matrix Burst Length Required**
  0: The QMatrix acquisition does not require any burst lengths.
  1: The QMatrix acquisition requires burst lengths for the current X line.
- **ATSTATE: Autonomous Touch Sensor State**
  0: The autonomous QTouch sensor is not active.
  1: The autonomous QTouch sensor is active.
- **ATSC: Autonomous Touch Sensor Status Interrupt**
  0: No status change in the autonomous QTouch sensor.
  1: Status change in the autonomous QTouch sensor.
- **ATCAL: Autonomous Touch Calibration Ongoing**
  0: The autonomous QTouch sensor is not calibrating.
  1: The autonomous QTouch sensor is calibrating.
- **ENABLED: Module Enabled**
  0: The module is disabled.
  1: The module is enabled.

28.7.14 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x40

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
-	-	-	-	-	-	ACQDONE	ACREADY
7	6	5	4	3	2	1	0
-	-	-	-	-	ATSC	ATCAL	-

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.

28.7.15 Interrupt Enable Register

Name: IER

Access Type: Write-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
-	-	-	-	-	-	ACQDONE	ACREADY
7	6	5	4	3	2	1	0
-	-	-	-	-	ATSC	ATCAL	-

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.

28.7.16 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x48

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
-	-	-	-	-	-	ACQDONE	ACREADY
7	6	5	4	3	2	1	0
-	-	-	-	-	ATSC	ATCAL	-

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.

28.7.17 Interrupt Mask Register

Name: IMR

Access Type: Read-only

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
-	-	-	-	-	-	ACQDONE	ACREADY
7	6	5	4	3	2	1	0
-	-	-	-	-	ATSC	ATCAL	-

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.

28.7.18 Acquisition Initiation and Selection Register

Name: AISR

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
-						ACQSEL	

- **ACQSEL: Acquisition Type Selection**

  A write to this register initiates an acquisition of the following type:

  00: QTouch Group A.

  01: QTouch Group B.

  10: QMatrix Group.

  11: Undefined behavior.

  A read of this register will return the value that was previously written.

28.7.19 Acquired Count Register

Name: ACOUNT

Access Type: Read-only

Offset: 0x54

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
Y							

23	22	21	20	19	18	17	16
SPORX							

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

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

- **Y: Y index**
  The Y index (for QMatrix method) associated with this count value.
- **SPORX: Sensor pair or X index**
  The sensor pair index (for QTouch method) or X index (for QMatrix method) associated with this count value.
- **COUNT: Count value**
  The signal (number of counts) acquired on the channel specified in the SPORX and Y fields.

When multiple acquired count values are read from a QTouch acquisition, the Y field will always be 0 and the SPORX value will increase monotonically. For example, suppose a QTouch acquisition is performed using sensor pairs SP1, SP4, and SP9. The first count read will have SPORX=1, the second read will have SPORX=4, and the third read will have SPORX=9.
When multiple acquired count values are read from a QMatrix acquisition, the SPORX value will stay the same while Y increases monotonically through all Y values in the group. Then SPORX will increase to the next X value in the group. For example, a QMatrix acquisition with X=2,3 and Y=4,7 would provide count values in the following order: X=2 and Y=4, then X=2 and Y=7, then X=3 and Y=4, and finally X=3 and Y=7.

28.7.20 Matrix Burst Length Register

Name: MBLEN

Access Type: Write-only

Offset: 0x58

Reset Value: -

31	30	29	28	27	26	25	24
BURST0							
23	22	21	20	19	18	17	16
BURST1							
15	14	13	12	11	10	9	8
BURST2							
7	6	5	4	3	2	1	0
BURST3							

- **BURSTx: Burst Length x**

  For QMatrix sensors, specifies how many times the switching sequence should be repeated before acquisition begins for each channel. Each count in the BURSTx field specifies 1 repeat of the switching sequence, so the actual burst length will be BURST. Before doing a QMatrix acquisition on one X line this register has to be written with the burst values for the current XY pairs. For each X line this register needs to be programmed with all the Y values. If Y values larger than 3 are used the register has to be written several times in order to specify all burst lengths.

  The Status Register bit MBLREQ is set to 1 when the CAT is waiting for values to be written into this register.

28.7.21 Discharge Current Source Register

Name: DICS

Access Type: Read/Write

Offset: 0x5C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
FSOURCES[7:0]							
23	22	21	20	19	18	17	16
GLEN	-	-	-	-	-	INTVREFSEL	INTREFSEL
15	14	13	12	11	10	9	8
-	-	-	TRIM				
7	6	5	4	3	2	1	0
SOURCES[7:0]							

- **FSOURCES: Force Discharge Current Sources**
  When FSOURCES[n] is 0, the corresponding discharge current source behavior depends on SOURCES[n].
  When FSOURCES[n] is 1, the corresponding discharge current source is forced to be enabled continuously. This is useful for testing or debugging but should not be done during normal acquisition.
- **GLEN: Global Enable**
  0: The current source module is globally disabled.
  1: The current source module is globally enabled.
- **INTVREFSEL: Internal Voltage Reference Select**
  0: The voltage for the reference resistor is generated from the internal band gap circuit.
  1: The voltage for the reference resistor is VDDIO/3.
- **INTREFSEL: Internal Reference Select**
  0: The reference current flows through an external resistor on the DIS pin.
  1: The reference current flows through the internal reference resistor.
- **TRIM: Reference Current Trimming**
  This field is used to trim the discharge current. 0x00 corresponds to the minimum current value, and 0x1F corresponds to the maximum current value.
- **SOURCES: Enable Discharge Current Sources**
  When SOURCES[n] is 0, the corresponding discharge current source is disabled.
  When SOURCES[n] is 1, the corresponding discharge current source is enabled at appropriate times during acquisition.

28.7.22 Spread Spectrum Configuration Register

Name: SSCFG

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
MAXDEV							

- **MAXDEV: Maximum Deviation**
  When spread spectrum burst is enabled, MAXDEV indicates the maximum number of prescaled clock cycles the burst pulse will be extended or shortened.

28.7.23 CSA Resistor Control Register

Name: CSARES

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
-							RES[16]

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

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

- **RES: Resistive Drive Enable**
  When RES[n] is 0, CSA[n] has the same drive properties as normal I/O pads.
  When RES[n] is 1, CSA[n] has a nominal output resistance of 1kOhm during the burst phase.

28.7.24 CSB Resistor Control Register

Name: CSBRES

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
-							RES[16]

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

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

- **RES: Resistive Drive Enable**
  When RES[n] is 0, CSB[n] has the same drive properties as normal I/O pads.
  When RES[n] is 1, CSB[n] has a nominal output resistance of 1kOhm during the burst phase.

28.7.25 Autonomous Touch Base Count Register

Name: ATBASE

Access Type: Read-only

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
COUNT[15:8]							

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

- **COUNT: Count value**
  The base count currently stored by the autonomous touch sensor. This is useful for autonomous touch debugging purposes.

28.7.26 Autonomous Touch Current Count Register

Name: ATCURR

Access Type: Read-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
COUNT[15:8]							

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

- **COUNT: Count value**
  The current count acquired by the autonomous touch sensor. This is useful for autonomous touch debugging purposes.

28.7.27 Analog Comparator Shift Offset Register x

Name: ACSHIx

Access Type: Read/Write

Offset: 0x80, 0x84, 0x88, 0x8C, 0x90, 0x94, 0x98, and 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
-				SHIVAL[11:8]			

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

- **SHIVAL: Shift Offset Value**
  Specifies the amount to shift the count value from each comparator. This allows the offset of each comparator to be compensated.

28.7.28 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0xF8

Reset Value: -

31	30	29	28	27	26	25	24
-							

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

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

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

- **SP[n]: Sensor pair implemented**
  0: The corresponding sensor pair is not implemented
  1: The corresponding sensor pair is implemented.

28.7.29 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.

28.8 Module Configuration

The specific configuration the CAT module 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 28-4. Module Configuration

Feature	CAT
Number of touch sensors/Size of matrix	Allows up to 17 touch sensors, or up to 16 by 8 matrix sensors to be interfaced.

Table 28-5. CAT Clocks

Clock Name	Description
CLK_CAT	Clock for the CAT bus interface
GCLK	The generic clock used for the CAT is GCLK4

Table 28-6. Register Reset Values

Register	Reset Value
VERSION	0x00000200
PARAMETER	0x0001FFFF

28.8.1 Resistive Drive

By default, the CAT pins are driven with normal I/O drive properties. Some of the CSA and CSB pins can optionally drive with a 1k output resistance for improved EMC.

To enable resistive drive on a pin, the user must write a one to the corresponding bit in the CSA Resistor Control Register (CSARES) or CSB Resistor Control Register (CSBRES) register.

29. Glue Logic Controller (GLOC)

Rev.: 1.0.0.0

29.1 Features

- **Glue logic for general purpose PCB design**
- **Programmable lookup table**
- **Up to four inputs supported per lookup table**
- **Optional filtering of output**

29.2 Overview

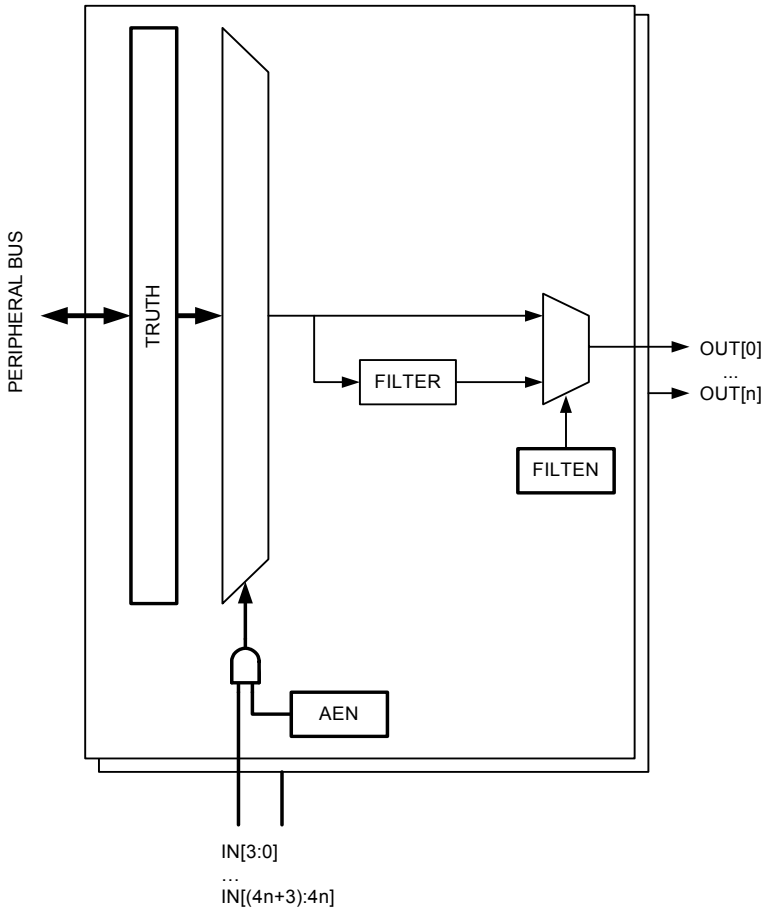
The Glue Logic Controller (GLOC) contains programmable logic which can be connected to the device pins. This allows the user to eliminate logic gates for simple glue logic functions on the PCB.

The GLOC consists of a number of lookup table (LUT) units. Each LUT can generate an output as a user programmable logic expression with four inputs. Inputs can be individually masked.

The output can be combinatorially generated from the inputs, or filtered to remove spikes.

29.3 Block Diagram

Figure 29-1. GLOC Block Diagram



29.4 I/O Lines Description

Table 29-1. I/O Lines Description

Pin Name	Pin Description	Type
IN0-INm	Inputs to lookup tables	Input
OUT0-OUTn	Output from lookup tables	Output

Each LUT have 4 inputs and one output. The inputs and outputs for the LUTs are mapped sequentially to the inputs and outputs. This means that LUT0 is connected to IN0 to IN3 and OUT0. LUT1 is connected to IN4 to IN7 and OUT1. In general, LUTn is connected to IN[4n] to IN[4n+3] and OUTn.

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

The pins used for interfacing the GLOC may be multiplexed with I/O Controller lines. The programmer must first program the I/O Controller to assign the desired GLOC pins to their peripheral function. If I/O lines of the GLOC 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 GLOC input and outputs actually in use.

29.5.2 Clocks

The clock for the GLOC bus interface (CLK_GLOC) 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 GLOC before disabling the clock, to avoid freezing the module in an undefined state.

Additionally, the GLOC depends on a dedicated Generic Clock (GCLK). The GCLK can be set to a wide range of frequencies and clock sources, and must be enabled by the System Control Interface (SCIF) before the GLOC filter can be used.

29.5.3 Debug Operation

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

29.6 Functional Description

29.6.1 Enabling the Lookup Table Inputs

Since the inputs to each lookup table (LUT) unit can be multiplexed with other peripherals, each input must be explicitly enabled by writing a one to the corresponding enable bit (AEN) in the corresponding Lookup Table Control Register (LUTCR).

If no inputs are enabled, the output OUTn will be the least significant bit in the TRUTHn register.

29.6.2 Configuring the Lookup Table

The lookup table in each LUT unit can generate any logic expression OUT as a function of up to four inputs, IN[3:0]. The truth table for the expression is written to the TRUTH register for the LUT. Table 29-2 shows the truth table for LUT0. The truth table for LUTn is written to TRUTHn, and the corresponing input and outputs will be IN[4n] to IN[4n+3] and OUTn.

Table 29-2. Truth Table for the Lookup Table in LUT0

IN[3]	IN[2]	IN[1]	IN[0]	OUT[0]
0	0	0	0	TRUTH0[0]
0	0	0	1	TRUTH0[1]
0	0	1	0	TRUTH0[2]
0	0	1	1	TRUTH0[3]
0	1	0	0	TRUTH0[4]
0	1	0	1	TRUTH0[5]
0	1	1	0	TRUTH0[6]
0	1	1	1	TRUTH0[7]
1	0	0	0	TRUTH0[8]
1	0	0	1	TRUTH0[9]
1	0	1	0	TRUTH0[10]
1	0	1	1	TRUTH0[11]
1	1	0	0	TRUTH0[12]
1	1	0	1	TRUTH0[13]
1	1	1	0	TRUTH0[14]
1	1	1	1	TRUTH0[15]

29.6.3 Output Filter

By default, the output OUTn is a combinatorial function of the inputs IN[4n] to IN[4n+3]. This may cause some short glitches to occur when the inputs change value.

It is also possible to clock the output through a filter to remove glitches. This requires that the corresponding generic clock (GCLK) has been enabled before use. The filter can then be enabled by writing a one to the Filter Enable (FILTEN) bit in LUTCRn. The OUTn output will be delayed by three to four GCLK cycles when the filter is enabled.

29.7 User Interface

Table 29-3. GLOC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00+n*0x08	Lookup Table Control Register n	LUTCRn	Read/Write	0x00000000
0x04+n*0x08	Truth Table Register n	TRUTHn	Read/Write	0x00000000
0x38	Parameter Register	PARAMETER	Read-only	- (1)
0x3C	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.

29.7.1 Lookup Table Control Register n

Name: LUTCRn

Access Type: Read/Write

Offset: 0x00+n*0x08

Reset Value: 0x00000000

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

- **FILTEN: Filter Enable**
  1: The output is glitch filtered
  0: The output is not glitch filtered
- **AEN: Enable IN Inputs**
  Input IN[n] is enabled when AEN[n] is one.
  Input IN[n] is disabled when AEN[n] is zero, and will not affect the OUT value.

29.7.2 Truth Table Register n

Name: TRUTHn

Access Type: Read/Write

Offset: 0x04+n*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
TRUTH[15:8]							

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

- **TRUTH: Truth Table Value**

  This value defines the output OUT as a function of inputs IN:
  OUT = TRUTH[IN]

29.7.3 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x38

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
LUTS							

- **LUTS: Lookup Table Units Implemented**
  This field contains the number of lookup table units implemented in this device.

29.7.4 VERSION Register

Name: VERSION

Access Type: Read-only

Offset: 0x3C

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.

29.8 Module Configuration

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

Table 29-4. GLOC Configuration

Feature	GLOC
Number of LUT units	2

Table 29-5. GLOC Clocks

Clock Name	Description
CLK_GLOC	Clock for the GLOC bus interface
GCLK	The generic clock used for the GLOC is GCLK5

Table 29-6. Register Reset Values

Register	Reset Value
VERSION	0x00000100
PARAMETER	0x00000002

30. aWire UART (AW)

Rev.: 2.1.0.0

30.1 Features

- **Asynchronous receiver or transmitter when the aWire system is not used for debugging.**
- **One- or two-pin operation supported.**

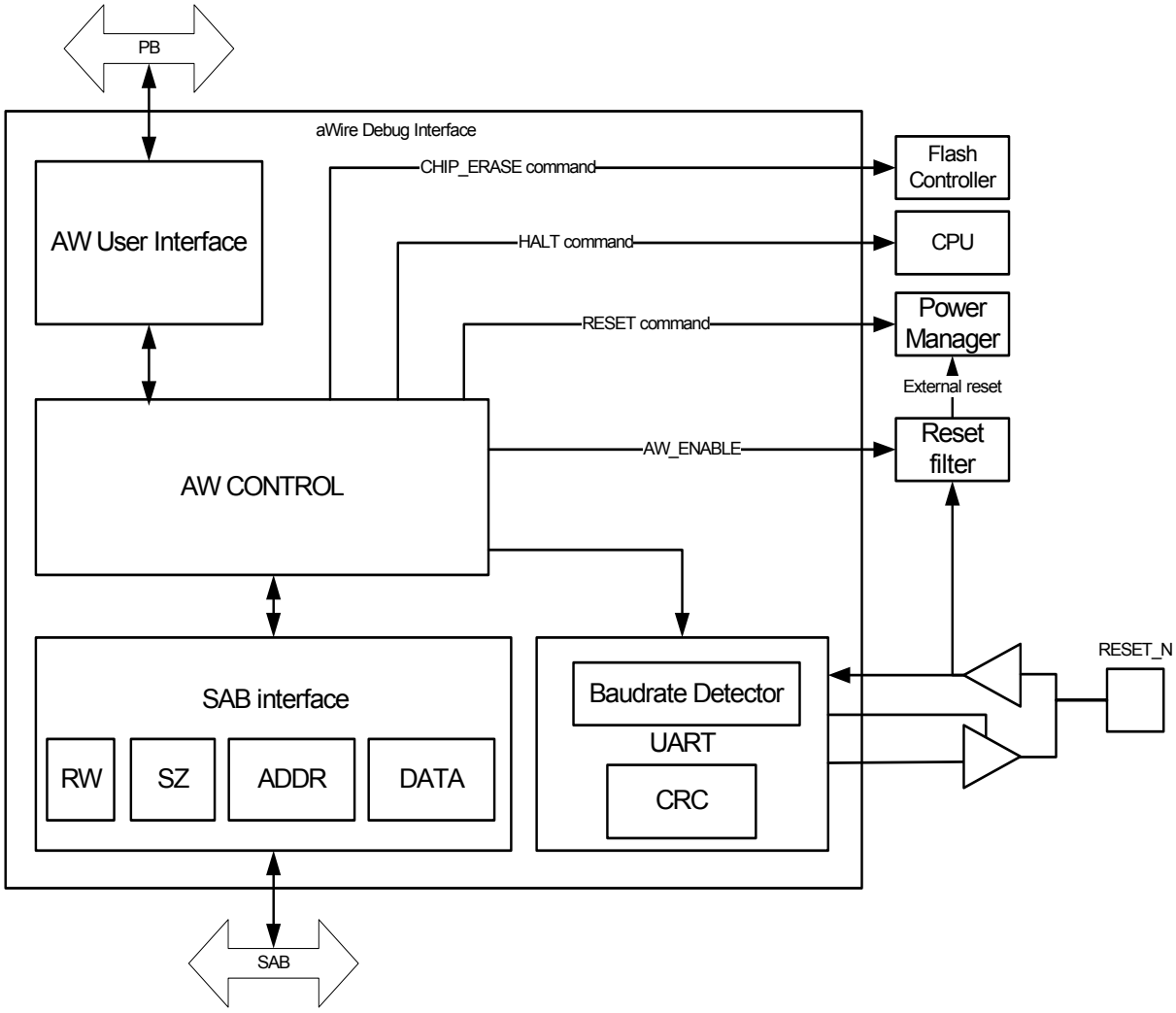
30.2 Overview

If the AW is not used for debugging, the aWire UART can be used by the user to send or receive data with one stop 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.

30.3 Block Diagram

Figure 30-1. aWire Debug Interface Block Diagram



30.4 I/O Lines Description

Table 30-1. I/O Lines Description

Name	Description	Type
DATA	aWire data multiplexed with the RESET_N pin.	Input/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 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.

30.5.2 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.

30.5.3 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 30.6.1).

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.

30.5.4 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.

30.5.5 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.

30.5.6 External Components

The AW needs an external pullup on the RESET_N pin to ensure that the pin is pulled up when the bus is not driven.

30.6 Functional Description

The aWire UART user interface can be used as a spare Asynchronous Receiver or Transmitter when AW is not used for debugging.

30.6.1 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 the 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.

30.6.2 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.

30.6.3 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.

30.6.4 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.

30.6.5 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.

30.6.6 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.

30.6.7 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.

30.7 User Interface

Table 30-2. 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.

30.7.1 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 30-3. aWire UART user interface Modes

MODE	Mode Description
0	Disabled
1	Receive
2	Transmit
3	Receive with resync.

30.7.2 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.

30.7.3 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.

30.7.4 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.

30.7.5 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.

30.7.6 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.

30.7.7 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.

30.7.8 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.

30.7.9 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.

30.7.10 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.

30.7.11 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.

30.8 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 30-4. Module clock name

Module name	Clock name
aWire	**CLK_AW**

Table 30-5. Register Reset Values

Register	Reset Value
VERSION	0x00000210

31. Programming and Debugging

31.1 Overview

The AT32UC3L 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.

31.2 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.

31.2.1 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 31-1.

Table 31-1. 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 31-1. 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

31.2.2 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.

31.2.2.1 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 31-2. SAB Security Measures

Security Measure	Control Location	Description
Secure mode	FLASHCDW SECURE bits set	Allocates a portion of the flash for secure code. This code cannot be read or debugged.
Security bit	FLASHCDW security bit set	Programming and debugging not possible, very restricted access.
User code programming	FLASHCDW 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 31-3. Secure Mode SAB Restrictions

Name	Address Start	Address End	Access
Secure flash area	0x580000000	0x580000000 + (USERROW[15:0] << 10)	Blocked
Secure RAM area	0x500000000	0x500000000 + (USERROW[31:16] << 10)	Blocked
Other accesses	-	-	As normal

Table 31-4. 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 31-5. 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

31.3 On-Chip Debug

Rev: 1.3.0.0

31.3.1 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**

31.3.2 Overview

Debugging on the AT32UC3L016/32/64 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. 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 AT32UC3L016/32/64 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.

31.3.3 I/O Lines Description

The OCD AUX trace port contains a number of pins, as shown in Table 31-6 on page 726. 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 31-6. 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

31.3.4 Product Dependencies

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

31.3.4.1 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.

31.3.4.2 Clocks

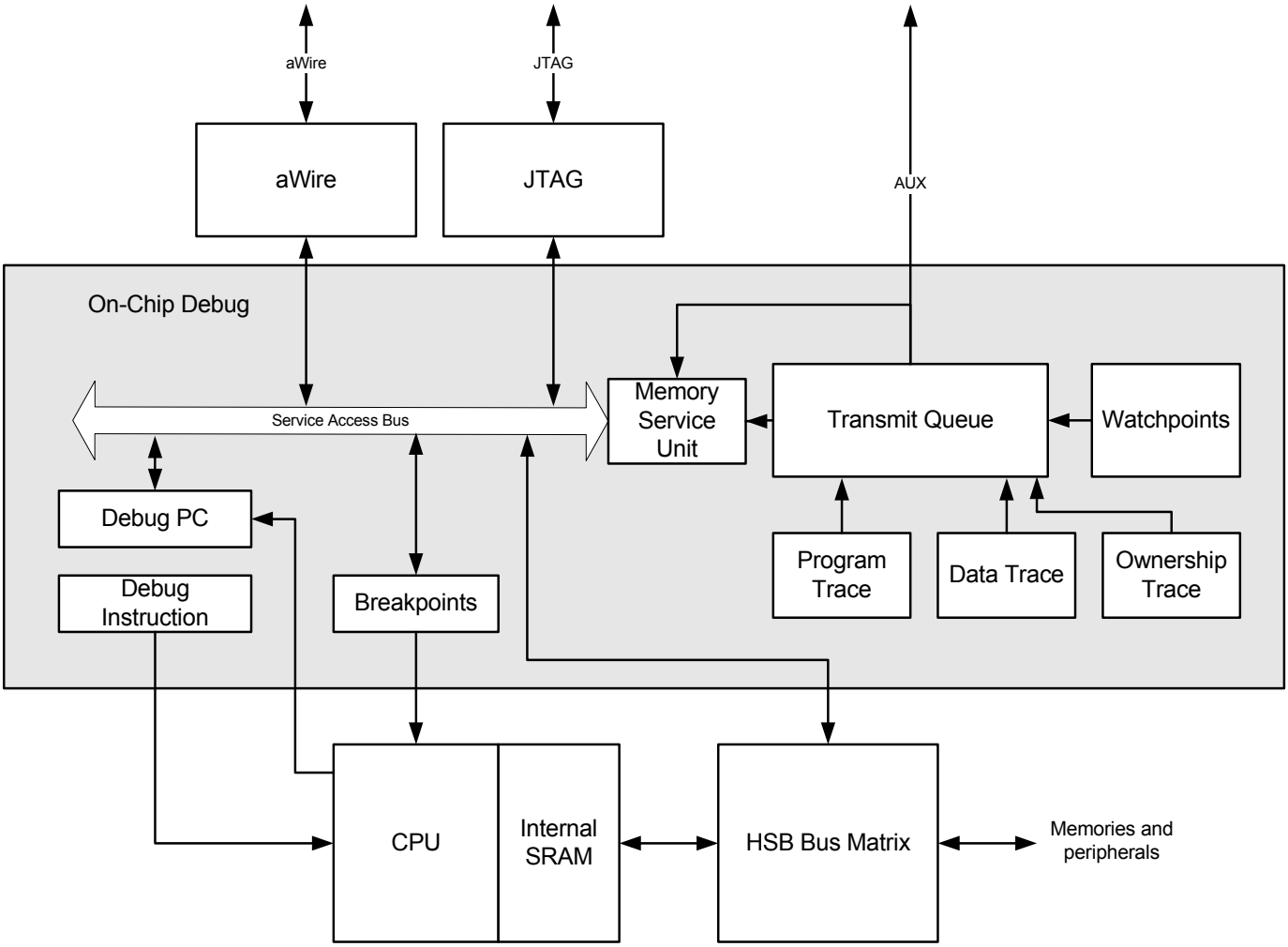
The OCD has a clock (CLK_OCD) running synchronously with the CPU clock. This clock is generated by the Power Manager. The clock is enabled at reset, and can be disabled by writing to the Power Manager.

31.3.4.3 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.

31.3.5 Block Diagram

Figure 31-1. On-Chip Debug Block Diagram



31.3.6 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 31-2. JTAG-based Debugger

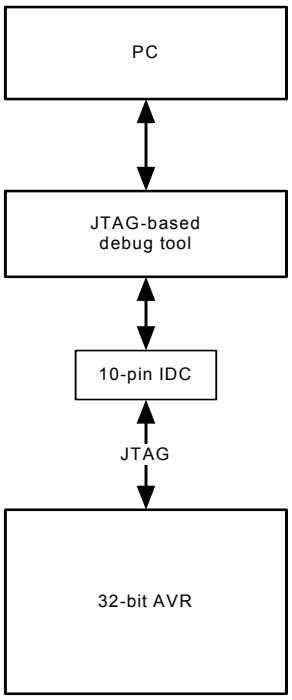
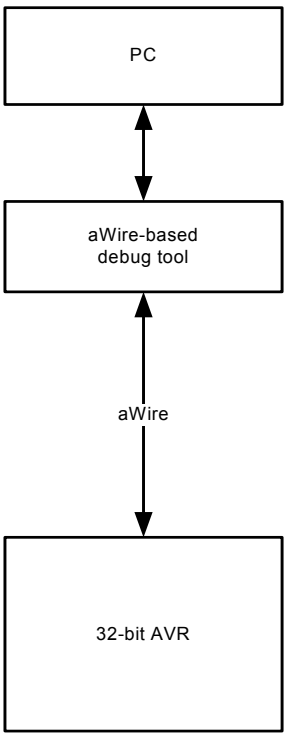


Figure 31-3. aWire-based Debugger



31.3.6.1 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.

31.3.6.2 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.

31.3.6.3 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.

31.3.6.4 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.

31.3.6.5 Program Counter Monitoring

Normally, the CPU would need to be halted for a debugger to examine the current PC value. However, the AT32UC3L016/32/64 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.

31.3.7 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.

31.3.7.1 *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.

31.3.7.2 *NanoTrace*

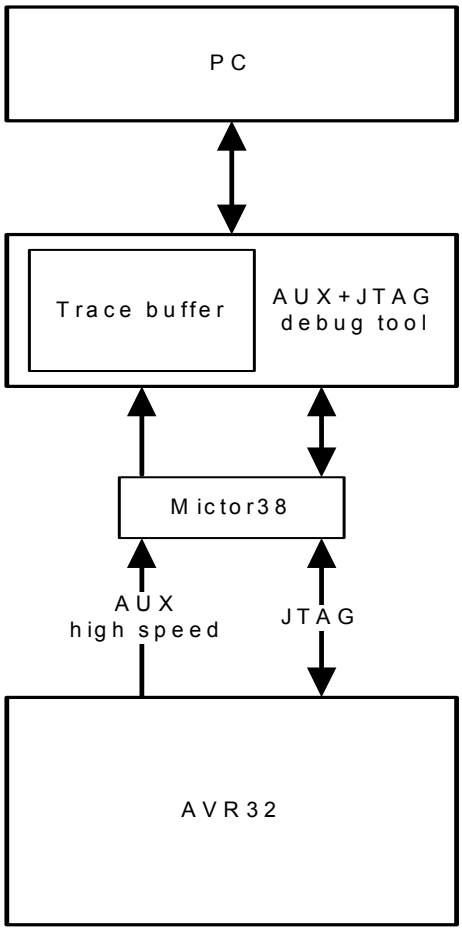
The MSU additionally supports NanoTrace. This is a 32-bit AVR-specific feature, in which trace data is output to memory instead of the AUX port. This allows the trace data to be extracted by the debugger through the SAB, enabling trace features for aWire- or JTAG-based debuggers. The user must write 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.

31.3.8 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 31-4. AUX+JTAG Based Debugger



31.3.8.1 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.

31.3.8.2 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.

31.3.8.3 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 AT32UC3L016/32/64 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.

31.3.8.4 *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.

31.3.8.5 *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.

31.3.8.6 *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.

31.3.8.7 *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 AT32UC3L016/32/64 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.

31.3.9 Module Configuration

The bit mapping of the Peripheral Debug Register (PDBG) is described in Table 31-7. Please refer to the On-Chip Debug chapter in the AVR32UC Technical Reference Manual for details.

Table 31-7. Bit mapping of the Peripheral Debug Register (PDBG)

Bit	Peripheral
0	AST
1	WDT

31.4 JTAG and Boundary-Scan (JTAG)

Rev: 2.2.1.4

31.4.1 Features

- **IEEE1149.1 compliant JTAG Interface**
- **Boundary-Scan Chain for board-level testing**
- **Direct memory access and programming capabilities through JTAG Interface**

31.4.2 Overview

The JTAG Interface offers a four pin programming and debug solution, including boundary-scan support for board-level testing.

Figure 31-5 on page 735 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.

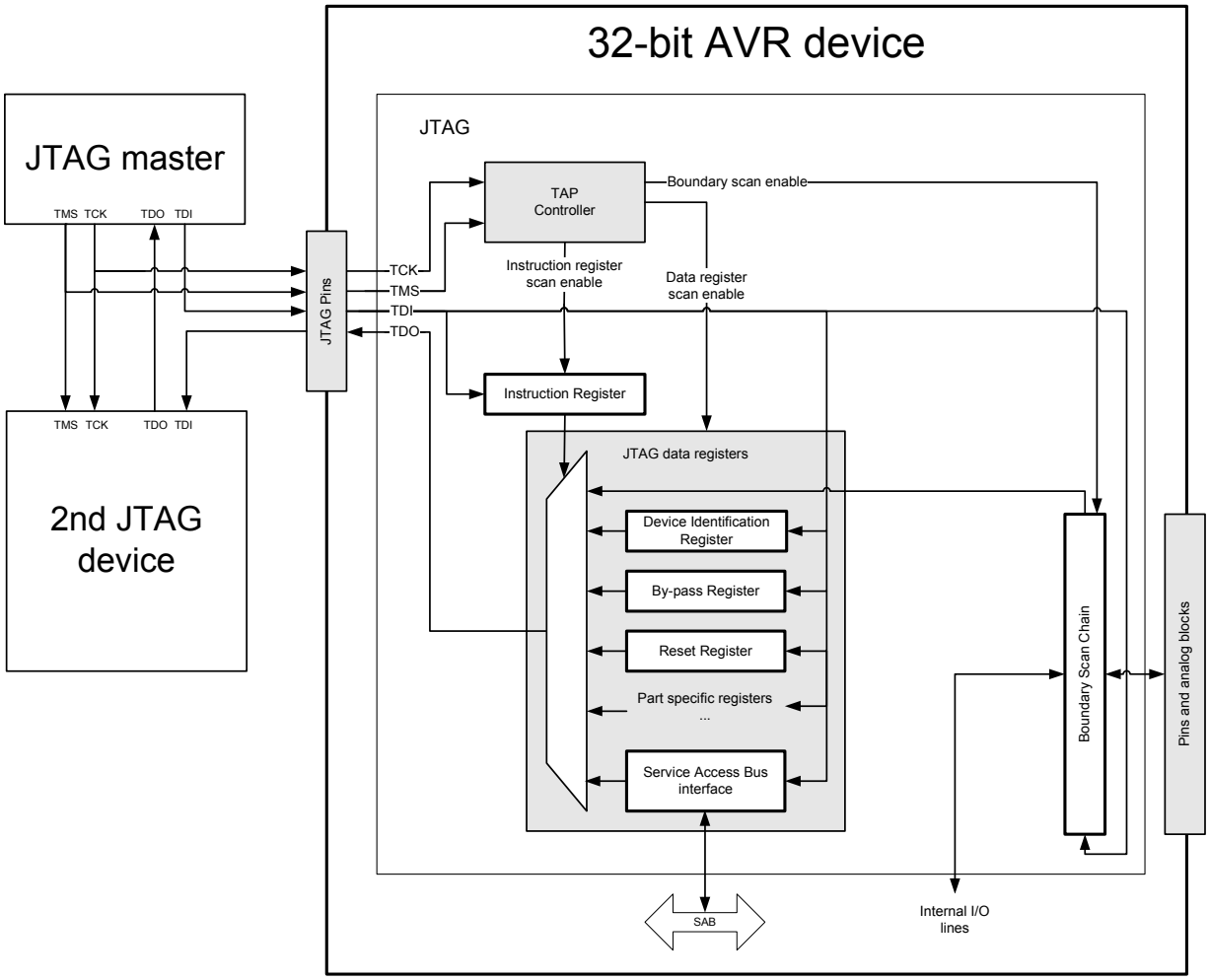
The Instruction Register holds JTAG instructions controlling the behavior of a Data Register. The Device Identification Register, Bypass Register, and the boundary-scan chain are the Data Registers used for board-level testing. The Reset Register can be used to keep the device reset during test or programming.

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 31.4.11.

Section 31.5 lists the supported JTAG instructions, with references to the description in this document.

31.4.3 Block Diagram

Figure 31-5. JTAG and Boundary-Scan Access



31.4.4 I/O Lines Description

Table 31-8. 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	

31.4.5 Product Dependencies

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

31.4.5.1 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 31.4.7.

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.

31.4.5.2 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.

31.4.5.3 Clocks

The JTAG Interface uses the external TCK pin as clock source. This clock must be provided by the JTAG master.

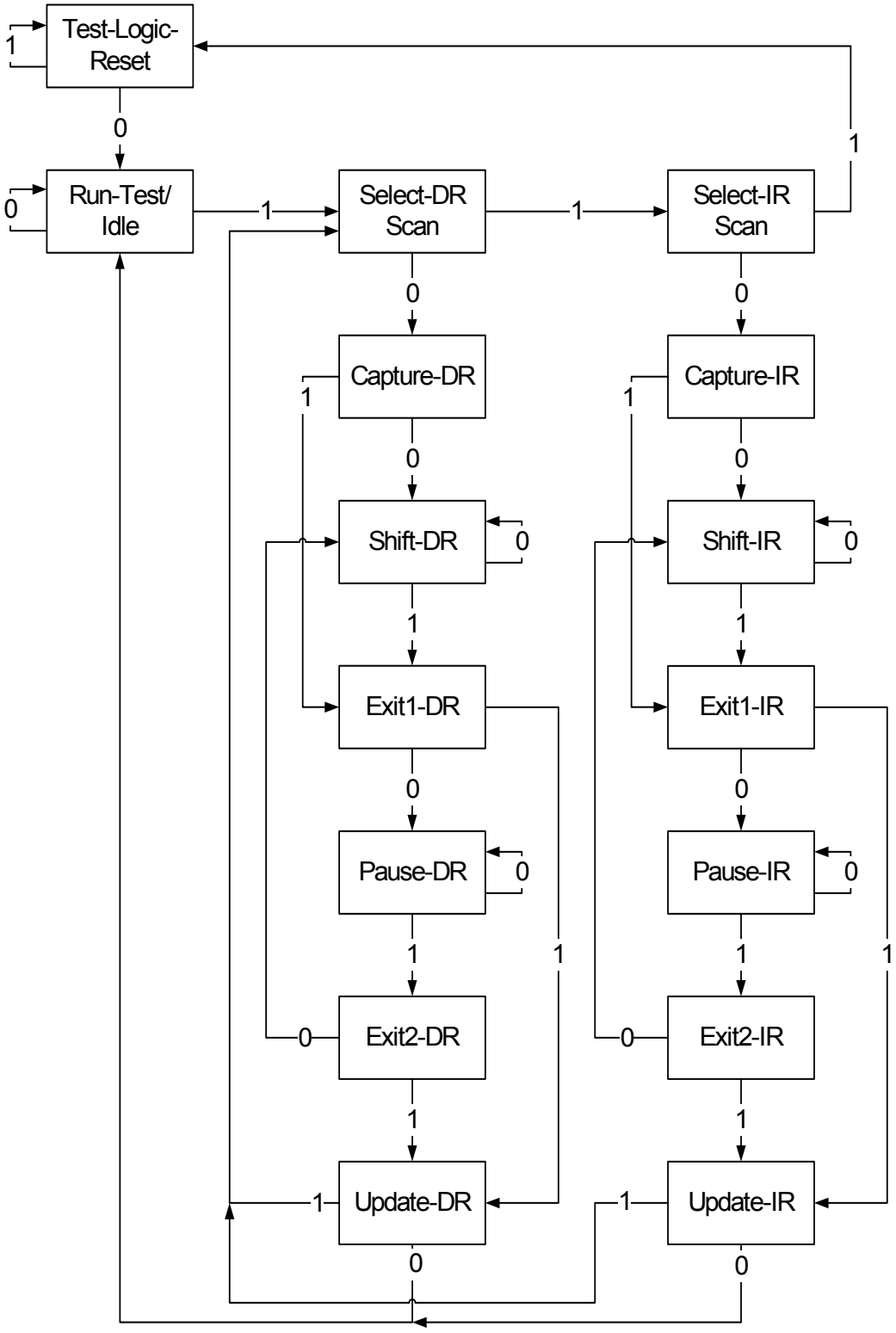
Instructions that use the SAB bus requires the internal main clock to be running.

31.4.6 JTAG Interface

The JTAG Interface is accessed through the dedicated JTAG pins shown in Table 31-8 on page 735. The TMS control line navigates the TAP controller, as shown in Figure 31-6 on page 737. 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 31.5.2, as well as a number of 32-bit AVR-specific private JTAG instructions described in Section 31.5.3. Each instruction selects a specific data register for the Shift-DR path, as described for each instruction.

Figure 31-6. TAP Controller State Diagram



31.4.7 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.

31.4.8 How to disable the module

To disable the JTAG pins the TCK pin must be held high while RESET_N pin is released.

31.4.9 Typical Sequence

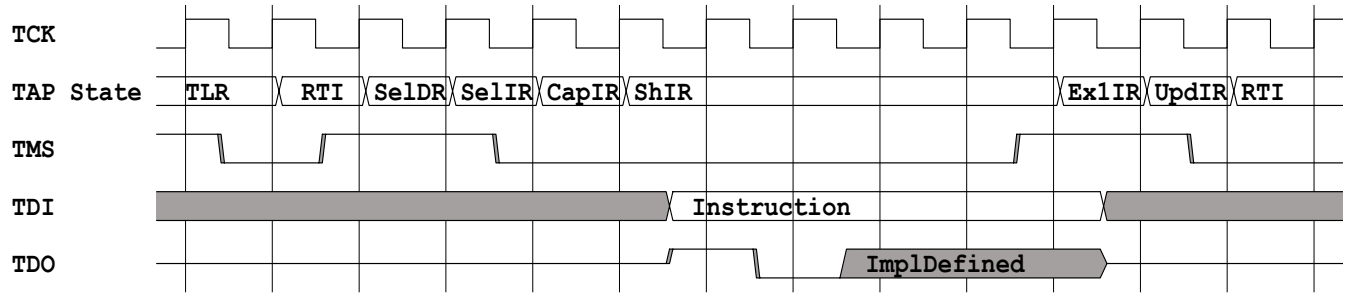
Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG Interface follows.

31.4.9.1 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 31.5 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 31-7. Scanning in JTAG Instruction



31.4.9.2 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.

31.4.10 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.

31.4.11 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.

31.4.11.1 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.

31.4.11.2 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.

31.4.11.3 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.

31.4.11.4 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.

31.4.11.5 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.

31.4.11.6 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 31.5.1.

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.

31.5 JTAG Instruction Summary

The implemented JTAG instructions in the 32-bit AVR are shown in the table below.

Table 31-9. 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

31.5.1 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.

31.5.1.1 Notation

Table 31-11 on page 743 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 31-10.

Table 31-10. 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 31.4.11.4 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 31.4.11.5 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 31-11. 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 31-11. 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**

31.5.2 Public JTAG Instructions

The JTAG standard defines a number of public JTAG instructions. These instructions are described in the sections below.

31.5.2.1 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 31-12. 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

31.5.2.2 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 31-13. 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.

31.5.2.3 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 31-14. 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.

31.5.2.4 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 31-15. 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.

31.5.2.5 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 31-16. CLAMP Details

Instructions	Details
IR input value	**00110** (0x06)
IR output value	p0001
DR Size	1
DR input value	x
DR output value	x

31.5.2.6 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 31-17. BYPASS Details

Instructions	Details
IR input value	**11111** (0x1F)
IR output value	p0001
DR Size	1
DR input value	x
DR output value	x

31.5.3 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.

31.5.3.1 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 31-18. 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**

31.5.3.2 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 31-19. 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**

31.5.3.3 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 31-20.

Table 31-20. 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 31-21. 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 31-21. 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**

31.5.3.4 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 31-22. 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 31-22. 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**

31.5.3.5 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 31-23. 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 31-23. 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.

31.5.3.6 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 31-24. 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

31.5.3.7 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 31-25. 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

31.5.3.8 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 31-26. AVR_RESET Details

Instructions	Details
IR input value	**01100** (0x0C)
IR output value	p0001

Table 31-26. AVR_RESET Details (Continued)

Instructions	Details
DR Size	Device specific.
DR input value	Device specific.
DR output value	Device specific.

31.5.3.9 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 31-27. 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

31.5.3.10 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 halts the CPU. Setting this bit to zero releases the CPU to run normally. The value shifted out from the data register is one if the CPU is halted. Before releasing the halt command the CPU needs to be reset to ensure that it will start at the reset startup address.

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 31-28. HALT Details

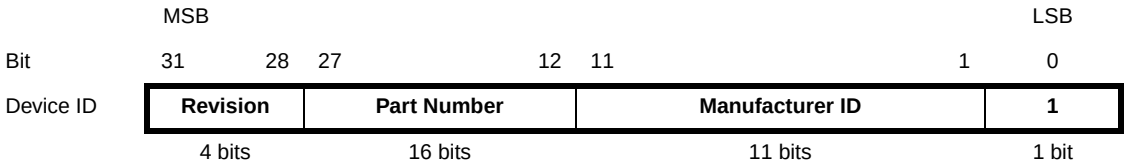
Instructions	Details
IR input value	**11100** (0x1C)
IR output value	p0001
DR Size	1 bit
DR input value	d
DR output value	d

31.5.4 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.

31.5.4.1 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

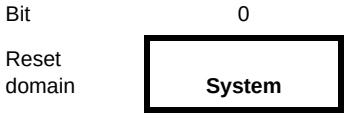
The different device configurations have different JTAG ID codes, as shown in Table 31-29. Note that if the flash controller is statically reset, the ID code will be undefined.

Table 31-29. Device and JTAG ID

Device Name	JTAG ID Code (R is the revision number)
AT32UC3L064	0xR203003F
AT32UC3L032	0xR203403F
AT32UC3L016	0xR203803F

31.5.4.2 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.

31.5.4.3 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 BDSL (Boundary Scan Description Language) file available at the Atmel web site.

31.6 aWire Debug Interface (AW)

Rev.: 2.1.0.1

31.6.1 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.**

31.6.2 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 31-8 on page 760 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.

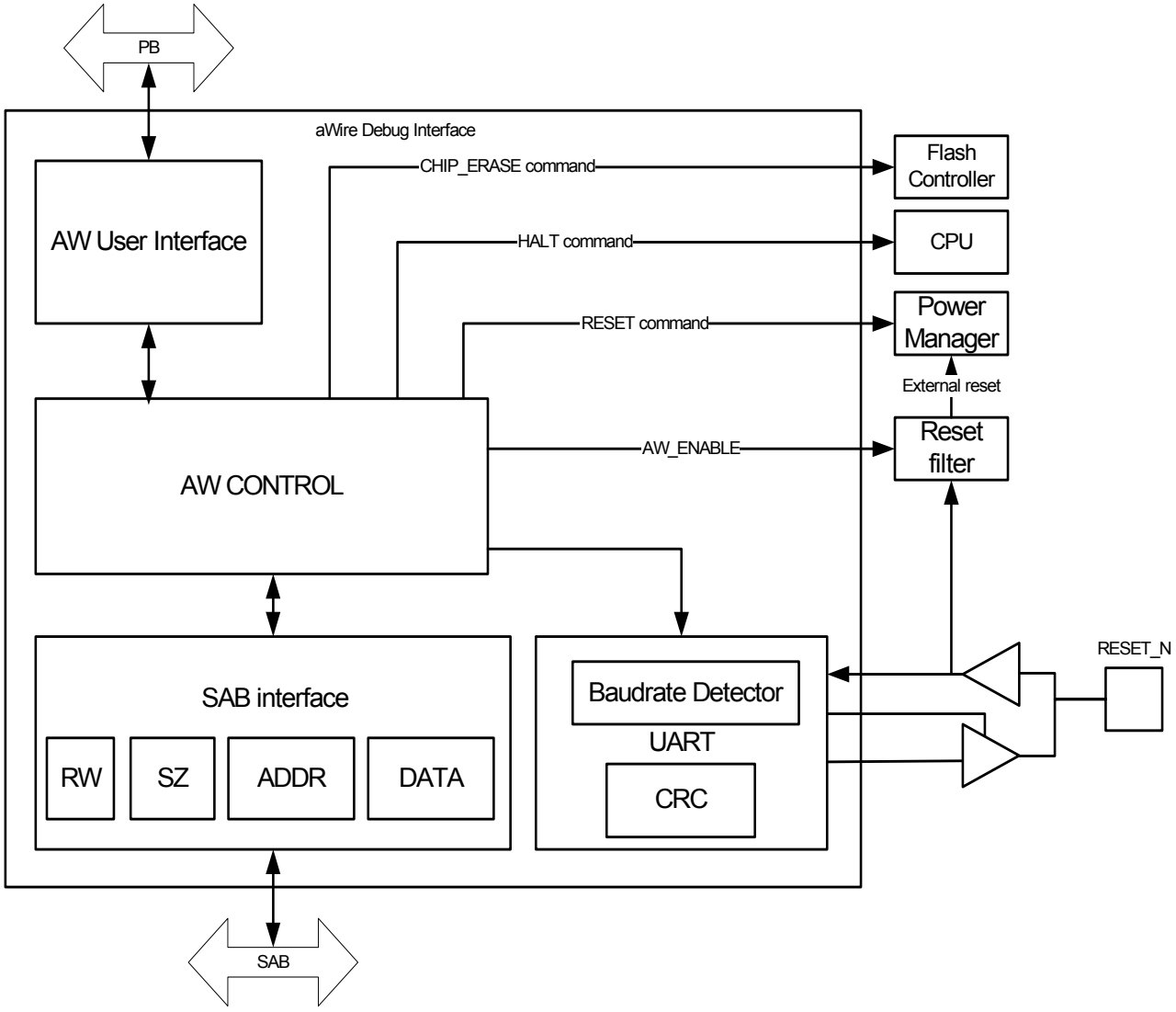
The Service Access Bus (SAB) interface contains address and data registers for the Service Access Bus, which gives access to On-Chip Debug, programming, and other functions in the device. The SAB offers several modes of access to the address and data registers, as discussed in Section 31.6.6.8.

Section 31.6.7 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.

31.6.3 Block Diagram

Figure 31-8. aWire Debug Interface Block Diagram



31.6.4 I/O Lines Description

Table 31-30. 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

31.6.5 Product Dependencies

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

31.6.5.1 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.

31.6.5.2 Power Management

When debugging through AW the system clocks are automatically turned on to allow debugging in sleep modes.

31.6.5.3 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.

31.6.5.4 External Components

The AW needs an external pullup on the RESET_N pin to ensure that the pin is pulled up when the bus is not driven.

31.6.6 Functional Description

31.6.6.1 aWire Communication Protocol

The AW is accessed through the RESET_N pin shown in Table 31-30 on page 760. 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 31-31. 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

Table 31-31. aWire Packet Format

Field	Number of bytes	Description	Comment	Optional
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

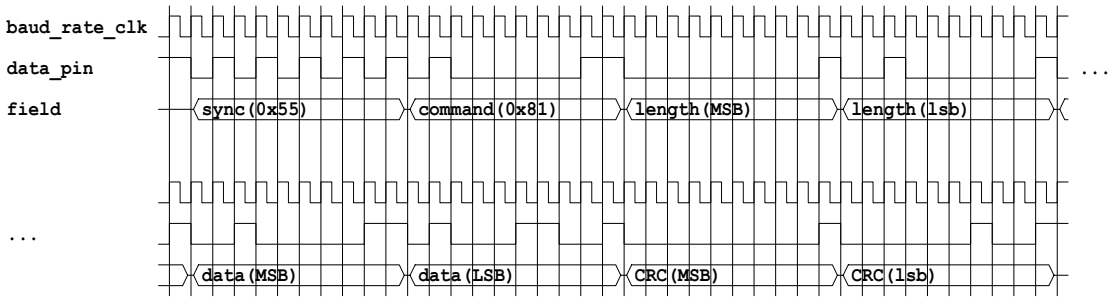
CRC calculation

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.

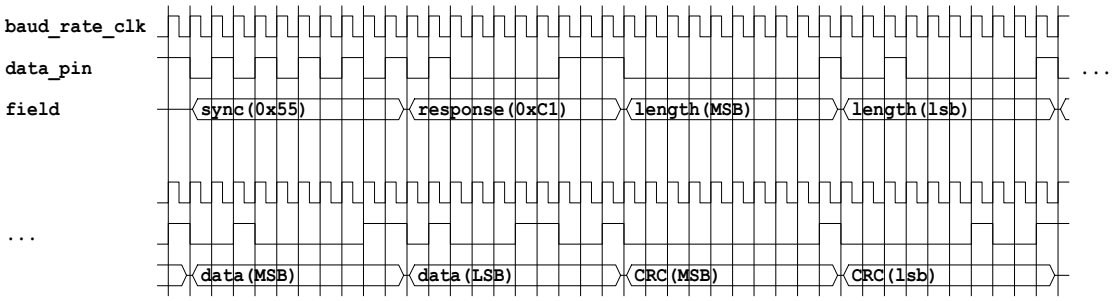
Figure 31-9. Example Command



Example response

Below is an example response from the slave with additional data.

Figure 31-10. 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.

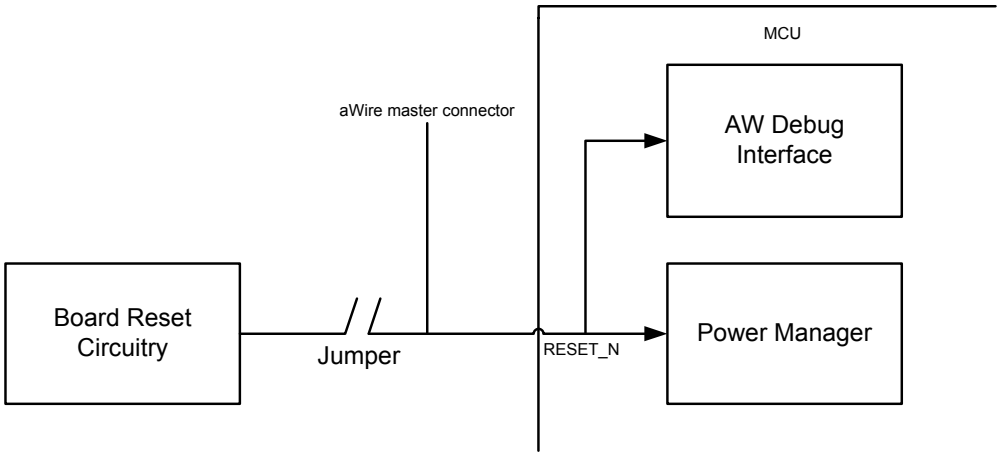
The AW will release the line immediately after the stop character has been transmitted.

During the direction change there can be a period when the line is not driven. An external pullup has to be added to RESET_N to keep the signal stable when neither master or slave is actively driving the line.

31.6.6.2 *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 31-11. Reset Circuitry and aWire.



31.6.6.3 *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.

31.6.6.4 *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.

31.6.6.5 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.

31.6.6.6 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.

31.6.6.7 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:

$$f_{brmax} = \frac{f_{aw}}{4}$$

31.6.6.8 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 31.6.8.5 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.

31.6.6.9 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.

31.6.7 aWire Command Summary

The implemented aWire commands are shown in the table below. The responses from the AW are listed in Section 31.6.8.

Table 31-32. 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.
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 31-33. 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.

31.6.7.1 AYA

This command asks the AW: “Are you alive”, where the AW should respond with an acknowledge.

Table 31-34. AYA Details

Command	Details
Command value	0x01
Additional data	N/A
Possible responses	0x40: ACK (Section 31.6.8.1) 0x41: NACK (Section 31.6.8.2)

31.6.7.2 JTAG_ID

This command instructs the AW to output the JTAG idcode in the following response.

Table 31-35. JTAG_ID Details

Command	Details
Command value	0x02
Additional data	N/A
Possible responses	0xC0: IDCODE (Section 31.6.8.3) 0x41: NACK (Section 31.6.8.2)

31.6.7.3 STATUS_REQUEST

Asks the AW for a status message.

Table 31-36. STATUS_REQUEST Details

Command	Details
Command value	0x03
Additional data	N/A
Possible responses	0xC4: STATUS_INFO (Section 31.6.8.7) 0x41: NACK (Section 31.6.8.2)

31.6.7.4 TUNE

Asks the AW for the current baud rate counter value.

Table 31-37. TUNE Details

Command	Details
Command value	0x04
Additional data	N/A
Possible responses	0xC3: BAUD_RATE (Section 31.6.8.6) 0x41: NACK (Section 31.6.8.2)

31.6.7.5 MEMORY_SPEED_REQUEST

Asks the AW for the relative speed between the aWire clock (RC120M) and the SAB interface.

Table 31-38. MEMORY_SPEED_REQUEST Details

Command	Details
Command value	0x05
Additional data	N/A
Possible responses	0xC5: MEMORY_SPEED (Section 31.6.8.8) 0x41: NACK (Section 31.6.8.2)

31.6.7.6 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 31-39. CHIP_ERASE Details

Command	Details
Command value	0x06
Additional data	N/A
Possible responses	0x40: ACK (Section 31.6.8.1) 0x41: NACK (Section 31.6.8.2)

31.6.7.7 DISABLE

Disables the AW. The AW will respond with an ACK response and then disable itself.

Table 31-40. DISABLE Details

Command	Details
Command value	0x07
Additional data	N/A
Possible responses	0x40: ACK (Section 31.6.8.1) 0x41: NACK (Section 31.6.8.2)

31.6.7.8 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 31-41. DISABLE Details

Command	Details
Command value	0x07
Additional data	N/A
Possible responses	0x40: ACK (Section 31.6.8.1) 0x41: NACK (Section 31.6.8.2)

31.6.7.9 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 31-42. 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 31-43. MEMORY_WRITE Details

Command	Details
Command value	0x80
Additional data	Size, Address and Data
Possible responses	0xC2: MEMORY_READWRITE_STATUS (Section 31.6.8.5) 0x41: NACK (Section 31.6.8.2)

31.6.7.10 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 31-44. 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 31-45. MEMORY_READ Details

Command	Details
Command value	0x81
Additional data	Size, Address and Length
Possible responses	0xC1: MEMDATA (Section 31.6.8.4) 0xC2: MEMORY_READWRITE_STATUS (Section 31.6.8.5) 0x41: NACK (Section 31.6.8.2)

31.6.7.11 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 31-46. 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 31.6.8.1) 0x41: NACK (Section 31.6.8.2)

31.6.7.12 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 31-47. 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 31.6.8.1) 0x41: NACK (Section 31.6.8.2)

31.6.7.13 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 31-48. SET_GUARD_TIME Details

Command	Details
Command value	0x84
Additional data	Guard time
Possible responses	0x40: ACK (Section 31.6.8.1) 0x41: NACK (Section 31.6.8.2)

31.6.8 aWire Response Summary

The implemented aWire responses are shown in the table below.

Table 31-49. 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.

31.6.8.1 ACK

The AW has received the command successfully and performed the operation.

Table 31-50. ACK Details

Response	Details
Response value	0x40
Additional data	N/A

31.6.8.2 NACK

The AW has received the command, but got a CRC mismatch.

Table 31-51. NACK Details

Response	Details
Response value	0x41
Additional data	N/A

31.6.8.3 IDCODE

The JTAG idcode for this device.

Table 31-52. IDCODE Details

Response	Details
Response value	0xC0
Additional data	JTAG idcode

31.6.8.4 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 31-53. MEMDATA Status Byte

status byte	Description
0x00	Read successful
0x01	SAB busy
0x02	Bus error (wrong address)
Other	Reserved

Table 31-54. MEMDATA Details

Response	Details
Response value	0xC1
Additional data	Data read, status byte, and byte count (2 bytes)

31.6.8.5 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 31-55. MEMORY_READWRITE_STATUS Status Byte

status byte	Description
0x00	Write successful
0x01	SAB busy
0x02	Bus error (wrong address)
Other	Reserved

Table 31-56. MEMORY_READWRITE_STATUS Details

Response	Details
Response value	0xC2
Additional data	Status byte and byte count (2 bytes)

31.6.8.6 BAUD_RATE

The current baud rate in the AW. See Section 31.6.6.7 for more details.

Table 31-57. BAUD_RATE Details

Response	Details
Response value	0xC3
Additional data	Baud rate

31.6.8.7 STATUS_INFO

A status message from AW.

Table 31-58. 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 31-59. STATUS_INFO Details

Response	Details
Response value	0xC4
Additional data	2 status bytes

31.6.8.8 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 31-60. MEMORY_SPEED Details

Response	Details
Response value	0xC5
Additional data	Clock cycle count (MS)

31.6.9 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.

32. Electrical Characteristics

32.1 Disclaimer

All values in this chapter are preliminary and subject to change without further notice.

32.2 Absolute Maximum Ratings*

Table 32-1. Absolute Maximum Ratings

Operating temperature..... -40°C to +85°C
Storage temperature..... -60°C to +150°C
Voltage on input pins (except for 5V pins) with respect to ground .....-0.3V to V_{VDD}[2]+0.3V
Voltage on 5V tolerant[1] pins with respect to ground .....-0.3V to 5.5V
Total DC output current on all I/O pins - VDDIO 120mA
Total DC output current on all I/O pins - VDDIN 36mA
Maximum operating voltage VDDCORE..... 1.98V
Maximum operating voltage VDDIO, VDDIN 3.6V

*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. 5V tolerant pins, see Section 3.2 "Peripheral Multiplexing on I/O lines" on page 9

2. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pad. Refer to Section 3.2 on page 9 for details.

32.3 Supply Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified and are certified for a junction temperature up to T_J = 100°C.

Table 32-2. Supply Characteristics[1]

Symbol	Parameter	Voltage		
		Min	Max	Unit
V_{VDDIO}	DC supply peripheral I/Os	1.62	3.6	V
V_{VDDIN}	DC supply peripheral I/Os, 1.8V single supply mode	1.62	1.98	V
	DC supply peripheral I/Os and internal regulator, 3.3V single supply mode	1.98	3.6	V
$V_{VDDCORE}$	DC supply core	1.62	1.98	V
V_{VDDANA}	Analog supply voltage	1.62	1.98	V
$V_{ADVREFP}$	Analog reference voltage	1.62	V_{VDDANA}	V

Note: 1. VDDANA = VDDCORE

Table 32-3. Supply Rise Rates and Order

Symbol	Parameter	Rise Rate			
		Min	Max	Unit	Comment
V_{VDDIO}	DC supply peripheral I/Os	0	2.5	V/µs	
V_{VDDIN}	DC supply peripheral I/Os and internal regulator	0.002(1)	2.5	V/µs	
$V_{VDDCORE}$	DC supply core	0	2.5	V/µs	Rise before or at the same time as VDDIO
V_{VDDANA}	Analog supply voltage	0	2.5	V/µs	Rise together with VDDCORE

Note: 1. Slower rise time requires external power-on reset circuit.

32.4 Maximum Clock Frequencies

These parameters are given in the following conditions:

- $V_{VDDCORE}$ = 1.62 to 1.98V
- Temperature = -40°C to 85°C

Table 32-4. Clock Frequencies

Symbol	Parameter	Conditions	Min	Max	Units
f_{CPU}	CPU clock frequency			50	MHz
f_{PBA}	PBA clock frequency			50	MHz
f_{PBB}	PBB clock frequency			50	MHz
f_{GCLK0}	GCLK0 clock frequency			150	MHz
f_{GCLK1}	GCLK1 clock frequency			150	MHz
f_{GCLK2}	GCLK2 clock frequency			80	MHz
f_{GCLK3}	GCLK3 clock frequency			110	MHz
f_{GCLK4}	GCLK4 clock frequency			110	MHz
f_{GCLK5}	GCLK5 clock frequency			80	MHz

32.5 Power Consumption

The values in Table 32-5 are measured values of power consumption under the following conditions, except where noted:

- Operating conditions internal core supply (Figure 32-1) - this is the default configuration
  - V_{VDDIN} = 3.0V
  - $V_{VDDCORE}$ = 1.62V
  - T_A = 25°C
- Operating conditions external core supply (Figure 32-2) - used only when noted
  - V_{VDDIN} = $V_{VDDCORE}$ = 1.8V
  - T_A = 25°C
- Oscillators

  - OSC0 (crystal oscillator) stopped
  - OSC32K (32KHz crystal oscillator) running with external 32KHz crystal
  - DFLL running at 50MHz with OSC32K as reference
- Clocks
  - DFLL used as main clock source
  - CPU, HSB, and PBB clocks undivided
  - PBA clock divided by 4
  - The following peripheral clocks running
    - PM, SCIF, AST, FLASHCDW, PBA bridge
  - All other peripheral clocks stopped
- I/Os are inactive with internal pull-up
- Flash enabled in high speed mode
- POR33 disabled

Table 32-5. Power Consumption for Different Modes

Mode	Conditions	Measured on	Consumption Typ	Unit
Active	-CPU running a recursive Fibonacci algorithm	Amp0	260	µA/MHz
	-CPU running a division algorithm		165	
Idle			92	
Frozen			58	
Standby			47	
Stop			37	µA
DeepStop			23	
Static	-OSC32K and AST stopped -Internal core supply		10	
	-OSC32K running -AST running at 1KHz -External core supply (Figure 32-2)		5.3	
	-OSC32K and AST stopped -External core supply (Figure 32-2)		4.7	
Shutdown	-OSC32K running -AST running at 1KHz		600	nA
	-AST and OSC32K stopped		9	

Figure 32-1. Measurement Schematic, Internal Core Supply

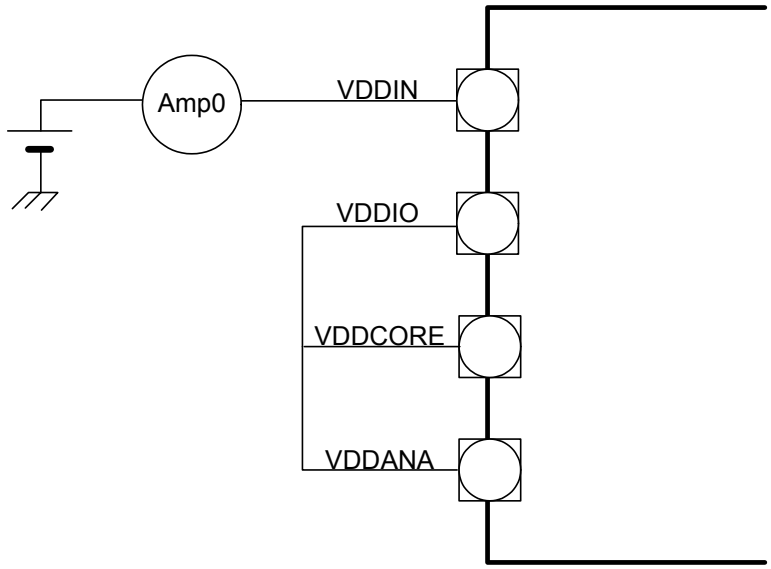
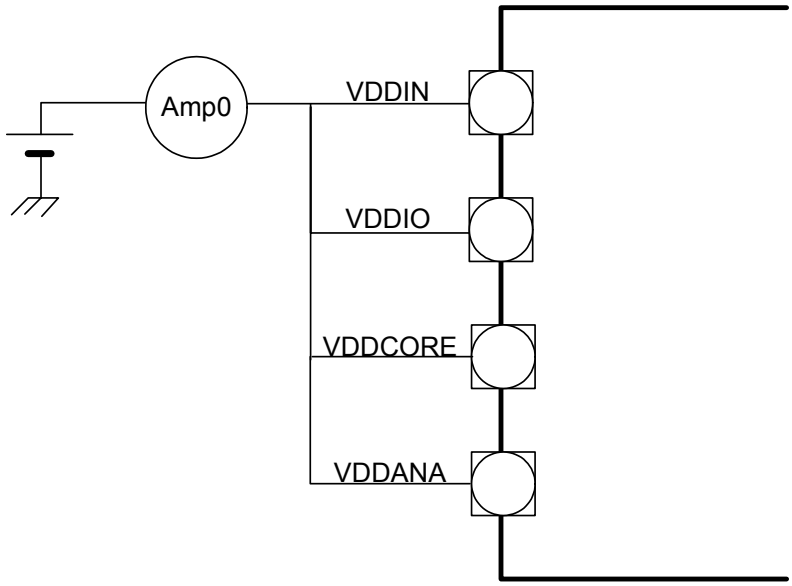


Figure 32-2. Measurement Schematic, External Core Supply



32.6 I/O Pad Characteristics

Table 32-6. Normal I/O Pad Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance		75	100	145	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3.0V	-0.3		0.3*V_{VDD}	V
		V_{VDD} = 1.62V	-0.3		0.3*V_{VDD}	
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V	0.7*V_{VDD}		V_{VDD} + 0.3	V
		V_{VDD} = 1.98V	0.7*V_{VDD}		V_{VDD} + 0.3	

Table 32-6. Normal I/O Pad Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{OL}	Output low-level voltage	V_{VDD} = 3.0V, I_{OL} = 3mA			0.4	V
		V_{VDD} = 1.62 V, I_{OL} = 2mA			0.4	
V_{OH}	Output high-level voltage	V_{VDD} = 3.0V, I_{OH} = 3mA	V_{VDD} - 0.4			V
		V_{VDD} = 1.62 V, I_{OH} = 2mA	V_{VDD} - 0.4			
I_{LEAK}	Input leakage current	Pull-up resistors disabled			1	µA

Notes: 1. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pad. Refer to Section 3.2 on page 9 for details.

Table 32-7. High-drive I/O Pad Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance	PA06	30	50	110	kOhm
		PA02, PB01, RESET	75	100	145	
		PA08, PA09	10	20	45	
V_{IL}	Input low-level voltage	V_{VDD} = 3.0V	-0.3		0.3*V_{VDD}	V
		V_{VDD} = 1.62V	-0.3		0.3*V_{VDD}	
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V	0.7*V_{VDD}		V_{VDD} + 0.3	V
		V_{VDD} = 1.98V	0.7*V_{VDD}		V_{VDD} + 0.3	
V_{OL}	Output low-level voltage	V_{VDD} = 3.0V, I_{OL} = 6mA			0.4	V
		V_{VDD} = 1.62 V, I_{OL} = 4mA			0.4	
V_{OH}	Output high-level voltage	V_{VDD} = 3.0V, I_{OH} = 6mA	V_{VDD}-0.4			V
		V_{VDD} = 1.62 V, I_{OH} = 4mA	V_{VDD}-0.4			
I_{LEAK}	Input leakage current	Pull-up resistors disabled			1	µA

Notes: 1. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pad. Refer to Section 3.2 on page 9 for details.

Table 32-8. 5V Tolerant Normal I/O Pad Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance		75	100	145	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3.0V	-0.3		0.3*V_{VDD}	V
		V_{VDD} = 1.62V	-0.3		0.3*V_{VDD}	
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V	0.7*V_{VDD}		5.5	V
		V_{VDD} = 1.98V	0.7*V_{VDD}		5.5	
V_{OL}	Output low-level voltage	V_{VDD} = 3.0V, I_{OL} = 3mA			0.4	V
		V_{VDD} = 1.62 V, I_{OL} = 2mA			0.4	
V_{OH}	Output high-level voltage	V_{VDD} = 3.0V, I_{OH} = 3mA	V_{VDD}-0.4			V
		V_{VDD} = 1.62 V, I_{OH} = 2mA	V_{VDD}-0.4			
I_{LEAK}	Input leakage current	5.5V, pull-up resistors disabled			1	µA

Notes: 1. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pad. Refer to Section 3.2 on page 9 for details.

Table 32-9. 5V Tolerant High-drive I/O Pad Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance		30	50	110	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3.0V	-0.3		0.3*V_{VDD}	V
		V_{VDD} = 1.62V	-0.3		0.3*V_{VDD}	
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V	0.7*V_{VDD}		5.5	V
		V_{VDD} = 1.98V	0.7*V_{VDD}		5.5	
V_{OL}	Output low-level voltage	V_{VDD} = 3.0V, I_{OL} = 6mA			0.4	V
		V_{VDD} = 1.62 V, I_{OL} = 4mA			0.4	
V_{OH}	Output high-level voltage	V_{VDD} = 3.0V, I_{OH} = 6mA	V_{VDD}-0.4			V
		V_{VDD} = 1.62 V, I_{OH} = 4mA	V_{VDD}-0.4			
I_{LEAK}	Input leakage current	5.5V, pull-up resistors disabled			1	µA

Notes: 1. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pad. Refer to Section 3.2 on page 9 for details.

Table 32-10. TWI Pad Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance		25	35	50	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3.0V	-0.3		0.3*V_{VDD}	V
		V_{VDD} = 1.62V	-0.3		0.3*V_{VDD}	
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V	0.7*V_{VDD}		V_{VDD} + 0.3	V
		V_{VDD} = 1.98V	0.7*V_{VDD}		V_{VDD} + 0.3	
V_{OL}	Output low-level voltage	I_{OL} = 3mA			0.4	V
I_{LEAK}	Input leakage current	Pull-up resistors disabled			1	µA
I_{IL}	Input low leakage				1	µA
I_{IH}	Input high leakage				1	µA
f_{MAX}	Max frequency	Cbus = 400pF, V_{VDD} > 2.0V	400			kHz

Notes: 1. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pad. Refer to Section 3.2 on page 9 for details.

32.7 Oscillator Characteristics

32.7.1 Oscillator 0 (OSC0) Characteristics

32.7.1.1 Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN.

Table 32-11. Digital Clock Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{CPXIN}	XIN clock frequency				50	MHz
t_{CPXIN}	XIN clock duty cycle		40		60	%

32.7.1.2 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 32-3. 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:

$$C_{LEXT} = 2(C_L - C_i) - C_{PCB}$$

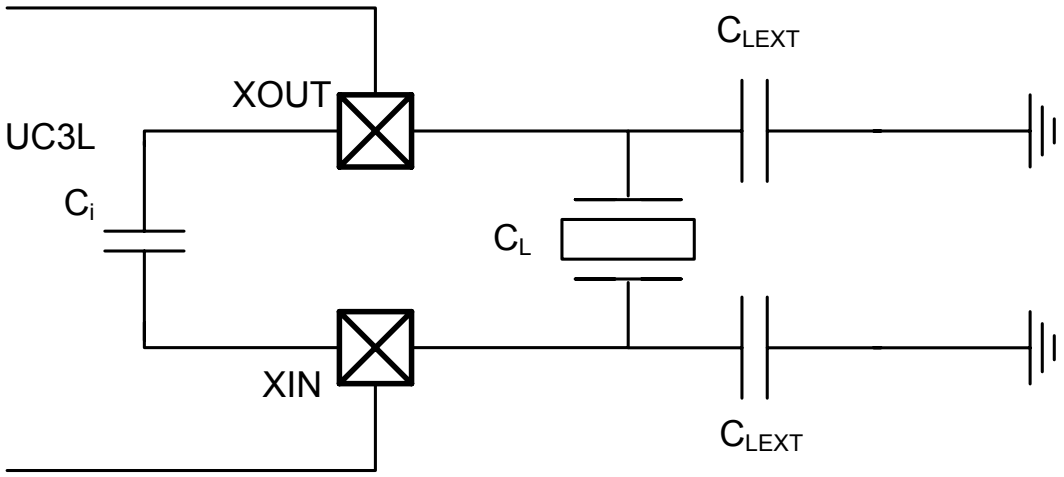
where C_{PCB} is the capacitance of the PCB.

Table 32-12. Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal oscillator frequency		3		16	MHz
C_L	Crystal load capacitance		6		18	pF
C_i	Internal equivalent load capacitance			2		pF
$t_{STARTUP}$	Startup time	SCIF.OSCCTRL.GAIN = 2(1)		30 000(2)		cycles

Notes: 1. Please refer to the SCIF chapter for details.
2. Nominal crystal cycles.

Figure 32-3. Oscillator Connection



32.7.2 32KHz Crystal Oscillator (OSC32K) Characteristics

Figure 32-3 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 32-13. 32 KHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal oscillator frequency			32 768		Hz
t_{ST}	Startup time	R_S = 60kOhm, C_L = 9pF		30 000(1)		cycles
C_L	Crystal load capacitance		6		12.5	pF

Table 32-13. 32 KHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
C_i	Internal equicalent load capacitance			2		pF
I_{OSC}	Current consumption			0.9		µA
R_S	Equivalent series resistance	32 768 Hz	35		85	kOhm

Note: 1. Nominal crystal cycles.

32.7.3 Digital Frequency Locked Loop (DFLL) Characteristics

Table 32-14. Digital Frequency Locked Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency		40		150	MHz
f_{REF}	Reference frequency		8		150	kHz
	FINE resolution	FINE>100, all COARSE values		0.25		%
	Accuracy	Fine lock, f_{REF}=32 kHz, SSG disabled		0.1	0.5	%
		Accurate lock, f_{REF}=32 kHz, dither clk RCSYS/2, SG disabled		0.06	0.5	
		Fine lock, f_{REF}=8-150 kHz, SSG disabled		0.2	1	
		Accurate lock, f_{REF}=8-150 kHz, dither clk RCSYS/2, SSG disabled		0.1	1	
	Power consumption			22		µA/MHz
$t_{STARTUP}$	Startup time	Within 90% of final values			100	µs
t_{LOCK}	Lock time	f_{REF} = 32 kHz, fine lock, SSG disabled		600		µs
		f_{REF} = 32 kHz, accurate lock, dithering clock = RCSYS/2, SSG disabled		1100		

Note: 1. Spread Spectrum Generator (SSG) is disabled by writing a zero to the EN bit in the SCIF.DFLL0SSG register.

32.7.4 120MHz RC Oscillator (RC120M) Characteristics

Table 32-15. Internal 120MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency	T = 25°C, $V_{VDDCORE}$ = 1.8V	88	120	152	MHz
	Temperature drift			+/-5		%
Duty	Duty cycle		40	50	60	%

32.7.5 32kHz RC Oscillator (RC32K) Characteristics

Table 32-16. 32kHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency	T = 25°C, V_{VDDIO} = 3.3V	20	32	44	kHz

32.7.6 System RC Oscillator (RCSYS) Characteristics

Table 32-17. System RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency	Calibrated at 85°C		115		kHz

32.8 Flash Characteristics

Table 32-18 gives the device maximum operating frequency depending on the number of flash wait states and the flash read mode. The FSW bit in the FLASHCDW FSR register controls the number of wait states used when accessing the flash memory.

Table 32-18. Maximum Operating Frequency

Flash Wait States	Read Mode	Maximum Operating Frequency
1	High speed read mode	50MHz
0		25MHz
1	Normal read mode	30MHz
0		15MHz

Table 32-19. Flash Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_{FPP}	Page programming time	f_{CLK_HSB}= 50MHz		5		ms
T_{FPE}	Page erase time			5		
T_{FFP}	Fuse programming time			1		
T_{FEA}	Full chip erase time (EA)			5		
T_{FCE}	JTAG chip erase time (CHIP_ERASE)	f_{CLK_HSB}= 115kHz		300		

Table 32-20. 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)		10k			cycles
t_{RET}	Data retention		15			years

32.9 Analog Characteristics

32.9.1 Voltage Regulator Characteristics

32.9.1.1 Electrical Characteristics

Table 32-21. Electrical Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{VDDIN}	Input voltage range		1.98	3.3	3.6	V
$V_{VDDCORE}$	Output voltage	V_{VDDIN} > 1.98V		1.8		V
	Output voltage accuracy	I_{OUT} = 0.1mA to 60mA, V_{VDDIN}>2.2V		2		%
		I_{OUT} = 0.1mA to 60mA, V_{VDDIN}=1.98V to 2.2V		4		
I_{OUT}	DC output current	Normal mode			60	mA
		Low power mode			1	mA
I_{SCR}	Static current of internal regulator	Normal mode		20		µA
		Low power mode		6		µA

32.9.1.2 Decoupling Requirements

Table 32-22. Decoupling Requirements

Symbol	Parameter	Condition	Typ	Techno.	Units
C_{IN1}	Input regulator capacitor 1		33		nF
C_{IN2}	Input regulator capacitor 2		100		nF
C_{IN3}	Input regulator capacitor 3		10		µF
C_{OUT1}	Output regulator capacitor 1		100		nF
C_{OUT2}	Output regulator capacitor 2		2.2	Tantalum 0.5<ESR<10	µF

Note: 1. Refer to Section 6.1.2 on page 36.

32.9.2 ADC Characteristics

Table 32-23. Channel Conversion Time and ADC Clock

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{ADC}	ADC clock frequency	10-bit resolution mode			6	MHz
		8-bit resolution mode			6	
$t_{STARTUP}$	Startup time	Return from Idle Mode		15		µs
	Sample and hold acquisition time		500			ns
t_{CONV}	Conversion time (latency)	f_{ADC} = 6MHz	11		26	cycles
	Throughput rate	f_{ADC} = 6MHz, 10-bit resolution mode, low impedance source			460	kSPS
		f_{ADC} = 6MHz, 8-bit resolution mode, low impedance source			460	

Table 32-24. External Voltage Reference Input

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{ADVREFP}$	Reference voltage range	$V_{ADVREFP} = V_{VDDANA}$	1.62		1.98	V
	Current consumption on V_{VDDANA}	On 13 samples with ADC Clock = 5MHz				mA
$I_{ADVREFP}$	Average current	f_{ADC} = 6MHz			250	µA

Table 32-25. Analog Inputs

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{ADn}	Input Voltage Range	10-bit mode	0		$V_{ADVREFP}$	V
		8-bit mode				

32.9.2.1 Applicable Conditions and Derating Data

Table 32-26. Transfer Characteristics 10-bit Resolution Mode

Parameter	Conditions	Min	Typ	Max	Units
Resolution			10		Bit
Integral non-linearity	ADC clock frequency = 6MHz		+/-2		LSB
Differential non-linearity		-0.9		1	
Offset error			+/-4		
Gain error			+/-4		

Table 32-27. Transfer Characteristics 8-bit Resolution Mode

Parameter	Conditions	Min	Typ	Max	Units
Resolution			8		Bit

Table 32-27. Transfer Characteristics 8-bit Resolution Mode

Parameter	Conditions	Min	Typ	Max	Units
Integral non-linearity	ADC clock frequency = 6MHz		+/-0.5		LSB
Differential non-linearity		-0.23		0.25	
Offset error			+/-1		
Gain error			+/-1		

32.9.3 Analog Comparator Characteristics

Table 32-28. Analog Comparator Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Positive input voltage range		-0.2		V_{VDDIO} + 0.3	V
	Negative input voltage range		-0.2		V_{VDDIO} - 0.6	V
	Statistical offset	V_{ACREFN} = 1.0V, f_{AC}= 12MHz, filter length=2, hysteresis=0.(1)		20		mV
f_{AC}	Clock frequency for GCLK4				12	MHz
$t_{STARTUP}$	Startup time		3			cycles
	Input current per pin			0.2		µA/MHz(2)

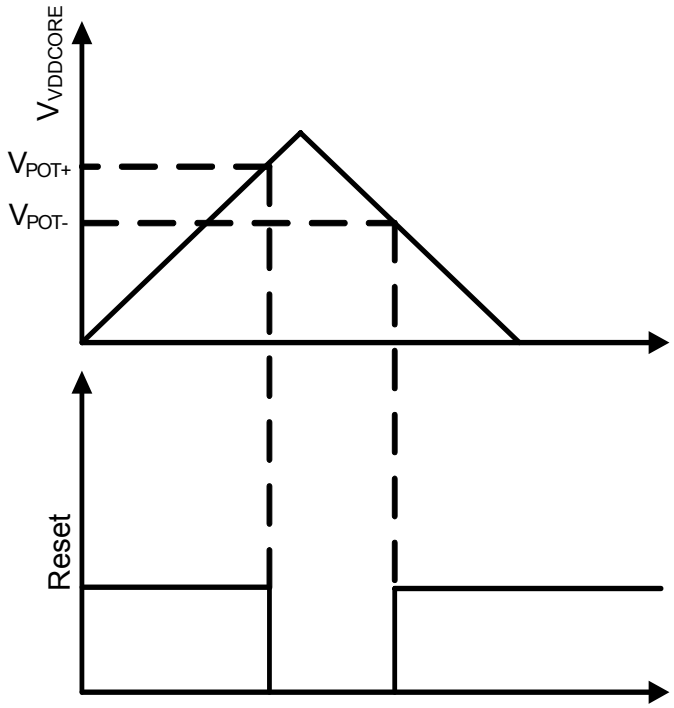
Notes: 1. AC.CONFn.FLEN and AC.CONFn.HYS fields, refer to the Analog Comparator Interface chapter.
2. Referring to f_{AC}.

32.9.4 POR18

Table 32-29. Power-on Reset Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT+}	Voltage threshold on $V_{VDDCORE}$ rising	T=25°C		1.45		V
V_{POT-}	Voltage threshold on $V_{VDDCORE}$ falling	T=25°C		1.32		V

Figure 32-4. POR18 Operating Principles

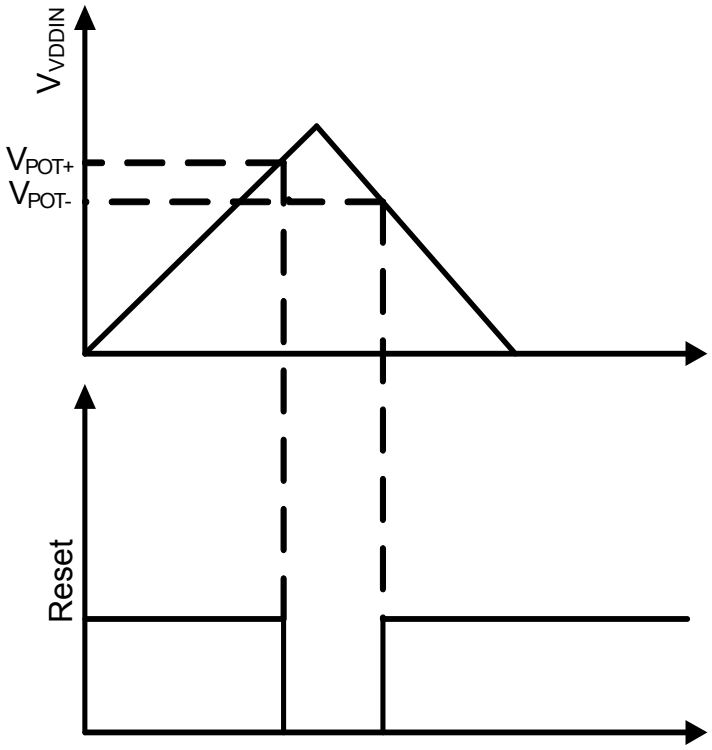


32.9.5 POR33

Table 32-30. POR33 Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT+}	Voltage threshold on V_{VDDIN} rising	T=25°C		1.49		V
V_{POT-}	Voltage threshold on V_{VDDIN} falling			1.45		

Figure 32-5. POR33 Operating Principles



32.9.6 Temperature Sensor

Table 32-31. Temperature Sensor Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Gradient			1		mV/°C

32.10 Timing Characteristics

32.10.1 RESET_N Characteristics

Table 32-32. RESET_N Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
t_{RESET}	RESET_N minimum pulse length		10		ns

33. Mechanical Characteristics

33.1 Thermal Considerations

33.1.1 Thermal Data

Table 33-1 summarizes the thermal resistance data depending on the package.

Table 33-1. Thermal Resistance Data

Symbol	Parameter	Condition	Package	Typ	Unit
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	TQFP48	63.2	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP48	21.8	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	QFN48	28.3	°C/W
θ_{JC}	Junction-to-case thermal resistance		QFN48	2.5	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	TLLGA48	30.06	°C/W
θ_{JC}	Junction-to-case thermal resistance		TLLGA48	TBD	

33.1.2 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. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

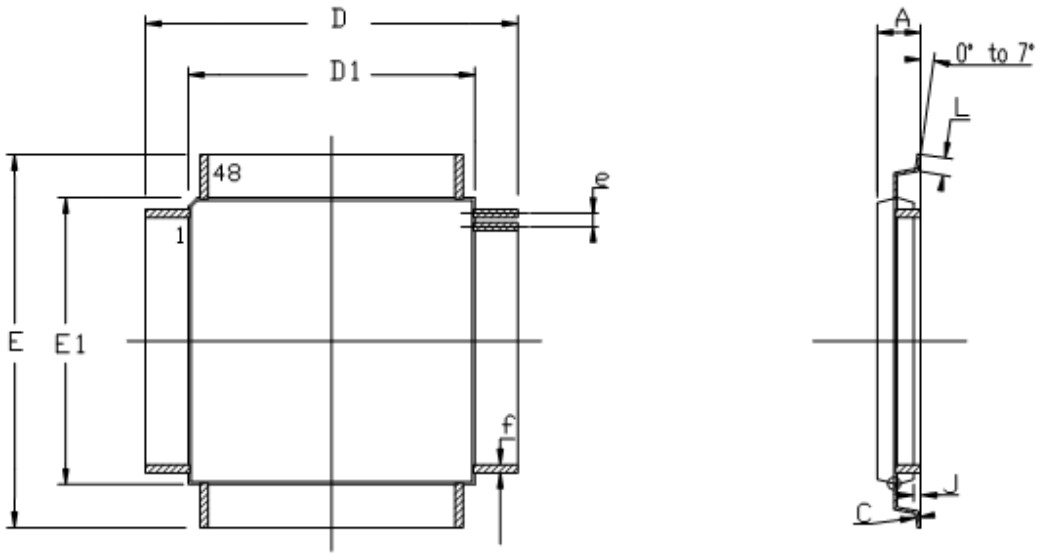
where:

- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 33-1.
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 33-1.
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W) estimated from data provided in the Section 32.5 on page 777.
- 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.

33.2 Package Drawings

Figure 33-1. TQFP-48 Package Drawing



COMMON DIMENSIONS IN MM

SYMBOL	Min	Max	NOTES
A	----	1.20	
A1	0.95	1.05	
C	0.09	0.20	
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
J	0.05	0.15	
L	0.45	0.75	

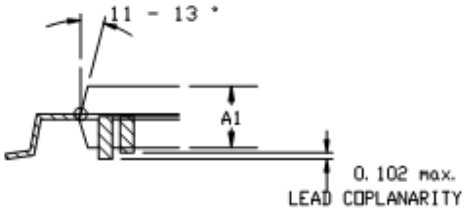


Table 33-2. Device and Package Maximum Weight

140	mg

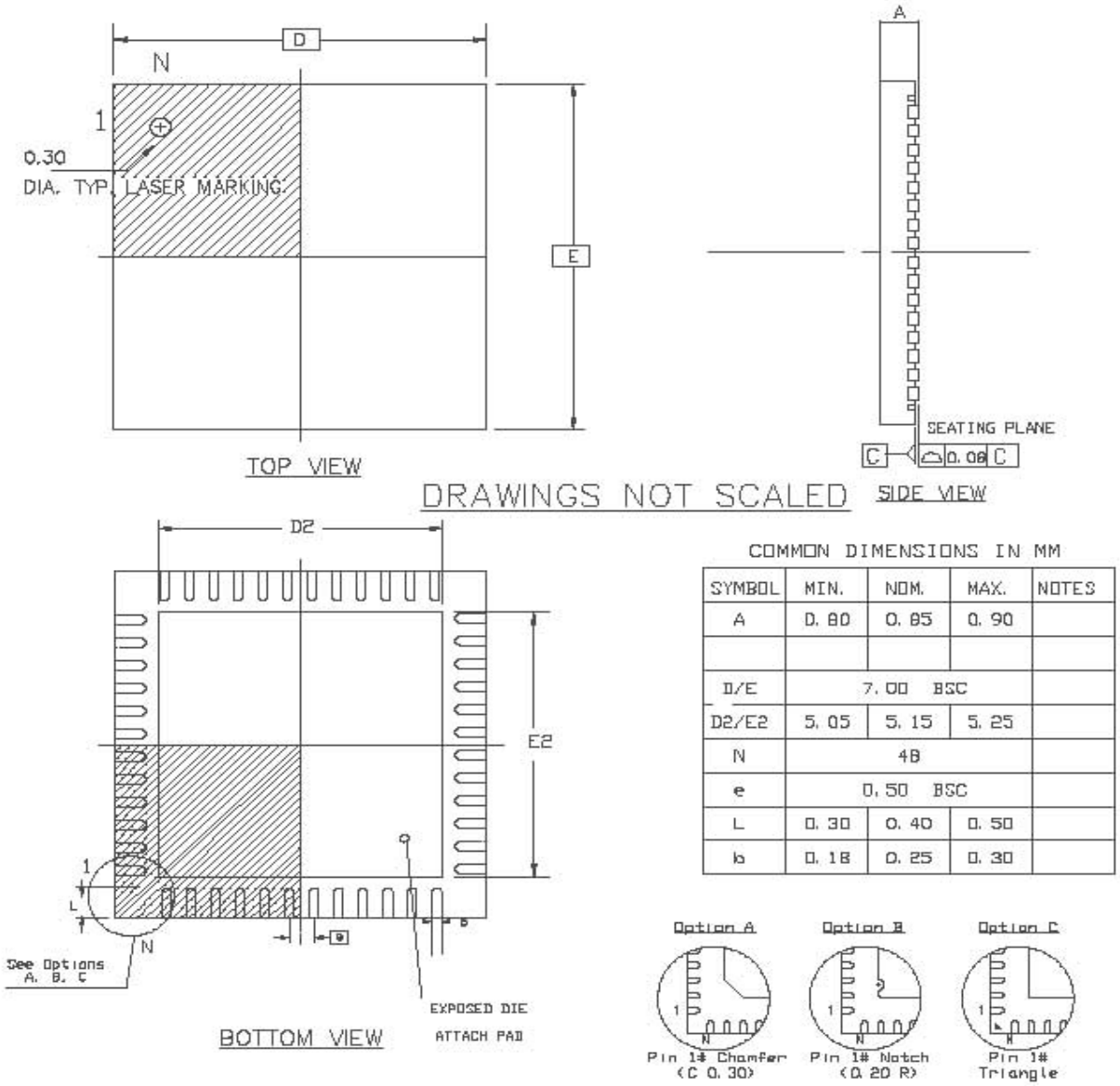
Table 33-3. Package Characteristics

Moisture Sensitivity Level	MSL3

Table 33-4. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

Figure 33-2. QFN-48 Package Drawing



SYMBOL	MIN.	NOM.	MAX.	NOTES
A	0.80	0.85	0.90	
D/E		7.00 BSC		
D2/E2	5.05	5.15	5.25	
N		48		
e		0.50 BSC		
L	0.30	0.40	0.50	
b	0.18	0.25	0.30	

Note: The exposed pad is not connected to anything.

Table 33-5. Device and Package Maximum Weight

140	mg

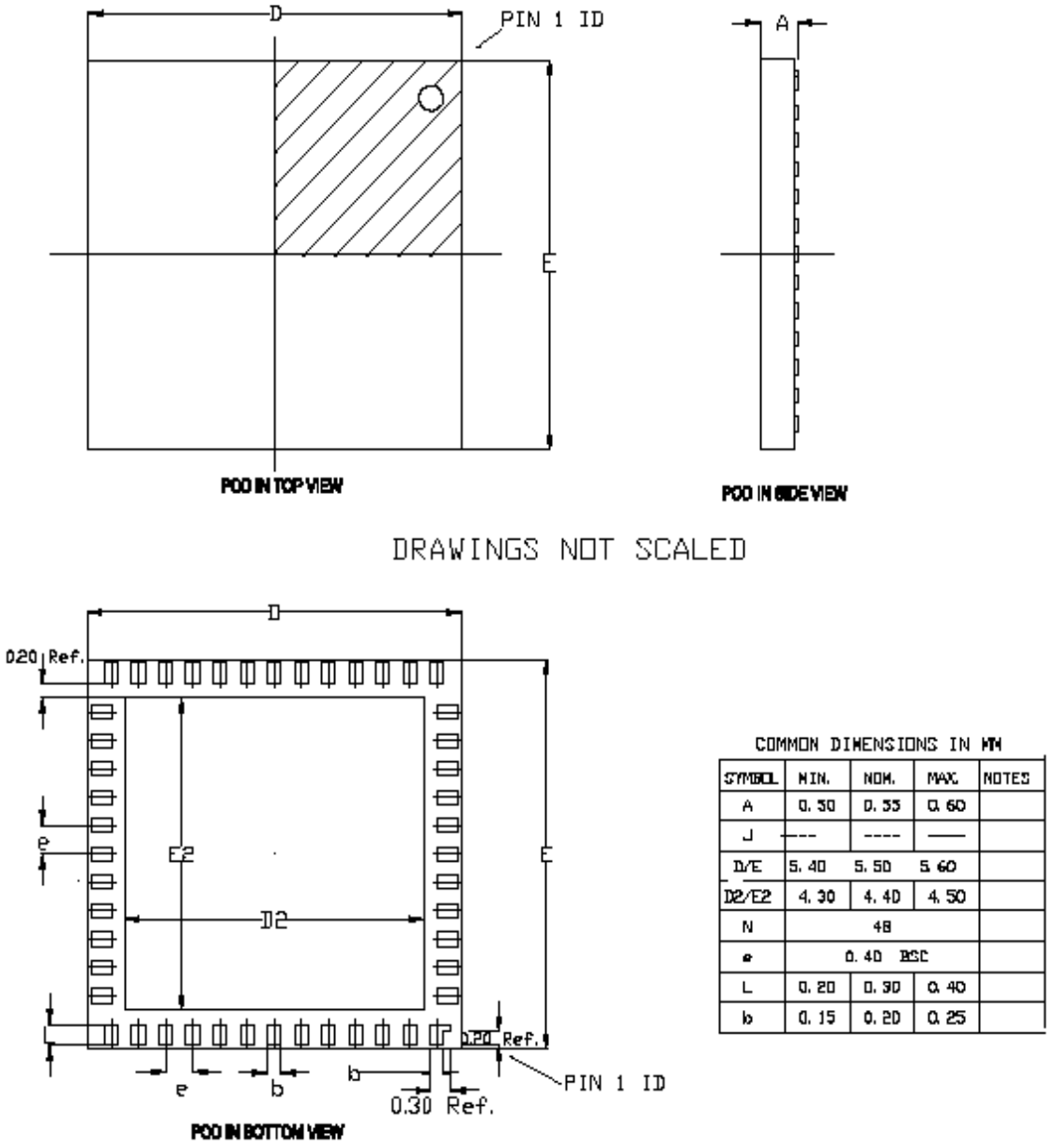
Table 33-6. Package Characteristics

Moisture Sensitivity Level	MSL3

Table 33-7. Package Reference

JEDEC Drawing Reference	M0-220
JESD97 Classification	E3

Figure 33-3. TLLGA-48 Package Drawing



COMMON DIMENSIONS IN MM

SYMBOL	MIN.	NOM.	MAX.	NOTES
A	0.50	0.55	0.60	
J	----	----	----	
D/E	5.40	5.50	5.60	
D2/E2	4.30	4.40	4.50	
N	48			
e	0.40 BSC			
L	0.20	0.30	0.40	
b	0.15	0.20	0.25	

Table 33-8. Device and Package Maximum Weight

39.3	mg

Table 33-9. Package Characteristics

Moisture Sensitivity Level	MSL3

Table 33-10. Package Reference

JEDEC Drawing Reference	M0-220
JESD97 Classification	E4

33.3 Soldering Profile

Table 33-11 gives the recommended soldering profile from J-STD-20.

Table 33-11. Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/s max
Preheat Temperature 175°C ±25°C	150-200°C
Time Maintained Above 217°C	60-150 s
Time within 5°C of Actual Peak Temperature	30 s
Peak Temperature Range	260°C
Ramp-down Rate	6°C/s max
Time 25°C to Peak Temperature	8 minutes max

A maximum of three reflow passes is allowed per component.

34. Ordering Information

Table 34-1. Ordering Information

Device	Ordering Code	Carrier Type	Package	Package Type	Temperature Operating Range
AT32UC3L064	AT32UC3L064-AUTES	ES	TQFP 48	JESD97 Classification E3	Industrial (-40°C to 85°C)
	AT32UC3L064-AUT	Tray			
	AT32UC3L064-AUR	Tape & Reel			
	AT32UC3L064-ZAUES	ES	QFN 48		
	AT32UC3L064-ZAUT	Tray			
	AT32UC3L064-ZAUR	Tape & Reel			
	AT32UC3L064-D3HES	ES	TLLGA 48	JESD97 Classification E4	
	AT32UC3L064-D3HT	Tray			
	AT32UC3L064-D3HR	Tape & Reel			
AT32UC3L032	AT32UC3L032-AUT	Tray	TQFP 48	JESD97 Classification E3	
	AT32UC3L032-AUR	Tape & Reel			
	AT32UC3L032-ZAUT	Tray	QFN 48		
	AT32UC3L032-ZAUR	Tape & Reel			
	AT32UC3L032-D3HT	Tray	TLLGA 48	JESD97 Classification E4	
	AT32UC3L032-D3HR	Tape & Reel			
AT32UC3L016	AT32UC3L016-AUT	Tray	TQFP 48	JESD97 Classification E3	
	AT32UC3L016-AUR	Tape & Reel			
	AT32UC3L016-ZAUT	Tray	QFN 48		
	AT32UC3L016-ZAUR	Tape & Reel			
	AT32UC3L016-D3HT	Tray	TLLGA 48	JESD97 Classification E4	
	AT32UC3L016-D3HR	Tape & Reel			

35. Errata

35.1 Rev. E

35.1.1 Processor and Architecture

1. **Privilege violation when using interrupts in application mode with protected system stack**
   If the system stack is protected by the MPU and an interrupt occurs in application mode, an MPU DTLB exception will occur.
   Fix/Workaround
   Make a DTLB Protection (Write) exception handler which permits the interrupt request to be handled in privileged mode.

2. **Hardware breakpoints may corrupt MAC results**
   Hardware breakpoints on MAC instructions may corrupt the destination register of the MAC instruction.
   Fix/Workaround
   Place breakpoints on earlier or later instructions.

35.1.2 FLASHCDW

1. **Flash Selfprogramming may fail in one wait state mode**
   Writes in flash and user pages may fail if executing code located in the address space mapped to the flash and if the flash controller is configured in one wait state mode (the Flash Wait State bit in the Flash Control Register (FCR.FWS) is 1).
   Fix/Workaround
   Solution 1: Configure the flash controller in zero wait state mode (FCR.FWS=0).
   Solution 2: Configure the HMATRIX master 1 (CPU Instruction) to use the unlimited burst length transfer mode (MCFG1.ULBT=0) and the HMATRIX slave 0 (FLASHCDW) to use the maximum slot cycle limit (SCFG0.SLOT_CYCLE=255).

35.1.3 Power Manager

1. **Clock sources will not be stopped in Static mode if the difference between CPU and PBx division factor is larger than 4**
   If the division factor between the CPU/HSB and PBx frequencies is more than 4 when entering a sleep mode where the system RC oscillator (RCSYS) is turned off, the high speed clock sources will not be turned off. This will result in a significantly higher power consumption during the sleep mode.
   Fix/Workaround
   Before going to sleep modes where RCSYS is stopped, make sure the division factor between the CPU/HSB and PBx frequencies is less than or equal to 4.

2. **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.

35.1.4 SCIF

1. **PCLKSR.OSC32RDY bit might not be cleared after disabling OSC32K**
   In some cases the OSC32RDY bit in the PCLKSR register will not be cleared when OSC32K is disabled.
   Fix/Workaround
   When re-enabling the OSC32K, read the PCLKSR.OSC32RDY bit. If this bit is:
   0: Follow normal procedures.
   1: Ignore the PCLKSR.OSC32RDY and ISR.OSC32RDY bit. Use the Frequency Meter (FREQM) to determine if the OSC32K clock is ready. The OSC32K clock is ready when the FREQM measures a non-zero frequency.

35.1.5 AST

1. **Reset may set status bits in the AST**
   If a reset occurs and the AST is enabled, the SR.ALARM0, SR.PER0, and SR.OVF bits may be set.
   Fix/Workaround
   If the part is reset and the AST is used, clear all bits in the Status Register (SR) before entering sleep mode.

2. **AST wake signal is released one AST clock cycle after the BUSY bit 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.

35.1.6 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 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.

35.1.7 GPIO

1. **Clearing GPIO interrupt may fail**
   Writing a one to the GPIO.IFRC register to clear the interrupt will be ignored if interrupt is enabled for the corresponding port.
   Fix / Workaround
   Disable the interrupt, clear the interrupt by writing a one to GPIO.IFRC, then enable the interrupt.

35.1.8 SPI

1. **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 by writing a one to the Software Reset bit in the Control Register (CR.SWRST).

2. **SPI Bad Serial Clock Generation on 2nd chip select when SCBR==1, CPOL==1, and NCPHA==0**
   When multiple chip selects are in use, if one of the baudrates is equal to 1 (CSRn.SCBR==1) and one of the others is not equal to 1, and CSRn.CPOL==1 and CSRn.NCPHA==0, an additional pulse will be generated on SCK.
   Fix/Workaround
   When multiple chip selects are in use, if one of the baudrates is equal to 1, the others must also be equal to 1 if CSRn.CPOL==1 and CSRn.NCPHA==0.

3. **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.

4. **Disabling SPI has no effect on the SR.TDRE bit**
   Disabling SPI has no effect on SR.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 SR.TDRE bit by writing to TDR. So if the SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR (as SR.TDRE stays high) until its buffer is empty, and all data written after the disable command is lost.
   Fix/Workaround
   Disable the PDCA, add 2 NOP (minimum), and disable the SPI. To continue the transfer, enable the SPI and the PDCA.

5. **SPI mode fault detection enable causes incorrect behavior**
   When mode fault detection is enabled (MR.MODFDIS==0), the SPI module may not operate properly.
   Fix/Workaround
   Always disable mode fault detection before using the SPI by writing a one to MR.MODFDIS.

35.1.9 TWI

1. **TWIM.SR.IDLE goes high immediately when NAK is received**
   When a NAK is received and there is a non-zero number of bytes to be transmitted, SR.IDLE goes high immediately and does not wait for the STOP condition to be sent. This does not cause any problem just by itself, but can cause a problem if software waits for SR.IDLE to go high and then immediately disables the TWIM by writing a one to CR.MDIS. Disabling the TWIM causes the TWCK and TWD pins to go high immediately, so the STOP condition will not be transmitted correctly.
   Fix/Workaround
   If possible, do not disable the TWIM. If it is absolutely necessary to disable the TWIM, there must be a software delay of at least two TWCK periods between the detection of SR.IDLE==1 and the disabling of the TWIM.

35.1.10 PWMA

1. **BUSY bit is never cleared after writes to the Control Register (CR)**
   When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR when the PWMA is disabled (CR.EN==0), the BUSY bit in the Status Register (SR.BUSY) will be set, but never cleared.
   Fix/Workaround
   When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR, make sure the PWMA is enabled, or simultaneously enable the PWMA by writing a one to CR.EN.

2. **Incoming peripheral events are discarded during duty cycle register update**
   Incoming peripheral events to all applied channels will be discarded if a duty cycle update is received from the user interface in the same PWMA clock period.
   Fix/Workaround
   Ensure that duty cycle writes from the user interface are not performed in a PWMA clock period when an incoming peripheral event is expected.

35.1.11 CAT

1. **CAT asynchronous wake will be delayed by one AST peripheral event period**
   If the CAT detects a condition that should asynchronously wake the chip in Static mode, the asynchronous wake will not occur until the next AST event. For example, if the AST is generating peripheral events to the CAT every 50 milliseconds, and the CAT detects a touch at t=9200 milliseconds, the asynchronous wake will occur at t=9250 milliseconds.
   Fix/Workaround
   None.

35.1.12 aWire

1. **aWire CPU clock speed robustness**
   The aWire memory speed request command counter wraps at clock speeds below approximately 5kHz.
   Fix/Workaround
   None.

2. **The aWire debug interface is reset after leaving Shutdown mode**
   If the aWire debug mode is used as debug interface and the program enters Shutdown mode, the aWire interface will be reset when the device receives a wake-up either from the WAKE_N pin or the AST.
   Fix/Workaround
   None.

35.1.13 I/O Pins

1. **PA17 has low ESD tolerance**
   PA17 only tolerates 500V ESD pulses (Human Body Model).
   Fix/Workaround
   Care must be taken during manufacturing and PCB design.

35.2 Rev. D

35.2.1 Processor and Architecture

1. **Privilege violation when using interrupts in application mode with protected system stack**
   If the system stack is protected by the MPU and an interrupt occurs in application mode, an MPU DTLB exception will occur.
   Fix/Workaround
   Make a DTLB Protection (Write) exception handler which permits the interrupt request to be handled in privileged mode.

2. **Hardware breakpoints may corrupt MAC results**
   Hardware breakpoints on MAC instructions may corrupt the destination register of the MAC instruction.
   Fix/Workaround
   Place breakpoints on earlier or later instructions.

35.2.2 FLASHCDW

1. **Flash Selfprogramming may fail in one wait state mode**
   Writes in flash and user pages may fail if executing code located in the address space mapped to the flash and if the flash controller is configured in one wait state mode (the Flash Wait State bit in the Flash Control Register (FCR.FWS) is 1).
   Fix/Workaround
   Solution 1: Configure the flash controller in zero wait state mode (FCR.FWS=0).
   Solution 2: Configure the HMATRIX master 1 (CPU Instruction) to use the unlimited burst length transfer mode (MCFG1.ULBT=0) and the HMATRIX slave 0 (FLASHCDW) to use the maximum slot cycle limit (SCFG0.SLOT_CYCLE=255).

35.2.3 Power Manager

1. **Clock sources will not be stopped in Static mode if the difference between CPU and PBx division factor is larger than 4**
   If the division factor between the CPU/HSB and PBx frequencies is more than 4 when entering a sleep mode where the system RC oscillator (RCSYS) is turned off, the high speed clock sources will not be turned off. This will result in a significantly higher power consumption during the sleep mode.
   Fix/Workaround
   Before going to sleep modes where RCSYS is stopped, make sure the division factor between the CPU/HSB and PBx frequencies is less than or equal to 4.

2. **External reset in Shutdown mode**
   If an external reset is asserted while the chip is in Shutdown mode, the Power Manager will register this as a Power-on reset (POR), and not as a SLEEP reset, in the Reset Cause register (RCAUSE).
   Fix/Workaround
   None.

3. **Disabling POR33 may generate spurious resets**
   Depending on operating conditions, POR33 may generate a spurious reset in one of the following cases:
   - When POR33 is disabled from the user interface.
   - When SM33 supply monitor is enabled.

- When entering Shutdown mode while debugging the chip using JTAG or aWire interface.
In the listed cases, writing a one to the bit VREGCR.POR33MASK in the System Control Interface (SCIF) to mask the POR33 reset will be ineffective.
Fix/Workaround
- Do not disable POR33 using the user interface.
- Do not use the SM33 supply monitor.
- Do not enter Shutdown mode if a debugger is connected to the chip.

4. **Instability when exiting sleep walking**
If all the following operating conditions are true, exiting sleep walking might lead to instability:
-The OSC0 is enabled in external clock mode (OSCCTRL0.OSCEN == 1 and OSCCTRL0.MODE == 0)
-A sleep mode where the OSC0 is automatically disabled is entered
-The chip enters sleep walking
Fix/Workaround
Do not run OSC0 in external clock mode if sleep walking is expected to be used.

5. **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.

35.2.4 SCIF

1. **PCLKSR.OSC32RDY bit might not be cleared after disabling OSC32K**
In some cases the OSC32RDY bit in the PCLKSR register will not be cleared when OSC32K is disabled.
Fix/Workaround
When re-enabling the OSC32K, read the PCLKSR.OSC32RDY bit. If this bit is:
0: Follow normal procedures.
1: Ignore the PCLKSR.OSC32RDY and ISR.OSC32RDY bit. Use the Frequency Meter (FREQM) to determine if the OSC32K clock is ready. The OSC32K clock is ready when the FREQM measures a non-zero frequency.

35.2.5 AST

1. **Reset may set status bits in the AST**
If a reset occurs and the AST is enabled, the SR.ALARM0, SR.PER0, and SR.OVF bits may be set.
Fix/Workaround
If the part is reset and the AST is used, clear all bits in the Status Register (SR) before entering sleep mode.

2. **AST wake signal is released one AST clock cycle after the BUSY bit 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.

35.2.6 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 s not cleared at all, the time before the reset will be twice as long as needed.

35.2.7 GPIO

1. **Clearing GPIO interrupt may fail**
   Writing a one to the GPIO.IFRC register to clear the interrupt will be ignored if interrupt is enabled for the corresponding port.
   Fix / Workaround
   Disable the interrupt, clear the interrupt by writing a one to GPIO.IFRC, then enable the interrupt.

35.2.8 SPI

1. **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 by writing a one to the Software Reset bit in the Control Register (CR.SWRST).

2. **SPI Bad Serial Clock Generation on 2nd chip select when SCBR==1, CPOL==1, and NCPHA==0**
   When multiple chip selects are in use, if one of the baudrates is equal to 1 (CSRn.SCBR==1) and one of the others is not equal to 1, and CSRn.CPOL==1 and CSRn.NCPHA==0, an additional pulse will be generated on SCK.
   Fix/Workaround
   When multiple chip selects are in use, if one of the baudrates is equal to 1, the others must also be equal to 1 if CSRn.CPOL==1 and CSRn.NCPHA==0.

3. **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.

4. **Disabling SPI has no effect on the SR.TDRE bit**
   Disabling SPI has no effect on SR.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 SR.TDRE bit by writing to TDR. So if the SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR (as SR.TDRE stays high) until its buffer is empty, and all data written after the disable command is lost.
   Fix/Workaround
   Disable the PDCA, add 2 NOP (minimum), and disable the SPI. To continue the transfer, enable the SPI and the PDCA.

5. **SPI mode fault detection enable causes incorrect behavior**
When mode fault detection is enabled (MR.MODFDIS==0), the SPI module may not operate properly.
Fix/Workaround
Always disable mode fault detection before using the SPI by writing a one to MR.MODFDIS.

35.2.9 TWI

1. **TWIM.SR.IDLE goes high immediately when NAK is received**
When a NAK is received and there is a non-zero number of bytes to be transmitted, SR.IDLE goes high immediately and does not wait for the STOP condition to be sent. This does not cause any problem just by itself, but can cause a problem if software waits for SR.IDLE to go high and then immediately disables the TWIM by writing a one to CR.MDIS. Disabling the TWIM causes the TWCK and TWD pins to go high immediately, so the STOP condition will not be transmitted correctly.
Fix/Workaround
If possible, do not disable the TWIM. If it is absolutely necessary to disable the TWIM, there must be a software delay of at least two TWCK periods between the detection of SR.IDLE==1 and the disabling of the TWIM.

35.2.10 PWMA

1. **BUSY bit is never cleared after writes to the Control Register (CR)**
When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR when the PWMA is disabled (CR.EN==0), the BUSY bit in the Status Register (SR.BUSY) will be set, but never cleared.
Fix/Workaround
When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR, make sure the PWMA is enabled, or simultaneously enable the PWMA by writing a one to CR.EN.

2. **Incoming peripheral events are discarded during duty cycle register update**
Incoming peripheral events to all applied channels will be discarded if a duty cycle update is received from the user interface in the same PWMA clock period.
Fix/Workaround
Ensure that duty cycle writes from the user interface are not performed in a PWMA clock period when an incoming peripheral event is expected.

35.2.11 CAT

1. **CAT asynchronous wake will be delayed by one AST peripheral event period**
If the CAT detects a condition that should asynchronously wake the chip in Static mode, the asynchronous wake will not occur until the next AST event. For example, if the AST is generating peripheral events to the CAT every 50 milliseconds, and the CAT detects a touch at t=9200 milliseconds, the asynchronous wake will occur at t=9250 milliseconds.
Fix/Workaround
None.

35.2.12 aWire

1. **aWire CPU clock speed robustness**
The aWire memory speed request command counter wraps at clock speeds below approximately 5kHz.
Fix/Workaround
None.

2. **The aWire debug interface is reset after leaving Shutdown mode**
   If the aWire debug mode is used as debug interface and the program enters Shutdown mode, the aWire interface will be reset when the device receives a wakeup either from the WAKE_N pin or the AST.
   Fix/Workaround
   None.

35.2.13 I/O Pins

1. **PA17 has low ESD tolerance**
   PA17 only tolerates 500V ESD pulses (Human Body Model).
   Fix/Workaround
   Care must be taken during manufacturing and PCB design.

35.2.14 Chip

1. **Increased Power Consumption in VDDIO in sleep modes**
   If the OSC0 is enabled in crystal mode when entering a sleep mode where the OSC0 is disabled, this will lead to an increased power consumption in VDDIO.
   Fix/Workaround
   Solution 1: Disable the OSC0 bt writing a zero to the Oscillator Enable bit in the System Control Interface (SCIF) Oscillator Control Register (SCIF.OSC0CTRL.OSCEN) before going to any sleep mode where the OSC0 is disabled
   Solution 2: Pull down or up XIN0 and XOUT0 with 1Mohm resistor.

2. **In 3.3V Single Supply Mode the Analog Comparator inputs affects the device's ability to start**
   When using the 3.3V Single Supply Mode the state of the Analog Comparator input pins can affect the device's ability to release POR reset.
   This is due to an interaction between the Analog Comparator input pins and the POR circuitry. The issue is not present in the 1.8V Single Supply Mode or the 3.3V Supply mode with 1.8V Regulated I/O Lines.
   Fix/Workaround:
   ACREFN (pin PA16) must be connected to GND until the POR reset is released and the Analog Comparator inputs should not be driven higher than 1.0V until the POR reset is released.

35.3 Rev. C

Not sampled.

35.4 Rev. B

35.4.1 Processor and Architecture

1. **Privilege violation when using interrupts in application mode with protected system stack**
   If the system stack is protected by the MPU and an interrupt occurs in application mode, an MPU DTLB exception will occur.
   Fix/Workaround
   Make a DTLB Protection (Write) exception handler which permits the interrupt request to be handled in privileged mode.

2. **Hardware breakpoints may corrupt MAC results**

Hardware breakpoints on MAC instructions may corrupt the destination register of the MAC instruction.
Fix/Workaround
Place breakpoints on earlier or later instructions.

3. **RETS behaves incorrectly when MPU is enabled**
RETS behaves incorrectly when MPU is enabled and MPU is configured so that system stack is not readable in unprivileged mode.
Fix/Workaround
Make system stack readable in unprivileged mode, or return from supervisor mode using rete instead of rets. This requires:
1. Changing the mode bits from 001 to 110 before issuing the instruction. Updating the mode bits to the desired value must be done using a single mtsr instruction so it is done atomically. Even if this step is described in general as not safe in the UC technical reference manual, it is safe in this very specific case.
2. Execute the RETE instruction.

35.4.2 FLASHCDW

1. **Flash Selfprogramming may fail in one wait state mode**
Writes in flash and user pages may fail if executing code located in the address space mapped to the flash and if the flash controller is configured in one wait state mode (the Flash Wait State bit in the Flash Control Register (FCR.FWS) is 1).
Fix/Workaround
Solution 1: Configure the flash controller in zero wait state mode (FCR.FWS=0).
Solution 2: Configure the HMATRIX master 1 (CPU Instruction) to use the unlimited burst length transfer mode (MCFG1.ULBT=0) and the HMATRIX slave 0 (FLASHCDW) to use the maximum slot cycle limit (SCFG0.SLOT_CYCLE=255).

2. **Chip Erase**
When performing a chip erase, the device may report that it is protected (IR=0x11) and that chiperase failed, even if the chip erase was succesful.
Fix/Workaround
Perform a reset before any further read and programming.

3. **Fuse Programming**
Programming of fuses does not work.
Fix/Workaround
Do not program fuses. All fuses will be erased during chiperase command.

4. **Wait 500 ns before reading from the flash after switching read mode**
After switching between normal read mode and high-speed read mode, the application must wait at least 500 ns before attempting any access to the flash.
Fix/Workaround
Solution 1: Make sure that the appropriate instructions are executed from RAM, and that a waiting-loop is executed from RAM waiting 500 ns or more before executing from flash.
Solution 2: Execute from flash with a clock with period longer than 500 ns. This guarantees that no new read access is attempted before the flash has had time to settle in the new read mode.

5. **VERSION register reads 0x100**
The VERSION register reads 0x100 instead of 0x102.
Fix/Workaround
None.

35.4.3 HMATRIX

1. **In the PRAS and PRBS registers, the MxPR fields are only two bits**
   In the PRAS and PRBS registers MxPR fields are only two bits wide, instead of four bits. The unused bits are undefined when reading the registers.
   Fix/Workaround
   Mask undefined bits when reading PRAS and PRBS.

35.4.4 SAU

1. **The SR.IDLE bit reads as zero**
   The IDLE bit in the Status Register (SR.IDLE) reads as zero.
   Fix/Workaround
   None.

2. **Open Mode is not functional**
   The Open Mode is not functional.
   Fix/workaround
   None.

3. **VERSION register reads 0x100**
   The VERSION register reads 0x100 instead of 0x110.
   Fix/Workaround
   None.

35.4.5 PDCA

1. **PCONTROL.CHxRES is nonfunctional**
   PCONTROL.CHxRES is nonfunctional. Counters are reset at power-on, and cannot be reset by software.
   Fix/Workaround
   SW needs to keep history of performance counters.

2. **Transfer error will stall a transmit peripheral handshake interface**
   If a transfer error is encountered on a channel transmitting to a peripheral, the peripheral handshake of the active channel will stall and the PDCA will not do any more transfers on the affected peripheral handshake interface.
   Fix/workaround
   Disable and then enable the peripheral after the transfer error.

3. **VERSION register reads 0x120**
   The VERSION register reads 0x120 instead of 0x122.
   Fix/Workaround
   None.

35.4.6 Power Manager

1. **Clock sources will not be stopped in Static mode if the difference between CPU and PBx division factor is larger than 4**
   If the division factor between the CPU/HSB and PBx frequencies is more than 4 when entering a sleep mode where the system RC oscillator (RCSYS) is turned off, the high speed clock sources will not be turned off. This will result in a significantly higher power consumption during the sleep mode.
   Fix/Workaround

Before going to sleep modes where RCSYS is stopped, make sure the division factor between the CPU/HSB and PBx frequencies is less than or equal to 4.

2. **Disabling POR33 may generate spurious resest**
   Depending on operating conditions, POR33 may generate a spurious reset in one of the following cases:
   - When POR33 is disabled from the user interface.
   - When SM33 supply monitor is enabled.
   - When entering Shutdown mode while debugging the chip using JTAG or aWire interface.

   In the listed cases, writing a one to the bit VREGCR.POR33MASK in the System Control Interface (SCIF) to mask the POR33 reset will be ineffective.
   Fix/Workaround
   - Do not disable POR33 using the user interface.
   - Do not use the SM33 supply monitor.
   - Do not enter Shutdown mode if a debugger is connected to the chip.

3. **CONFIG register reads 0x4F**
   The CONFIG register reads 0x4F instead of 0x43.
   Fix/Workaround
   None.

4. **PB writes via debugger in sleep modes are blocked during sleepwalking**
   During sleepwalking, PB writes performed by a debugger will be discarded by all PB modules except the module that is requesting the clock.
   Fix/Workaround
   None.

5. **VERSION register reads 0x400**
   The VERSION register reads 0x400 instead of 0x411.
   Fix/Workaround
   None.

6. **WCAUSE register should not be used**
   The WCAUSE register should not be used.
   Fix/Workaround
   None.

7. **Static mode cannot be entered if the WDT is using OSC32K**
   If the WDT is using OSC32K as clock source and the user tries to enter Static mode, the Deepstop mode will be entered instead.
   Fix/Workaround
   None.

8. **It is not possible to mask the request clock requests**
   It is not possible to mask the request clock requests using PPCR.
   Fix/Workaround
   None.

9. **Clock failure detector (CFD) does not work**
   The clock failure detector does not work.
   Fix/Workaround
   None.

10. **Instability when exiting sleep walking**

If all the following operating conditions are true, exiting sleep walking might lead to instability:
-The OSC0 is enabled in external clock mode (OSCCTRL0.OSCEN == 1 and OSCCTRL0.MODE == 0)
-A sleep mode where the OSC0 is automatically disabled is entered
-The chip enters sleep walking
Fix/Workaround
Do not run OSC0 in external clock mode if sleep walking is expected to be used.

35.4.7 SCIF

1. **The DFLL should be slowed down before disabled**
   The frequency of the DFLL should be set to minimum before disabled.
   Fix/Workaround
   Before disabling the DFLL the value of the COARSE register should be set to zero.

2. **Writing to ICR masks new interrupts received in the same clock cycle**
   Writing to ICR masks any new SCIF interrupt received in the same clock cycle, regardless of write value.
   Fix/Workaround
   For every interrupt except BODDET, SM33DET, and VREGOK the CLKSR register can be read to detect new interrupts. BODDET, SM33DET, and VREGOK interrupts will not be generated if they occur when writing to ICR.

3. **FINE value for DFLL is not correct when dithering is disabled**
   In open loop mode, the FINE value used by the DFLL DAC is offset by two compared to the value written to the DFLL0CONF.FINE field. I.e. the value to the DFLL DAC is DFLL0CONF.FINE-0x002. If DFLL0CONF.FINE is written to 0x000, 0x001, or 0x002 the value to the DFLL DAC will be 0x1FE, 0x1FF, or 0x000 respectively.
   Fix/Workaround
   Write the desired value added by two to the DFLL0CONF.FINE field.

4. **BODVERSION register reads 0x100**
   The BODVERSION register reads 0x100 instead of 0x101.
   Fix/Workaround
   None.

5. **BRIFA is non-functional**
   BRIFA is non-functional.
   Fix/Workaround
   None.

6. **VREGCR.DEEPMODEDISABLE bit is not readable**
   VREGCR.DEEPMODEDISABLE bit is not readable.
   Fix/Workaround
   None.

7. **DFLL step size should be 7 or lower below 30 MHz**
   If max step size is above 7, the DFLL might not lock at the correct frequency if the target frequency is below 30 MHz.
   Fix/Workaround
   If the target frequency is below 30 MHz, use max step size (DFLL0MAXSTEP.MAXSTEP) of 7 or lower.

8. **Generic clock sources are kept running in sleep modes**
   If a clock is used as a source for a generic clock when going to a sleep mode where clock sources are stopped, the source of the generic clock will be kept running. Please refer to the Power Manager chapter for details about sleep modes.
   Fix/Workaround
   Disable generic clocks before going to sleep modes where clock sources are stopped to save power.

9. **DFLL clock is unstable with a fast reference clock**
   The DFLL clock can be unstable when a fast clock is used as reference clock in closed loop mode.
   Fix/Workaround
   Use the 32 KHz crystal oscillator clock or a clock with similar frequency as DFLLIF reference clock.

10. **DFLLIF indicates coarse lock too early**
    The DFLLIF might indicate coarse lock too early, the DFLL will lose coarse lock and regain it later.
    Fix/Workaround
    Use max step size (DFLL0MAXSTEP.MAXSTEP) of 4 or higher.

11. **DFLLIF dithering does not work**
    The DFLLIF dithering does not work.
    Fix/Workaround
    None.

12. **SCIF VERSION register reads 0x100**
    The VERSION register reads 0x100 instead of 0x102.
    Fix/Workaround
    None.

13. **DFLLVERSION register reads 0x200**
    The DFLLVERSION register reads 0x200 instead of 0x201.
    Fix/Workaround
    None.

14. **RCCRVERSION register reads 0x100**
    The RCCRVERSION register reads 0x100 instead of 0x101.
    Fix/Workaround
    None.

15. **OSC32VERSION register reads 0x100**
    The OSC32VERSION register reads 0x100 instead of 0x101.
    Fix/Workaround
    None.

16. **VREGVERSION register reads 0x100**
    The VREGVERSION register reads 0x100 instead of 0x101.
    Fix/Workaround
    None.

17. **RC120MVERSION register reads 0x100**
    The RC120MVERSION register reads 0x100 instead of 0x101.
    Fix/Workaround

None.

18. **GCLK5 is non-functional**
GCLK5 is non-functional.
Fix/Workaround
None.

19. **DFLLIF might loose fine lock when dithering is disabled**
When dithering is disabled, and fine lock has been acquired the DFLL might loose the fine lock resulting in a up to 20% over-/undershoot.
Fix/Workaround
Solution 1: When the DFLL is used as main clock source the target frequency of the DFLL should be 20% below the maximum operating frequency of the CPU. Don't use the DFLL as clock source for frequency sensitive applications.
Solution 2: Do not use the DFLL in closed loop mode.

20. **PCLKSR.OSC32RDY bit might not be cleared after disabling OSC32K**
In some cases the OSC32RDY bit in the PCLKSR register will not be cleared when OSC32K is disabled.
Fix/Workaround
When re-enabling the OSC32K, read the PCLKSR.OSC32RDY bit. If this bit is:
0: Follow normal procedures.
1: Ignore the PCLKSR.OSC32RDY and ISR.OSC32RDY bit. Use the Frequency Meter (FREQM) to determine if the OSC32K clock is ready. The OSC32K clock is ready when the FREQM measures a non-zero frequency.

35.4.8 AST

1. **AST wake signal is released one AST clock cycle after the BUSY bit 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.

35.4.9 WDT

1. **Clearing of the WDT in window mode**
In window mode, if the WDT is cleared 2^{TBAN} CLK_WDT cycles after entering the window, the counter will be cleared, but will not exit the window. If this occurs, the SR.WINDOW bit will not be cleared after clearing the WDT.
Fix/Workaround
Check SR.WINDOW immediately after clearing the WDT. If set then clear the WDT once more.

2. **VERSION register reads 0x400**
The VERSION register reads 0x400 instead of 0x402.
Fix/Workaround
None.

3. **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 the WDT counter is cleared after the first half of the timeout period, you will get a Watchdog reset immediately. If the WDT coutner s not cleared at all, the time before the reset will be twice as long as needed.

35.4.10 FREQM

1. **Measured clock (CLK_MSR) sources 15-17 are shifted**

   CLKSEL = 14 selects the RC120M AW clock, CLKSEL = 15 selects the RC120M clock, and CLKSEL = 16 selects the RC32K clock as source for the measured clock (CLK_MSR).

   Fix/Workaround

   None.

2. **GCLK5 can not be used as source for the CLK_MSR**

   The frequency for GCLK5 can not be measured by the FREQM.

   Fix/Workaround

   None.

35.4.11 GPIO

1. **GPIO interrupt can not be cleared when interrupts are disabled**

   The GPIO interrupt can not be cleared unless the interrupt is enabled for the pin.

   Fix/Workaround

   Enable interrupt for the corresponding pin, then clear the interrupt.

2. **VERSION register reads 0x210**

   The VERSION register reads 0x210 instead of 0x211.

   Fix/Workaround

   None.

35.4.12 USART

1. **The RTS output does not function correctly in hardware handshaking mode**

   The RTS signal is not generated properly when the USART receives data in hardware handshaking mode. When the Peripheral DMA receive buffer becomes full, the RTS output should go high, but it will stay low.

   Fix/Workaround

   Do not use the hardware handshaking mode of the USART. If it is necessary to drive the RTS output high when the Peripheral DMA receive buffer becomes full, use the normal mode of the USART. Configure the Peripheral DMA Controller to signal an interrupt when the receive buffer is full. In the interrupt handler code, write a one to the RTSDIS bit in the USART Control Register (CR). This will drive the RTS output high. After the next DMA transfer is started and a receive buffer is available, write a one to the RTSEN bit in the USART CR so that RTS will be driven low.

35.4.13 SPI

1. **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 by writing a one to the Software Reset bit in the Control Register (CR.SWRST).

2. **SPI Bad Serial Clock Generation on 2nd chip select when SCBR==1, CPOL==1, and NCPHA==0**
   When multiple chip selects are in use, if one of the baudrates is equal to 1 (CSRn.SCBR==1) and one of the others is not equal to 1, and CSRn.CPOL==1 and CSRn.NCPHA==0, an additional pulse will be generated on SCK.
   Fix/Workaround
   When multiple chip selects are in use, if one of the baudrates is equal to 1, the others must also be equal to 1 if CSRn.CPOL==1 and CSRn.NCPHA==0.

3. **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.

4. **Disabling SPI has no effect on the SR.TDRE bit**
   Disabling SPI has no effect on SR.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 SR.TDRE bit by writing to TDR. So if the SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR (as SR.TDRE stays high) until its buffer is empty, and all data written after the disable command is lost.
   Fix/Workaround
   Disable the PDCA, add 2 NOP (minimum), and disable the SPI. To continue the transfer, enable the SPI and the PDCA.

6. **SPI mode fault detection enable causes incorrect behavior**
   When mode fault detection is enabled (MR.MODFDIS==0), the SPI module may not operate properly.
   Fix/Workaround
   Always disable mode fault detection before using the SPI by writing a one to MR.MODFDIS.

35.4.14 TWI

1. **TWIM Version Register reads zero**
   TWIM Version Register (VR) reads zero instead of 0x101.
   Fix/Workaround
   None.

2. **TWIS Version Register reads zero**
   TWIS Version Register (VR) reads zero instead of 0x112.
   Fix/Workaround
   None.

3. **TWIS CR.STREN does not work in deep sleep modes**
   When the device is in Stop, DeepStop, or Static mode, address reception will not wake the device if both CR.SOAM and CR.STREN are one.
   Fix/Workaround
   Do not write both CR.STREN and CR.SOAM to one if the device needs to wake from deep sleep modes.

4. **TWI pins are not SMBus compliant**

The TWI pins draws current when the pins are supplied with 3.3V and the part is left unpowered.
Fix/Workaround
None.

5. **PA21, PB04, and PB05 are not 5V tolerant**
Pins PA21, PB04, and PB05 are only 3.3V tolerant, not 5V tolerant.
Fix/Workaround
None.

6. **PB04 SMBALERT function should not be used**
The SMBALERT function from TWIMS0 should not be selected on pin PB04.
Fix/Workaround
None.

7. **TWIMS0.TWCK on PB05 is non-functional**
TWIMS0.TWCK on PB05 is non-functional.
Fix/Workaround
Use TWI0.TWCK on other pins.

8. **TWIM STOP bit in IMR always read as zero**
The STOP bit in IMR always reads as zero.
Fix/Workaround
None.

9. **TWIM.SR.IDLE goes high immediately when NAK is received**
When a NAK is received and there is a non-zero number of bytes to be transmitted, SR.IDLE goes high immediately and does not wait for the STOP condition to be sent. This does not cause any problem just by itself, but can cause a problem if software waits for SR.IDLE to go high and then immediately disables the TWIM by writing a one to CR.MDIS. Disabling the TWIM causes the TWCK and TWD pins to go high immediately, so the STOP condition will not be transmitted correctly.
Fix/Workaround
If possible, do not disable the TWIM. If it is absolutely necessary to disable the TWIM, there must be a software delay of at least two TWCK periods between the detection of SR.IDLE==1 and the disabling of the TWIM.

10. **Disabled TWIM drives TWD and TWCK low**
When the TWIM is disabled, it drives the TWD and TWCK signals with logic level zero. This can lead to communication problems with other devices on the TWI bus.
Fix/Workaround
Enable the TWIM first and then enable the TWD and TWCK peripheral pins in the GPIO controller. If it is necessary to disable the TWIM, first disable the TWD and TWCK peripheral pins in the GPIO controller and then disable the TWIM.

35.4.15 PWMA

1. **PARAMETER register reads 0x2424**
The PARAMETER register reads 0x2424 instead of 0x24.
Fix/Workaround
None.

2. **Open drain mode does not work**
The open drain mode does not work.

Fix/Workaround
None.

3. **VERSION register reads 0x100**
The VERSION register reads 0x100 instead of 0x101.
Fix/Workaround
None.

4. **Writing to the duty cycle registers when the timebase counter overflows can give an undefined result**
The duty cycle registers will be corrupted if written when the timebase counter overflows. If the duty cycle registers are written exactly when the timebase counter overflows at TOP, the duty cycle registers may become corrupted.
Fix/Workaround
Write to the duty cycle registers only directly after the Timebase Overflow bit in the status register is set.

5. **BUSY bit is never cleared after writes to the Control Register (CR)**
When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR when the PWMA is disabled (CR.EN==0), the BUSY bit in the Status Register (SR.BUSY) will be set, but never cleared.
Fix/Workaround
When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR, make sure the PWMA is enabled, or simultaneously enable the PWMA by writing a one to CR.EN.

6. **Incoming peripheral events are discarded during duty cycle register update**
Incoming peripheral events to all applied channels will be discarded if a duty cycle update is received from the user interface in the same PWMA clock period.
Fix/Workaround
Ensure that duty cycle writes from the user interface are not performed in a PWMA clock period when an incoming peripheral event is expected.

35.4.16 TC

1. **When the main clock is RCSYS, TIMER_CLOCK5 is equal to CLK_PBA**
When the main clock is generated from RCSYS, TIMER_CLOCK5 is equal to CLK_PBA and not CLK_PBA/128.
Fix/Workaround
None.

35.4.17 ADCIFB

1. **Pendetect in sleep modes without CLK_ADCIFB will not wake the system**
The pendetect will not wake the system from a sleep mode if the clock for the ADCIFB (CLK_ADCIFB) is turned off.
Fix/Workaround
Use a sleep mode where CLK_ADCIFB is not turned off to wake the part using pendetect.

2. **8-bit mode is not working**
Do not use the 8-bit mode of the ADCIFB.
Fix/Workaround
Use the 10-bit mode and shift right by 2 bits.

3. **ADC channels six to eight are non-functional**
   ADC channels six to eight are non-functional.
   Fix/Workaround
   None.

4. **VERSION register reads 0x100**
   The VERSION register reads 0x100 instead of 0x101.
   Fix/Workaround
   None.

35.4.18 ACIFB

1. **Negative offset**
   The static offset of the analog comparator is appriximately -50mV.
   Fix/Workaround
   None.

2. **Generic clock sources in sleep modes**
   The ACIFB should not use RC32K or CLK_1K as generic clock source if the chip uses sleep modes.
   Fix/Workaround
   None.

3. **VERSION register reads 0x200**
   The VERSION register reads 0x200 instead of 0x212.
   Fix/Workaround
   None.

4. **CONFW.WEVSRC and CONFW.WEVEN are not correctly described in the user interface**
   CONFW.WEVSRC is only two bits instead of three bits wide. Only values 0, 1, and 2 can be written to this register. CONFW.WEVEN is in bit position 10 instead of 11.
   Fix/Workaround
   Only write values 0, 1, and 2 to CONFW.WEVSRC. When reading CONFW.WEVSRC, disregard the third bit. Read/write bit 10 to access CONFW.WEVEN.

35.4.19 CAT

1. **Switch off discharge current when reaching 0V**
   The discharge current will switch off when reaching MGCFG1.MAX, not when reaching 0V.
   Fix/Workaround
   None.

2. **CAT external capacitors are not clamped to ground when CAT is idle**
   The CAT module does not clamp the external capacitors to ground when it is idle. The capacitors are left floating, so they could accumulate small amounts of charge.
   Fix/workaround
   None.

3. **DISHIFT field is stuck at zero**
   The DISHIFT field in the MGCFG1, TGACFG1, TGBCFG1, and ATCFG1 registers is stuck at zero and cannot be written to a different value. Capacitor discharge time will be determined only by the DILEN field.
   Fix/Workaround

None.

4. **MGCFG2.ACCTRL bit is stuck at zero**
The ACCTRL bit in the MGCFG2 register is stuck at zero and cannot be written to one. The analog comparators will be constantly enabled.
Fix/Workaround
None.

5. **MGCFG2.CONSEN field is stuck at zero**
The CONSEN field in the MGCFG2 register is stuck at zero and cannot be written to a different value. The CAT consensus filter does not function properly, so termination of QMatrix data acquisition is controlled only by the MAX field in MGCFG1.
Fix/Workaround
None.

6. **VERSION register reads 0x100**
The VERSION register reads 0x100 instead of 0x200.
Fix/Workaround
None.

1. **CAT asynchronous wake will be delayed by one AST peripheral event period**
If the CAT detects a condition that should asynchronously wake the chip in Static mode, the asynchronous wake will not occur until the next AST event. For example, if the AST is generating peripheral events to the CAT every 50 milliseconds, and the CAT detects a touch at t=9200 milliseconds, the asynchronous wake will occur at t=9250 milliseconds.
Fix/Workaround
None.

35.4.20 GLOC

1. **GLOC is non-functional**
Glue Logic Controller (GLOC) is non-functional.
Fix/Workaround
None.

35.4.21 aWire

1. **aWire PB mapping and PB clock mask number**
The aWire PB has a different PB address and PB clock mask number.
Fix/Workaround
Use aWire PB address 0xFFFF6C00 and PB clock (PBAMASK) 24.

2. **SAB multiaccess reads are not working**
Reading more than one word, halfword, or byte in one command is not working correctly.
Fix/Workaround
Split the access into several single word, halfword, or byte accesses.

3. **If a reset happens during the last SAB write, the aWire will stall**
If a reset happens during the last word, halfword, or byte write the aWire will wait forever for an acknowledge from the SAB.
Fix/Workaround
Reset the aWire by keeping the RESET_N line low for 100 ms.

4. **aWire enable does not work in Static mode**

aWire enable does not work in Static mode.
Fix/Workaround
None.

5. **VERSION register reads 0x200**
The VERSION register reads 0x200 instead of 0x210.
Fix/Workaround
None.

6. **The aWire debug interface is reset after leaving Shutdown mode**
If the aWire debug mode is used as debug interface and the program enters Shutdown mode, the aWire interface will be reset when the device receives a wakeup either from the WAKE_N pin or the AST.
Fix/Workaround
None.

35.4.22 I/O Pins

1. **PB10 is not 3.3V tolerant**
PB10 should be grounded on the PCB and left unused.
Fix/Workaround
None.

2. **Analog multiplexing consumes extra power**
Current consumption on VDDIO increases when the voltage on analog inputs is close to VDDIO/2.
Fix/Workaround
None.

3. **PA02, PB01, PB04, PB05, and RESET_N have half of the pull-up strength**
Pins PA02, PB01, PB04, PB05, and RESET_N have half of the specified pull-up strength.
Fix/Workaround
None.

4. **OCD MCKO and MDO[3] are swapped in the AUX1 mapping**
When using the OCD AUX1 mapping of trace signals MDO[3] is located on pin PB05 and MCKO is located on PB01.
Fix/Workaround
Swap pins PB01 and PB05 if using OCD AUX1.

5. **The JTAG is enabled at power up**
The JTAG function on pins PA00, PA01, PA02, and PA03, are enabled after startup. Normal I/O module functionality is not possible on these pins.
Fix/Workaround
Add a 10kOhm pullup on the reset line.

35.4.23 Chip

1. **Power consumption in static mode is too high**
Power consumption in static mode is too high when PA21 is high.
Fix/Workaround
Ensure PA21 is low.

2. **Shutdown mode is not functional**

Do not enter Shutdown mode.
Fix/Workaround
None.

3. **VDDIN current consumption increase above 1.8V**
   When VDDIN increases above 1.8V, current on VDDIN increases with up to 40µA.
   Fix/Workaround
   None.

4. **Increased Power Consumption in VDDIO in sleep modes**
   If the OSC0 is enabled in crystal mode when entering a sleep mode where the OSC0 is disabled, this will lead to an increased power consumption in VDDIO.
   Fix/Workaround
   Solution 1: Disable the OSC0 bt writing a zero to the Oscillator Enable bit in the System Control Interface (SCIF) Oscillator Control Register (SCIF.OSC0CTRL.OSCEN) before going to any sleep mode where the OSC0 is disabled
   Solution 2: Pull down or up XIN0 and XOUT0 with 1Mohm resistor.

36. 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.

36.1 Rev. D - 06/2010

1. Ordering Information: Ordering code for TQFP ES changed from AT32UC3L064-AUES to AT32UC3L064-AUTES. TLLGA48 Tray option added.

36.2 Rev. C - 06/2010

1. Features and Description: Added QTouch library support.
2. USART: Description of unimplemented features removed.
3. Electrical Characteristics: Power Consumption numbers updated. Flash timing numbers added.

36.3 Rev. B - 05/2010

1. Package and Pinout: Added pinout figure for TLLGA48 package.
2. Package and Pinout, GPIO function multiplexing:TWIMS0-TWCK on PA20 removed. ADCIFB-AD[3] on PA17 removed, number of ADC channels are 8, not 9.
3. I/O Lines Considerations: Added: Following pins have high-drive capability: PA02, PA06, PA08, PA09, and PB01.
   Some TWI0 pins are SMBUS compliant (PA21, PB04, PB05).
4. HMATRIX Masters: PDCA is master 4, not master 3. SAU is master 3, not master 4.
5. SAU: IDLE bit added in the Status Register.
6. PDCA: Number of PDCA performance monitors is device dependent.
7. Peripheral Event System: Chapter updated.
8. PM: Bits in RCAUSE registers removed and renamed (JTAGHARD and AWIREHARD renamed to JTAG and AWIRE respectively, JTAG and AWIRE removed. BOD33 bit removed).
9. PM: RCAUSE.BOD33 bit removed. SM33 reset will be detected as a POR reset.
10. PM: WDT can be used as wake-up source if WDT is clocked from 32 KHz oscillator.
11. PM: Entering Shutdown mode description updated.
12. SCIF: DFLL output frequency is 40-150 MHz, not 20-150 MHz or 30-150 MHz.
13. SCIF: Temperature sensor is connected to ADC channel 9, not 7.
14. SCIF: Updated the oscillator connection figure for OSC0
15. GPIO: Removed unimplemented features (pull-down, buskeeper, drive strength, slew rate, schmitt trigger, open drain).

16. SPI: RDR.PCS field removed (RDR[19:16]).
17. TWIS: Figures updated.
18. ADCIFB: The sample and hold time and the startup time formulas have been corrected (ADC Configuration Register).
19. ADCIFB: Updated ADC signal names.
20. ACIFB: CONFW.WEVSRC is bit 8-10, CONFW.EWEVEN is bit 11. CONF.EVENP and CONF.EVENN bits are swapped.
21. CAT: Matrix size is 16 by 8, not 18 by 8.
22. Electrical Characteristics: General update.
23. Mechanical Characteristics: Added numbers for package drawings.
24. Mechanical Characteristics: In the TQFP-48 package drawing the Lead Coplanarity is 0.102mm, not 0.080mm.
25. Ordering Information: Ordering code for TLLGA-48 package updated.

36.4 Rev. A – 06/2009

1. Initial revision.

Table of Contents

Features *1*

1 Description *3*

2 Overview *5*

2.1 Block Diagram 5

2.2 Configuration Summary 6

3 Package and Pinout *7*

3.1 Package 7

3.2 Peripheral Multiplexing on I/O lines 9

3.3 Signal Descriptions 13

3.4 I/O Line Considerations 16

4 Processor and Architecture *18*

4.1 Features 18

4.2 AVR32 Architecture 18

4.3 The AVR32UC CPU 19

4.4 Programming Model 23

4.5 Exceptions and Interrupts 27

5 Memories *32*

5.1 Embedded Memories 32

5.2 Physical Memory Map 32

5.3 Peripheral Address Map 33

5.4 CPU Local Bus Mapping 34

6 Supply and Startup Considerations *36*

6.1 Supply Considerations 36

6.2 Startup Considerations 40

7 Flash Controller (FLASHCDW) *41*

7.1 Features 41

7.2 Overview 41

7.3 Product Dependencies 41

7.4 Functional Description 42

7.5 Flash Commands 47

7.6 General-purpose Fuse Bits 49

7.7 Security Bit 52

7.8 User Interface 53
7.9 Fuse Settings 63
7.10 Module Configuration 65

8 HSB Bus Matrix (HMATRIX) 66

8.1 **Features 66**
8.2 Overview 66
8.3 Product Dependencies 66
8.4 Functional Description 66
8.5 User Interface 70
8.6 Module Configuration 78

9 Secure Access Unit (SAU) 80

9.1 Features 80
9.2 Overview 80
9.3 Block Diagram 81
9.4 Product Dependencies 82
9.5 Functional Description 82
9.6 User Interface 86
9.7 Module Configuration 103

10 Peripheral DMA Controller (PDCA) 104

10.1 Features 104
10.2 Overview 104
10.3 Block Diagram 105
10.4 Product Dependencies 105
10.5 Functional Description 106
10.6 Performance Monitors 108
10.7 User Interface 110
10.8 Module Configuration 138

11 Peripheral Event System 140

11.1 Features 140
11.2 Overview 140
11.3 Peripheral Event System Block Diagram 140
11.4 Functional Description 140
11.5 Application Example 143

12 Interrupt Controller (INTC) 144

12.1 Features 144
12.2 Overview 144
12.3 Block Diagram 144
12.4 Product Dependencies 145
12.5 Functional Description 145
12.6 User Interface 148
12.7 Module Configuration 152

13 Power Manager (PM) 155

13.1 Features 155
13.2 Overview 155
13.3 Block Diagram 156
13.4 I/O Lines Description 156
13.5 Product Dependencies 156
13.6 Functional Description 157
13.7 User Interface 165
13.8 Module Configuration 189

14 System Control Interface (SCIF) 190

14.1 Features 190
14.2 Overview 190
14.3 I/O Lines Description 190
14.4 Product Dependencies 190
14.5 Functional Description 191
14.6 User Interface 203
14.7 Module Configuration 250

15 Asynchronous Timer (AST) 251

15.1 Features 251
15.2 Overview 251
15.3 Block Diagram 252
15.4 Product Dependencies 252
15.5 Functional Description 253
15.6 User Interface 259
15.7 Module Configuration 280

16 Watchdog Timer (WDT) 281

16.1 Features 281
16.2 Overview 281

16.3 Block Diagram 281
16.4 Product Dependencies 281
16.5 Functional Description 282
16.6 User Interface 287
16.7 Module Configuration 293

17 External Interrupt Controller (EIC) 294

17.1 Features 294
17.2 Overview 294
17.3 Block Diagram 295
17.4 I/O Lines Description 295
17.5 Product Dependencies 295
17.6 Functional Description 296
17.7 User Interface 300
17.8 Module Configuration 316

18 Frequency Meter (FREQM) 317

18.1 Features 317
18.2 Overview 317
18.3 Block Diagram 317
18.4 Product Dependencies 317
18.5 Functional Description 318
18.6 User Interface 320
18.7 Module Configuration 331

19 General-Purpose Input/Output Controller (GPIO) 332

19.1 Features 332
19.2 Overview 332
19.3 Block Diagram 332
19.4 I/O Lines Description 333
19.5 Product Dependencies 333
19.6 Functional Description 334
19.7 User Interface 339
19.8 Module Configuration 362

20 Universal Synchronous Asynchronous Receiver Transmitter (USART)

20.1 Features 363
20.2 Overview 363
20.3 Block Diagram 364

20.4 I/O Lines Description 365
20.5 Product Dependencies 366
20.6 Functional Description 367
20.7 User Interface 408
20.8 Module Configuration 431

21 Serial Peripheral Interface (SPI) 432

21.1 Features 432
21.2 Overview 432
21.3 Block Diagram 433
21.4 Application Block Diagram 433
21.5 I/O Lines Description 434
21.6 Product Dependencies 434
21.7 Functional Description 434
21.8 User Interface 445
21.9 Module Configuration 471

22 Two-Wire Master Interface (TWIM) 472

22.1 Features 472
22.2 Overview 472
22.3 List of Abbreviations 473
22.4 Block Diagram 473
22.5 Application Block Diagram 474
22.6 I/O Lines Description 474
22.7 Product Dependencies 474
22.8 Functional Description 476
22.9 User Interface 488
22.10 Module Configuration 505

23 Two-Wire Slave Interface (TWIS) 506

23.1 Features 506
23.2 Overview 506
23.3 List of Abbreviations 507
23.4 Block Diagram 507
23.5 Application Block Diagram 508
23.6 I/O Lines Description 508
23.7 Product Dependencies 508
23.8 Functional Description 509

23.9 User Interface 518
23.10 Module Configuration 535

24 Pulse Width Modulation Controller (PWMA) 536

24.1 Features 536
24.2 Overview 536
24.3 Block Diagram 537
24.4 I/O Lines Description 537
24.5 Product Dependencies 537
24.6 Functional Description 538
24.7 User Interface 543
24.8 Module Configuration 557

25 Timer/Counter (TC) 558

25.1 Features 558
25.2 Overview 558
25.3 Block Diagram 559
25.4 I/O Lines Description 559
25.5 Product Dependencies 559
25.6 Functional Description 560
25.7 User Interface 575
25.8 Module Configuration 595

26 ADC Interface (ADCIFB) 596

26.1 Features 596
26.2 Overview 596
26.3 Block Diagram 597
26.4 I/O Lines Description 598
26.5 Product Dependencies 598
26.6 Functional Description 599
26.7 Resistive Touch Screen 604
26.8 Operating Modes 609
26.9 User Interface 611
26.10 Module Configuration 629

27 Analog Comparator Interface (ACIFB) 630

27.1 Features 630
27.2 Overview 630
27.3 Block Diagram 631

27.4 I/O Lines Description 631
27.5 Product Dependencies 631
27.6 Functional Description 632
27.7 Peripheral Event Triggers 637
27.8 AC Test mode 637
27.9 User Interface 638
27.10 Module Configuration 652

28 Capacitive Touch Module (CAT) 653

28.1 Features 653
28.2 Overview 653
28.3 Block Diagram 654
28.4 I/O Lines Description 655
28.5 Product Dependencies 655
28.6 Functional Description 657
28.7 User Interface 663
28.8 Module Configuration 694

29 Glue Logic Controller (GLOC) 695

29.1 Features 695
29.2 Overview 695
29.3 Block Diagram 695
29.4 I/O Lines Description 696
29.5 Product Dependencies 696
29.6 Functional Description 696
29.7 User Interface 698
29.8 Module Configuration 703

30 aWire UART (AW) 704

30.1 Features 704
30.2 Overview 704
30.3 Block Diagram 704
30.4 I/O Lines Description 705
30.5 Product Dependencies 705
30.6 Functional Description 705
30.7 User Interface 708
30.8 Module Configuration 721

31 Programming and Debugging 722

31.1 Overview 722
31.2 Service Access Bus 722
31.3 On-Chip Debug 725
31.4 JTAG and Boundary-Scan (JTAG) 734
31.5 JTAG Instruction Summary 742
31.6 aWire Debug Interface (AW) 759

32 Electrical Characteristics 776

32.1 Disclaimer 776
32.2 Absolute Maximum Ratings* 776
32.3 Supply Characteristics 776
32.4 Maximum Clock Frequencies 777
32.5 Power Consumption 777
32.6 I/O Pad Characteristics 779
32.7 Oscillator Characteristics 781
32.8 Flash Characteristics 784
32.9 Analog Characteristics 785
32.10 Timing Characteristics 789

33 Mechanical Characteristics 790

33.1 Thermal Considerations 790
33.2 Package Drawings 791
33.3 Soldering Profile 794

34 Ordering Information 795

35 Errata 796

35.1 Rev. E 796
35.2 Rev. D 800
35.3 Rev. C 804
35.4 Rev. B 804

36 Datasheet Revision History 819

36.1 Rev. D - 06/2010 819
36.2 Rev. C - 06/2010 819
36.3 Rev. B - 05/2010 819
36.4 Rev. A – 06/2009 820

Table of Contents i

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

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2010 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, AVR® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Общество с ограниченной ответственностью «МосЧип» ИНН 7719860671 / КПП 771901001
Адрес: 105318, г.Москва, ул.Щербаковская д.3, офис 1107

Данный компонент на территории Российской Федерации

Вы можете приобрести в компании MosChip.

Для оперативного оформления запроса Вам необходимо перейти по данной ссылке:

http://moschip.ru/get-element

Вы можете разместить у нас заказ для любого Вашего проекта, будь то серийное производство или разработка единичного прибора.

В нашем ассортименте представлены ведущие мировые производители активных и пассивных электронных компонентов.

Нашей специализацией является поставка электронной компонентной базы двойного назначения, продукции таких производителей как XILINX, Intel (ex.ALTERA), Vicor, Microchip, Texas Instruments, Analog Devices, Mini-Circuits, Amphenol, Glenair.

Сотрудничество с глобальными дистрибьюторами электронных компонентов, предоставляет возможность заказывать и получать с международных складов практически любой перечень компонентов в оптимальные для Вас сроки.

На всех этапах разработки и производства наши партнеры могут получить квалифицированную поддержку опытных инженеров.

Система менеджмента качества компании отвечает требованиям в соответствии с ГОСТ Р ИСО 9001, ГОСТ РВ 0015-002 и ЭС РД 009

Офис по работе с юридическими лицами:

105318, г.Москва, ул.Щербаковская д.3, офис 1107, 1118, ДЦ «Щербаковский»

Телефон: +7 495 668-12-70 (многоканальный)

Факс: +7 495 668-12-70 (доб.304)

E-mail: info@moschip.ru

Skype отдела продаж:
moschip.ru
moschip.ru_4
moschip.ru_6
moschip.ru_9