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 61 DMIPS Running at 48 MHz from Flash (1 Flash Wait State)
    - Up to 34 DMIPS Running at 24 MHz from Flash (0 Flash Wait State)
- Multi-Hierarchy Bus System
  - High-Performance Data Transfers on Separate Buses for Increased Performance
  - 7 Peripheral DMA Channels Improve Speed for Peripheral Communication
- Internal High-Speed Flash
  - 128 Kbytes, and 64 Kbytes Versions
  - Single-Cycle Access up to 24 MHz
  - Prefetch Buffer Optimizing Instruction Execution at Maximum Speed
  - 4 ms Page Programming Time and 8 ms Full-Chip Erase Time
  - 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
  - 16 Kbytes
- Interrupt Controller (INTC)
  - Autovectored Low Latency Interrupt Service with Programmable Priority
- External Interrupt Controller (EIC)
- System Functions
  - Power and Clock Manager
  - SleepWalking™ Power Saving Control
  - Internal System RC Oscillator (RCSYS)
  - 32 KHz Oscillator
  - Clock Failure Detection
  - One Multipurpose Oscillator and two Phase Locked Loop (PLL)
- 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
- Universal Serial Bus (USB)
  - Device 2.0 full speed and low speed
  - Flexible End-Point Configuration and Management
  - On-chip Transceivers Including Pull-Ups
- Three 16-bit Timer/Counter (TC) Channels
  - External Clock Inputs, PWM, Capture and Various Counting Capabilities
- 7 PWM Channels (PWMA)
  - 12-bit PWM up to 150 MHz Source Clock
- Three 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

32-bit AVR® Microcontroller

ATUC128D3
ATUC64D3
ATUC128D4
ATUC64D4

32133D–11/2011

- One Master and One Slave Two-Wire Interfaces (TWI), 400 kbit/s I^2C-compatible
- One 8-channel Analog-To-Digital Converter (ADC)
- One Inter-IC Sound Controller (IISC) with Stereo Capabilities
- Autonomous Capacitive Touch Button (QTouch®) Capture
  - Up to 25 Touch Buttons
  - QWheel® and QSlide® Compatible
- QTouch® Library Support
  - Capacitive Touch Buttons, Sliders, and Wheels
  - QTouch® and QMatrix® Acquisition
  - Hardware assisted QTouch® Acquisition
- One Programmable Glue Logic Controller(GLOC) for General Purpose PCB Design
- On-Chip Non-Intrusive Debug System
  - Nexus Class 2+, Runtime Control
  - aWire™ Single-Pin Programming and Debug Interface Muxed with Reset Pin
  - 64-pin and 48-pin TQFP/QFN (51 and 35 GPIO Pins)
- Four High-Drive I/O Pins
- Single 3.3V Power Supply or Dual 1.8V-3.3V Power Supply

1. Description

The UC3D is a complete System-On-Chip microcontroller based on the AVR32UC RISC processor running at frequencies up to 48MHz. AVR32UC is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density, and high performance.

The processor implements a fast and flexible interrupt controller for supporting modern operating systems and real-time operating systems.

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

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 Power Manager improves design flexibility and security. Power monitoring is supported by on-chip Power-On Reset (POR), and Brown-Out Detector (BOD). The device features several oscillators, such as Oscillator 0 (OSC0), 32 KHz Oscillator and system RC oscillator (RCSYS), and two Phase Lock Loop (PLL). Either of these oscillators/PLLs can be used as source for the system clock.

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 32KHz crystal oscillator can operate in a 1- or 2-pin mode, trading pin usage and accuracy.

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

The Full-Speed USB 2.0 Device interface supports several USB Classes at the same time thanks to the rich End-Point configuration.

The device includes three 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 12-bit PWM channels which can be synchronized and controlled from a common timer. Seven PWM channels are available, 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 UC3D also features many communication interfaces for communication intensive applications. In addition to standard serial interfaces like USART, SPI or TWI, USB is available. The USART supports different communication modes, like SPI mode.

A general purpose 8-channel ADC is provided; It features a fully configurable sequencer that handles many conversions. Window Mode allows each ADC channel to be used like a simple Analog Comparator.

The Inter-IC Sound controller (IISC) provides easy access to digital audio interfaces following I2S stereo standard.

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 25 touch sensors. One touch sensor can be configured to operate autonomously without software interaction,allowing wakeup from sleep modes when activated.

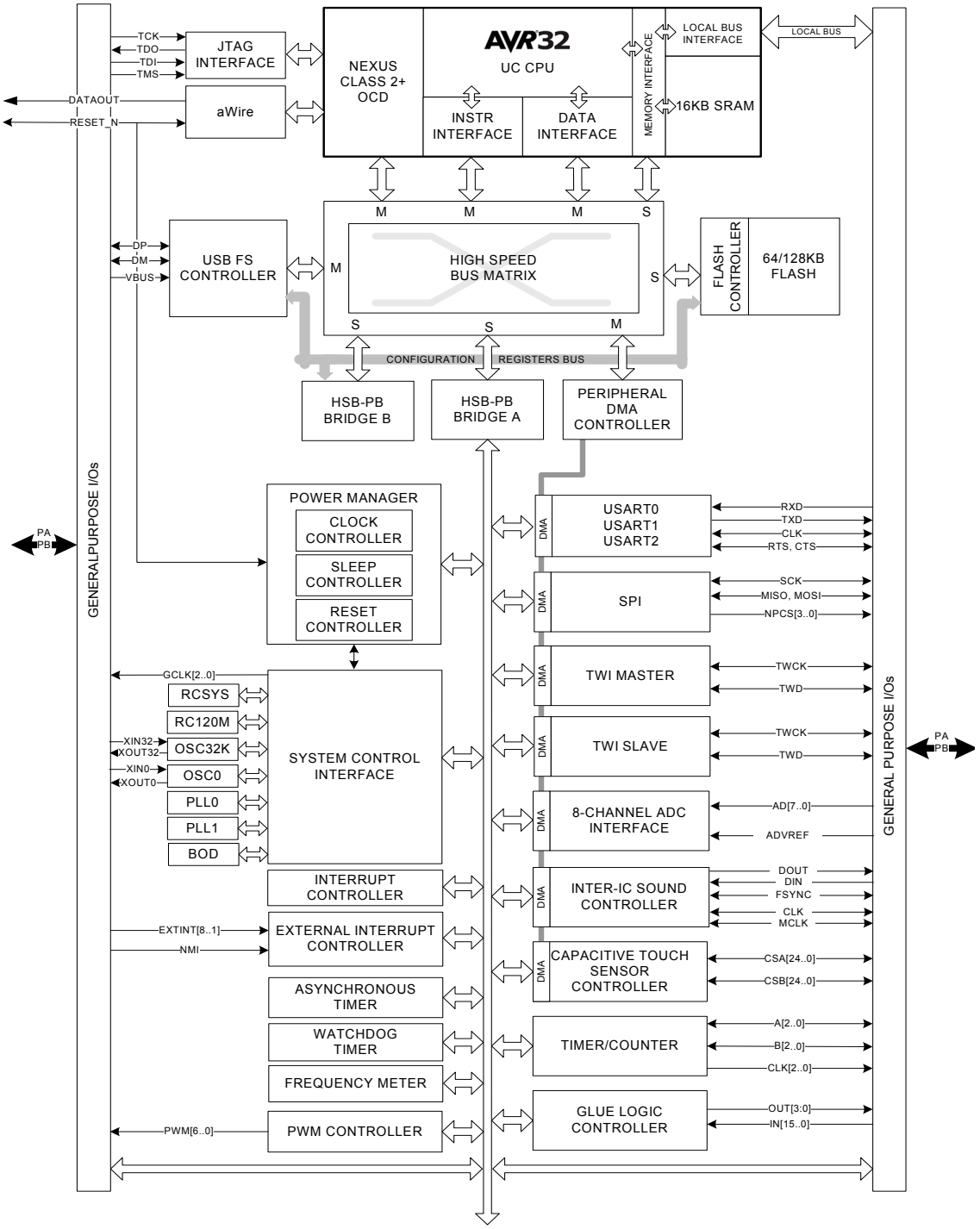
Atmel also offers the QTouch library for embedding capacitive touch buttons, sliders, and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and included fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS®) 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 UC3D integrates a class 2+ Nexus 2.0 On-Chip Debug (OCD) System, with full-speed read/write memory access, in addition to basic runtime control. 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	ATUC128/64D3	ATUC128/64D4
Flash	128/64 KB	128/64KB
SRAM	16 KB	16 KB
Package	TQFP64, QFN64	TQFP48, QFN48
GPIO	51	35
FS USB Device	1	
Hi-drive pins	4	
External Interrupts	9	7
TWI Master/Slave	1/1	
USART	3	
Peripheral DMA Channels	7	
SPI	1	
Asynchronous Timers	1	
Timer/Counter Channels	3	
PWM channels	7	
Inter-IC Sound	1	
Frequency Meter	1	
Watchdog Timer	1	
Power Manager	1	
Oscillators	2x Phase Locked Loop 80-240 MHz (PLL) 1x Crystal Oscillator 0.4-20 MHz (OSC0) 1x Crystal Oscillator 32 KHz (OSC32K) 1x RC Oscillator 120 MHz (RC120M) 1x RC Oscillator 115 kHz (RCSYS)	
10-bit ADC channels	8	6
Capacitive Touch Sensor supported	25	17
Glue Logic Control Inputs/Outputs	16/4	14/4
JTAG	1	
aWire	1	
Max Frequency	48 MHz	

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

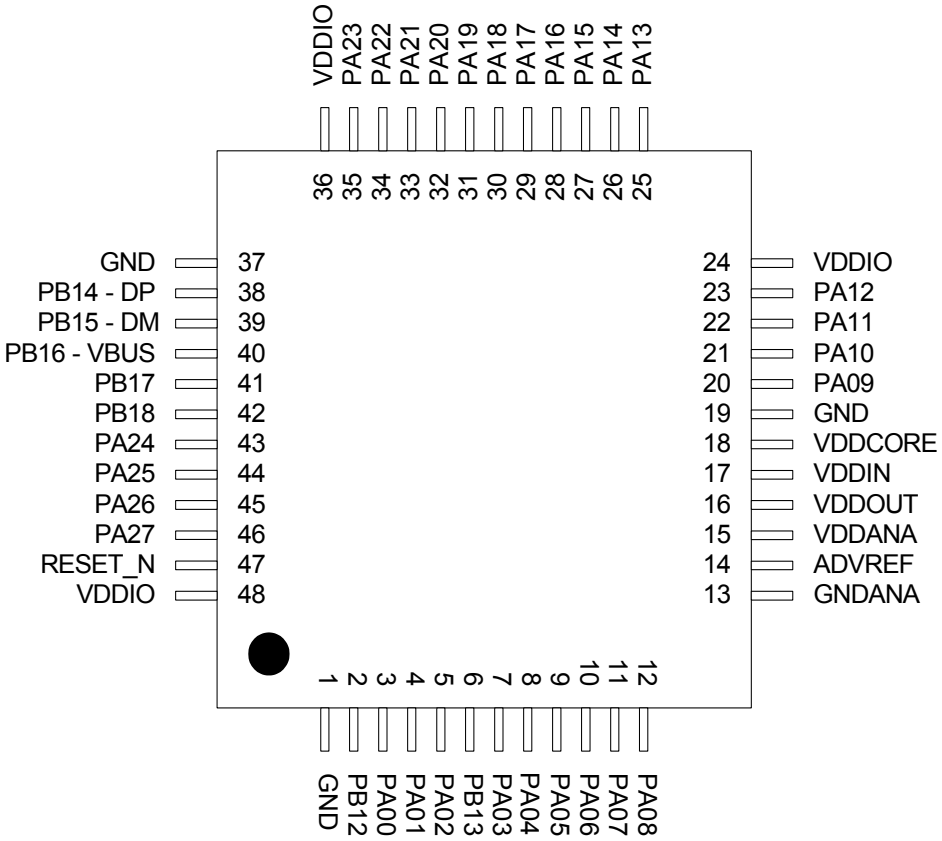
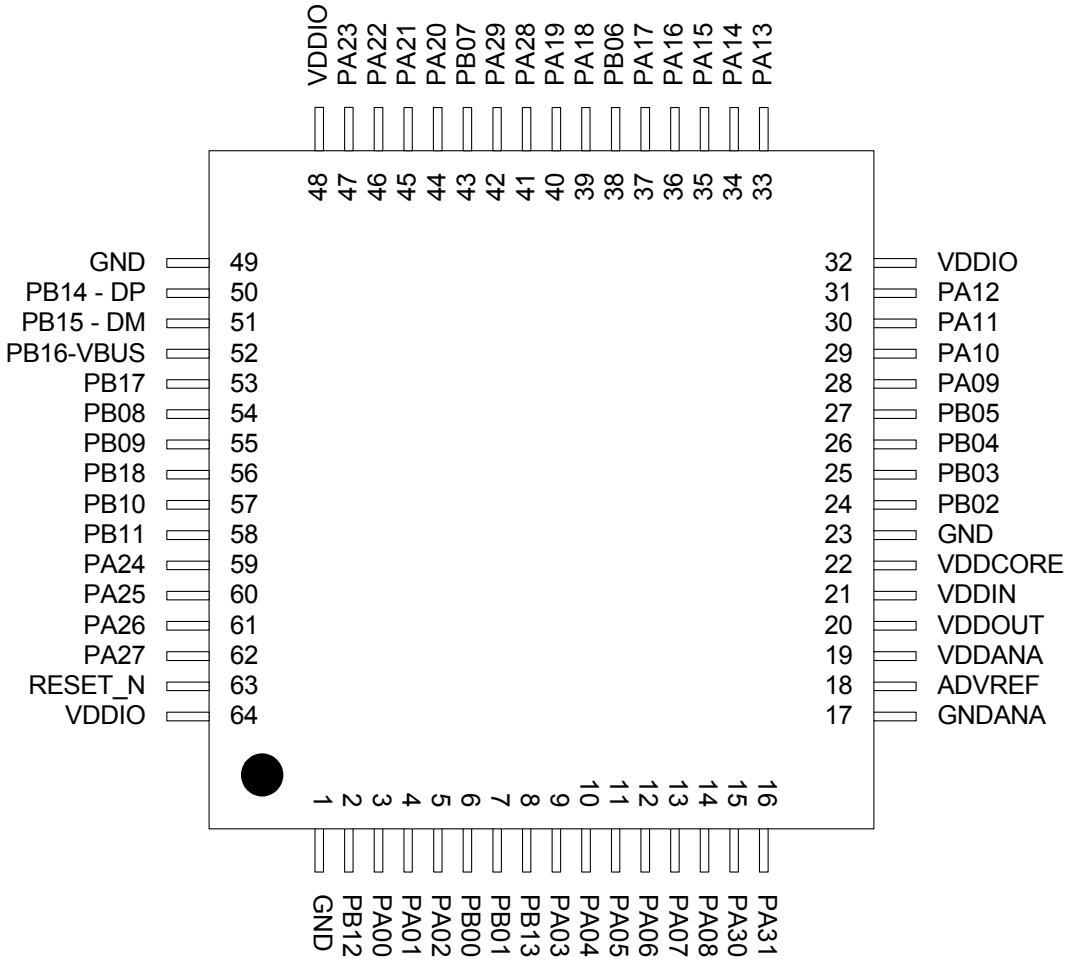


Figure 3-2. TQFP64/QFN64 Pinout



Note: On QFN packages, the exposed pad is not connected to anything internally, but should be soldered to ground to increase board level reliability.

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. Multiplexed Signals on I/O Pins

48-pin Package	64-pin Package	PIN	GPIO	Supply	Pad Type	GPIO Function				Other Functions
						A	B	C	D	
3	3	PA00	0	VDDIO	Normal I/O	SPI - MISO	PWMA - PWMA[1]	GLOC - IN[0]	CAT - CSB[0]	JTAG-TDI
4	4	PA01	1	VDDIO	Normal I/O	SPI - MOSI	PWMA - PWMA[2]	GLOC - IN[1]	CAT - CSA[1]	JTAG-TDO
5	5	PA02	2	VDDIO	Normal I/O	SPI - SCK	PWMA - PWMA[3]	GLOC - IN[2]	CAT - CSB[1]	JTAG-TMS
7	9	PA03	3	VDDANA	Analog I/O	PKGANA - ADCIN0	SCIF - GCLK[0]	GLOC - IN[5]	CAT - CSB[2]	
8	10	PA04	4	VDDANA	Analog I/O	PKGANA - ADCIN1	SCIF - GCLK[1]	GLOC - IN[6]	CAT - CSA[3]	

Table 3-1. Multiplexed Signals on I/O Pins

48-pin Package	64-pin Package	PIN	GPIO	Supply	Pad Type	GPIO Function				Other Functions
						A	B	C	D	
9	11	PA05	5	VDDANA	Analog I/O	EIC - EXTINT[8]	PKGANA - ADCIN2	GLOC - OUT[1]	CAT - CSB[3]	
10	12	PA06	6	VDDANA	Analog I/O	EIC - EXTINT[1]	PKGANA - ADCIN3	GLOC - IN[7]	CAT - CSA[4]	
11	13	PA07	7	VDDANA	Analog I/O	PWMA - PWMA[0]	PKGANA - ADCIN4	GLOC - IN[8]	CAT - CSB[4]	
12	14	PA08	8	VDDANA	Analog I/O	PWMA - PWMA[1]	PKGANA - ADCIN5	GLOC - IN[9]	CAT - CSA[5]	
20	28	PA09	9	VDDIO	Normal I/O, 5V tolerant	TWIMS - TWCK	SPI - NPCS[2]	USART1 - CTS	CAT - CSB[5]	
21	29	PA10	10	VDDIO	Normal I/O, 5V tolerant	TWIMS - TWD	SPI - NPCS[3]	USART1 - RTS	CAT - CSA[6]	
22	30	PA11	11	VDDIO	Normal I/O	USART0 - RTS	TC - A2	PWMA - PWMA[0]	CAT - CSB[6]	OSC32 - XIN
23	31	PA12	12	VDDIO	Normal I/O	USART0 - CTS	TC - B2	PWMA - PWMA[1]	CAT - CSA[7]	OSC32 - XOUT
25	33	PA13	13	VDDIO	Normal I/O	EIC - EXTINT[0]	PWMA - PWMA[2]	USART0 - CLK	CAT - CSB[7]	
26	34	PA14	14	VDDIO	Normal I/O	SPI - MOSI	PWMA - PWMA[3]	EIC - EXTINT[2]	CAT - CSA[8]	
27	35	PA15	15	VDDIO	Normal I/O	SPI - SCK	PWMA - PWMA[4]	USART2 - CLK	CAT - CSB[8]	
28	36	PA16	16	VDDIO	Normal I/O	SPI - NPCS[0]	TC - CLK1	PWMA - PWMA[4]	CAT - CSA[9]	
29	37	PA17	17	VDDIO	Normal I/O	SPI - NPCS[1]	TC - CLK2	SPI - SCK	CAT - CSB[9]	
30	39	PA18	18	VDDIO	Normal I/O	USART0 - RXD	PWMA - PWMA[5]	SPI - MISO	CAT - CSA[10]	OSC0 - XIN
31	40	PA19	19	VDDIO	Normal I/O	USART0 - TXD	PWMA - PWMA[6]	SPI - MOSI	CAT - CSB[10]	OSC0 - XOUT
32	44	PA20	20	VDDIO	Normal I/O	USART1 - CLK	TC - CLK0	USART2 - RXD	CAT - CSA[11]	
33	45	PA21	21	VDDIO	Normal I/O	PWMA - PWMA[2]	TC - A1	USART2 - TXD	CAT - CSB[11]	
34	46	PA22	22	VDDIO	Normal I/O	PWMA - PWMA[6]	TC - B1	ADCIFD - EXTTRIG	CAT - CSA[12]	
35	47	PA23	23	VDDIO	Normal I/O	USART1 - TXD	SPI - NPCS[1]	EIC - EXTINT[3]	CAT - CSB[12]	
43	59	PA24	24	VDDIO	Normal I/O	USART1 - RXD	SPI - NPCS[0]	EIC - EXTINT[4]	CAT - CSB[15]	
44	60	PA25	25	VDDIO	Normal I/O	SPI - MISO	PWMA - PWMA[3]	EIC - EXTINT[5]	CAT - CSA[16]	
45	61	PA26	26	VDDIO	Normal I/O	IISC - IWS	USART2 - TXD	TC - A0	CAT - CSB[16]	
46	62	PA27	27	VDDIO	Normal I/O	IISC - ISCK	USART2 - RXD	TC - B0	CAT - CSA[0]	
	41	PA28	28	VDDIO	Normal I/O	USART0 - CLK	PWMA - PWMA[4]	SPI - MISO	CAT - CSB[21]	
	42	PA29	29	VDDIO	Normal I/O	TC - CLK0	TC - CLK1	SPI - MOSI	CAT - CSA[22]	
	15	PA30	30	VDDANA	Analog I/O	PKGANA - ADCIN6	EIC - EXTINT[6]	SCIF - GCLK[2]	CAT - CSA[18]	
	16	PA31	31	VDDANA	Analog I/O	PKGANA - ADCIN7	EIC - EXTINT[7]	PWMA - PWMA[6]	CAT - CSB[18]	
	6	PB00	32	VDDIO	Normal I/O	TC - A0	EIC - EXTINT[4]	USART2 - CTS	CAT - CSA[17]	
	7	PB01	33	VDDIO	Normal I/O	TC - B0	EIC - EXTINT[5]	USART2 - RTS	CAT - CSB[17]	
	24	PB02	34	VDDIO	Normal I/O	EIC - EXTINT[6]	TC - A1	USART1 - TXD	CAT - CSA[19]	
	25	PB03	35	VDDIO	Normal I/O	EIC - EXTINT[7]	TC - B1	USART1 - RXD	CAT - CSB[19]	
	26	PB04	36	VDDIO	Normal I/O	USART1 - CTS	SPI - NPCS[3]	TC - CLK2	CAT - CSA[20]	
	27	PB05	37	VDDIO	Normal I/O	USART1 - RTS	SPI - NPCS[2]	PWMA - PWMA[5]	CAT - CSB[20]	
	38	PB06	38	VDDIO	Normal I/O	IISC - ISCK	PWMA - PWMA[5]	GLOC - IN[15]	CAT - CSA[21]	
	43	PB07	39	VDDIO	Normal I/O	IISC - ISDI	EIC - EXTINT[2]	GLOC - IN[11]	CAT - CSB[22]	
	54	PB08	40	VDDIO	Normal I/O	IISC - IWS	EIC - EXTINT[0]	GLOC - IN[14]	CAT - CSA[23]	
	55	PB09	41	VDDIO	Normal I/O	IISC - ISCK	IISC - IMCK	GLOC - IN[3]	CAT - CSB[23]	
	57	PB10	42	VDDIO	Normal I/O	IISC - ISDO	TC - A2	USART0 - RXD	CAT - CSA[24]	
	58	PB11	43	VDDIO	Normal I/O	IISC - IWS	TC - B2	USART0 - TXD	CAT - CSB[24]	

Table 3-1. Multiplexed Signals on I/O Pins

48-pin Package	64-pin Package	PIN	GPIO	Supply	Pad Type	GPIO Function				Other Functions
						A	B	C	D	
2	2	PB12	44	VDDIO	Normal I/O	SPI - NPCS[0]	IISC - IMCK	GLOC - OUT[0]		JTAG-TCK
6	8	PB13	45	VDDIO	Normal I/O	CAT - SYNC	SCIF - GCLK[2]	GLOC - IN[4]	CAT - CSA[2]	
38	50	PB14	46	VDDIO	Normal I/O	USBC - DP	USART0 - RXD	GLOC - OUT[2]	CAT - CSA[13]	
39	51	PB15	47	VDDIO	Normal I/O	USBC - DM	USART0 - TXD	GLOC - IN[12]	CAT - CSB[13]	
40	52	PB16	48	VDDIO	Input only, 5V tolerant	USBC - VBUS		GLOC - IN[10]		USB-VBUS
41	53	PB17	49	VDDIO	Normal I/O	IISC - ISDO	USART0 - RTS	GLOC - IN[13]		
42	56	PB18	50	VDDIO	Normal I/O	IISC - ISDI	CAT - SYNC	GLOC - OUT[3]	CAT - CSA[15]	

See Section 4. for a description of the various peripheral signals.

Refer to "Electrical Characteristics" on page 716 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

3.2.3 JTAG Port Connections

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

Table 3-3. JTAG Pinout

48-pin or 64-pin Package	Pin Name	JTAG Pin
2	PB12	TCK
5	PA02	TMS
4	PA01	TDO
3	PA00	TDI

3.2.4 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 3-4. Oscillator Pinout

48-pin Package	64-pin Package	Pin	Oscillator Function
30	39	PA18	XIN0
31	40	PA19	XOUT0
22	30	PA11	XIN32
23	31	PA12	XOUT32

3.2.5 Other Functions

The functions listed in Table 3-5 are not mapped to the normal GPIO functions.The aWire DATA pin will only be active after the aWire is enabled. The aWire DATAOUT pin will only be active after the aWire is enabled and the 2-pin mode command has been sent.

Table 3-5. Other Functions

48-Pin Package	64-Pin Package	Pin	Function
47	63	RESET_N	aWire DATA
2	2	PB12	aWire DATAOUT

4. Signal Descriptions

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

Table 4-1. Signal Descriptions List

Signal Name	Function	Type	Active Level	Comments
aWire - AW				
DATA	aWire data	I/O		
DATAOUT	aWire data output for 2-pin mode	I/O		
External Interrupt Controller - EIC				
NMI	Non-Maskable Interrupt	Input		
EXTINT8 - EXTINT1	External interrupt	Input		
JTAG module - JTAG				
TCK	Test Clock	Input		
TDI	Test Data In	Input		
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		
Power Manager - PM				
RESET_N	Reset	Input	Low	
Basic Pulse Width Modulation Controller - PWMA				
PWMA6 - PWMA0	PWMA channel waveforms	Output		
System Control Interface - SCIF				
GCLK2 - GCLK0	Generic clock	Output		
XIN0	Oscillator 0 XIN Pin	Analog		
XOUT0	Oscillator 0 XOUT Pin	Analog		
XIN32	32K Oscillator XIN Pin	Analog		
XOUT32	32K Oscillator XOUT Pin	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 - TC				
A0	Channel 0 Line A	I/O		
A1	Channel 1 Line A	I/O		

Table 4-1. Signal Descriptions List

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 Master- TWIM				
TWCK	Two-wire Serial Clock			
TWD	Two-wire Serial Data			
Two Wire Interface Slave- TWIS				
TWCK	Two-wire Serial Clock			
TWD	Two-wire Serial Data			
Universal Synchronous/Asynchronous Receiver/Transmitter - USART0/1/2				
CLK	Clock	I/O		
CTS	Clear To Send	Input	Low	
RTS	Request To Send	Output	Low	
RXD	Receive Data	Input		
TXD	Transmit Data	Output		
Universal Serial Bus 2.0 Full Speed Interface - USBC				
DM	DM for USB FS			
DP	DP for USB FS			
VBUS	VBUS			
IIS Controller - IISC				
IBCK	IIS Serial Clock	I/O		
ISDI	IIS Serial Data In	Input		
ISDO	IIS Serial Data Out	Output		
IWS	IIS Word Select	I/O		
IMCK	IIS Master Clock	Output		
Capacitive Touch Sensor - CAT				
CSA24 - CSA0	Capacitive Sensor Group A	I/O		
CSB24 - CSB0	Capacitive Sensor Group B	I/O		
SYNC	Synchronize signal	Input		
Glue Logic Controller - GLOC				
IN15 - IN0	Inputs to lookup tables	Input		
OUT3 - OUT0	Outputs from lookup tables	Output		
ADC controller interface - ADCIFD				

Table 4-1. Signal Descriptions List

EXTTRIG	ADCIFD EXTTRIG	Input		
AD7 - AD0	ADC Inputs	Analog		
Power				
VDDIO	Digital I/O Power Supply	Power Input		3.0 V to 3.6V.
VDDANA	Analog Power Supply	Power Input		3.0 V to 3.6V
ADVREF	Analog Reference Voltage	Power Input		2.6 V to 3.6 V
VDDCORE	Core Power Supply	Power Input		1.65 V to 1.95 V
VDDIN	Voltage Regulator Input	Power Input		3.0 V to 3.6V
VDDOUT	Voltage Regulator Output	Power Output		1.65 V to 1.95V
GNDANA	Analog Ground	Ground		
GND	Ground	Ground		
General Purpose I/O pin - GPIOA, GPIOB				
PA31 - PA00	General Purpose I/O Controller GPIO A	I/O		
PB18 - PB00	General Purpose I/O Controller GPIO B	I/O		

4.1 I/O Line Considerations

4.1.1 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. TDO pin is an output, driven at VDDIO, and has no pull-up resistor. These JTAG pins can be used as GPIO pins and muxed with peripherals when the JTAG is disabled.

4.1.2 RESET_N Pin

The RESET_N pin is a schmitt input and integrates a programmable pull-up resistor to VDDIO. 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 the application.

4.1.3 TWI Pins

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

4.1.4 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 Controller. After reset, I/O lines default as inputs with pull-up resistors disabled.

4.1.5 High drive pins

Four I/O lines can be used to drive twice current than other I/O capability (see Electrical Characteristics section).

48-pin Package	64-pin Package	Pin Name
32	44	PA20
33	45	PA21
34	46	PA22
35	47	PA23

4.2 Power Considerations

4.2.1 Power Supplies

The UC3D has several types of power supply pins:

- VDDIO: Powers Digital I/O lines. Voltage is 3.3V nominal.
- VDDIN: Powers the internal regulator. Voltage is 3.3V nominal.
- VDDCORE : Powers the internal core digital logic. Voltage is 1.8 V nominal.
- VDDANA: Powers the ADC and Analog I/O lines. Voltage is 3.3V nominal.

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

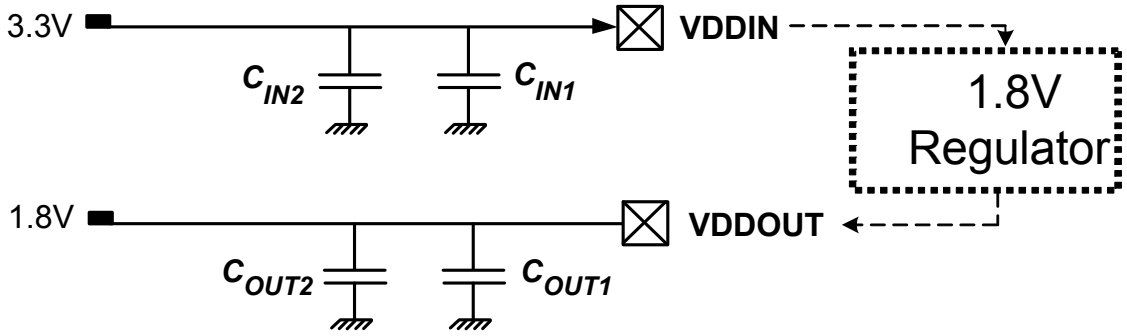
Refer to ”Electrical Characteristics” on page 716 for power consumption on the various supply pins.

4.2.2 Voltage Regulator

The UC3D embeds a voltage regulator that converts from 3.3V nominal to 1.8V with a load of up to 100 mA. The regulator is intended to supply the logic, memories, oscillators and PLLs. See Section 4.2.3 for regulator connection figures.

Adequate output supply decoupling is mandatory on VDDOUT to reduce ripple and avoid oscillations. The best way to achieve this is to use two capacitors in parallell between VDDOUT and GND as close to the chip as possible. Please refer to Section 32.9.1 for decoupling capacitors values and regulator characteristics. VDDOUT can be connected externally to the 1.8V domains to power external components.

Figure 4-1. Supply Decoupling



For decoupling recommendations for VDDIO, VDDANA and VDDCORE, please refer to the Schematic checklist.

4.2.3 Regulator Connection

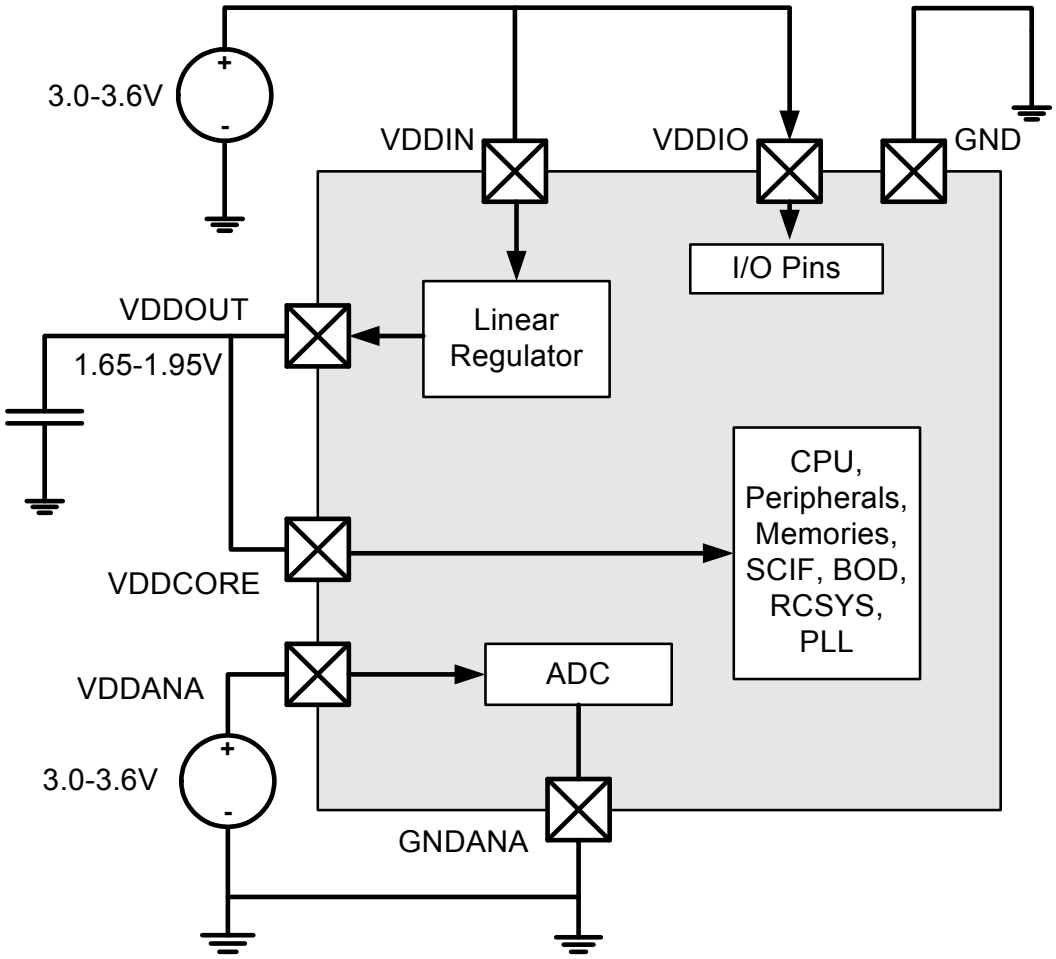
The UC3D supports two power supply configurations:

- 3.3V single supply mode
- 3.3V - 1.8V dual supply mode

4.2.3.1 3.3V Single Supply Mode

In 3.3V single supply mode the internal regulator is connected to the 3.3V source (VDDIN pin). The regulator output (VDDOUT) needs to be externally connected to VDDCORE pin to supply internal logic. Figure 4-2 shows the power schematics to be used for 3.3V single supply mode.

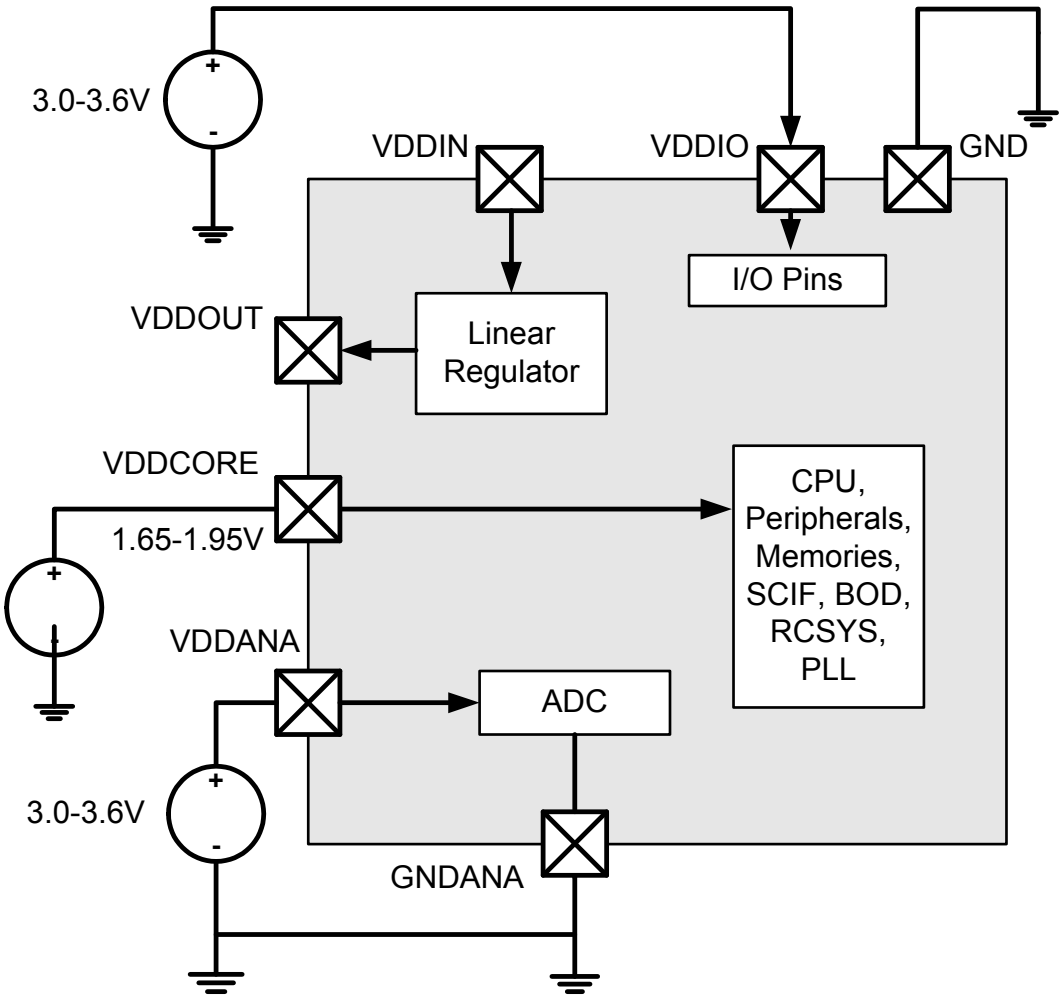
Figure 4-2. 3.3V Single Power Supply mode



4.2.3.2 *3.3V + 1.8V Dual Supply Mode*

In dual supply mode the internal regulator is not used (unconnected), VDDIO is powered by 3.3V supply and VDDCORE is powered by a 1.8V supply as shown in Figure 4-3.

Figure 4-3. 3.3V + 1.8V Dual Power Supply Mode.



4.2.4 Power-up Sequence

4.2.4.1 *Maximum Rise Rate*

To avoid risk of latch-up, the rise rate of the power supplies must not exceed the values described in Supply Characteristics table in the Electrical Characteristics chapter.

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

4.2.4.2 *Minimum Rise Rate*

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

See Supply Characteristics table in the Electrical Characteristics chapter 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 RESET_N until VDDIN rises above 1.2V.

5. Processor and Architecture

Rev: 2.1.2.0

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

5.1 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**

5.2 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.

5.3 The AVR32UC CPU

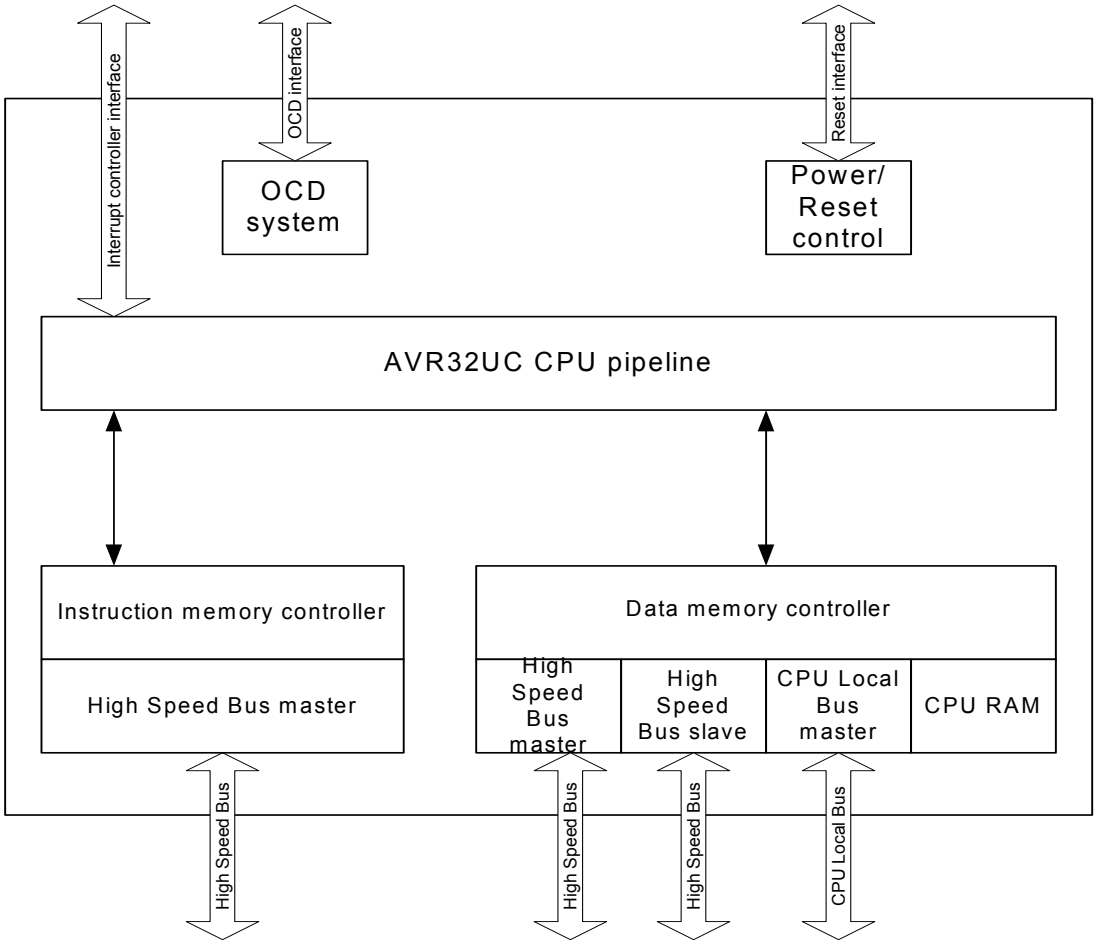
The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced On-Chip Debug (OCD) system, and no caches. Java acceleration hardware is not implemented.

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 5-1 on page 22 displays the contents of AVR32UC.

Figure 5-1. Overview of the AVR32UC CPU



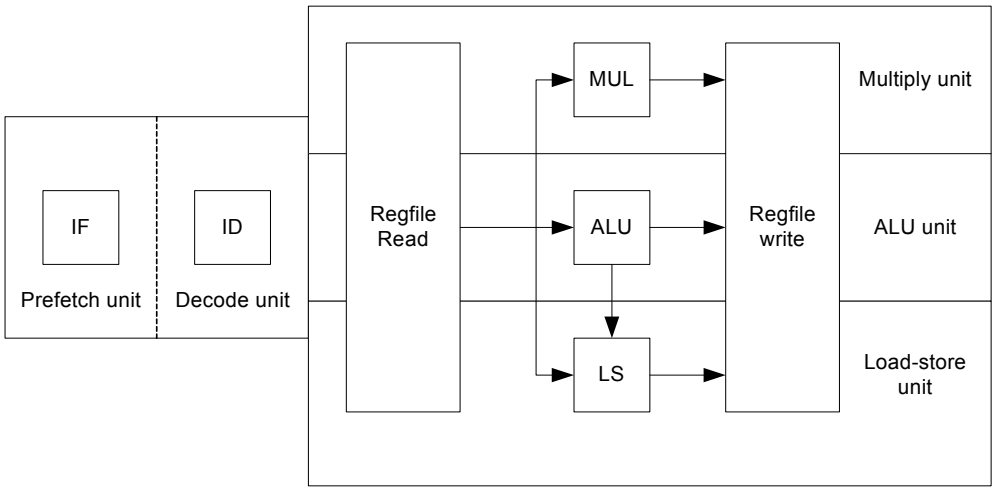
5.3.1 Pipeline Overview

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 5-2 on page 23 shows an overview of the AVR32UC pipeline stages.

Figure 5-2. The AVR32UC Pipeline



5.3.2 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.

5.3.2.1 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.

5.3.2.2 Java Support

AVR32UC does not provide Java hardware acceleration.

5.3.2.3 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 5-1. Instructions with Unaligned Reference Support

Instruction	Supported Alignment
ld.d	Word
st.d	Word

5.3.2.4 *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

5.3.2.5 *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.

5.4 Programming Model

5.4.1 Register File Configuration

The AVR32UC register file is shown below.

Figure 5-3. 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

5.4.2 Status Register Configuration

The Status Register (SR) is split into two halfwords, one upper and one lower, see Figure 5-4 and Figure 5-5. 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 5-4. The Status Register High Halfword

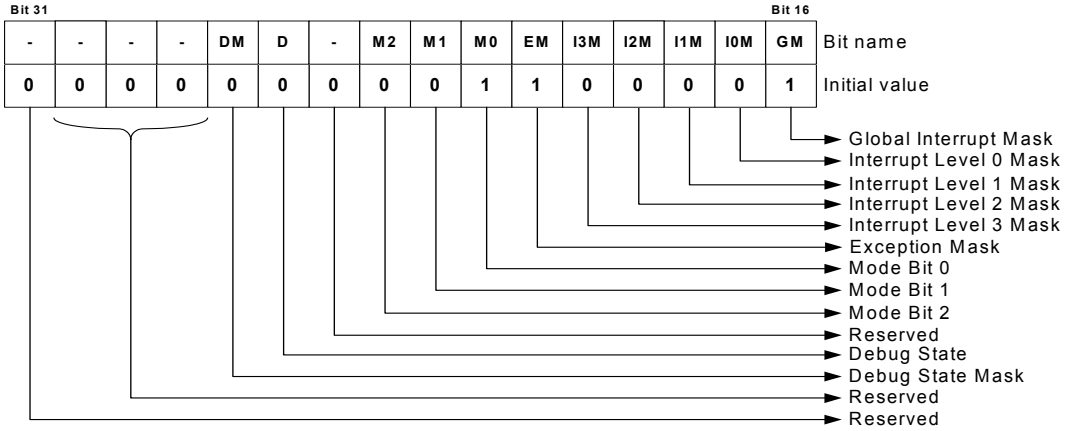
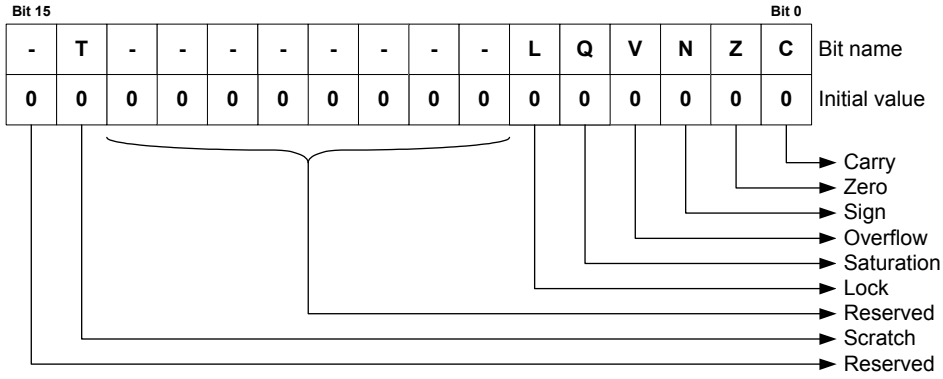


Figure 5-5. The Status Register Low Halfword



5.4.3 Processor States

5.4.3.1 Normal RISC State

The AVR32 processor supports several different execution contexts as shown in Table 5-2.

Table 5-2. 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.

5.4.3.2 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.

5.4.4 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 5-3. 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
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

Table 5-3. System Registers (Continued)

Reg #	Address	Name	Function
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
90-102	360-408	Reserved	Reserved for future use
103-111	412-444	Reserved	Reserved for future use
112-191	448-764	Reserved	Reserved for future use
192-255	768-1020	IMPL	IMPLEMENTATION DEFINED

5.5 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 5-4 on page 32. 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.

5.5.1 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.

5.5.2 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 5-4 on page 32, 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.

5.5.3 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.

5.5.4 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.

5.5.5 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.

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.

If several events occur on the same instruction, they are handled in a prioritized way. The priority ordering is presented in Table 5-4 on page 32. 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 5-4 on page 32. Some of the exceptions are unused in AVR32UC since it has no MMU, coprocessor interface, or floating-point unit.

Table 5-4. 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			
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			
14	EVBA+0x18			
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			
25	EVBA+0x70			
26	EVBA+0x3C			
27	EVBA+0x40			
28	EVBA+0x44			

6. Memories

6.1 Embedded Memories

- **Internal High-Speed Flash**
  - **128Kbytes (ATUC128D)**
  - **64Kbytes (ATUC64D)**
    - **0 Wait State Access at up to 24 MHz in Worst Case Conditions**
    - **1 Wait State Access at up to 48 MHz in Worst Case Conditions**
    - **Pipelined Flash Architecture, allowing burst reads from sequential Flash locations, hiding penalty of 1 wait state access**
    - **100 000 Write Cycles, 15-year Data Retention Capability**
    - **4ms Page Programming Time, 8 ms Chip Erase Time**
    - **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**
  - **16Kbytes**

6.2 Physical Memory Map

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

Table 6-1. UC3D Physical Memory Map

Device		Embedded SRAM	Embedded Flash	HSB-PB Bridge A	HSB-PB Bridge B
Start Address		0x0000_0000	0x8000_0000	0xFFFF_0000	0xFFFE_0000
Size	ATUC128D	16 Kbytes	128 Kbytes	64 Kbytes	64 Kbytes
	ATUC64D	16 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes

6.3 Peripheral Address Map

Table 6-2. Peripheral Address Mapping

Address		Peripheral Name
0xFFFE0000	USBC	USB 2.0 Interface - USBC
0xFFFE1000	HMATRIX	HSB Matrix - HMATRIX
0xFFFE1400	FLASHCDW	Flash Controller - FLASHCDW
0xFFFF0000	PDCA	Peripheral DMA Controller - PDCA
0xFFFF1000	INTC	Interrupt controller - INTC

Table 6-2. Peripheral Address Mapping

0xFFFF1400	PM	Power Manager - PM
0xFFFF1800	AST	Asynchronous Timer - AST
0xFFFF1C00	WDT	Watchdog Timer - WDT
0xFFFF2000	EIC	External Interrupt Controller - EIC
0xFFFF2800	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	SPI	Serial Peripheral Interface - SPI
0xFFFF4000	TWIM	Two-wire Master Interface - TWIM
0xFFFF4400	TWIS	Two-wire Slave Interface - TWIS
0xFFFF4800	PWMA	Pulse Width Modulation Controller - PWMA
0xFFFF4C00	IISC	Inter-IC Sound (I2S) Controller - IISC
0xFFFF5000	TC	Timer/Counter - TC
0xFFFF5400	ADCIFD	ADC controller interface - ADCIFD
0xFFFF5800	SCIF	System Control Interface - SCIF
0xFFFF5C00	FREQM	Frequency Meter - FREQM
0xFFFF6000	CAT	Capacitive Touch Module - CAT

Table 6-2. Peripheral Address Mapping

0xFFFF6400	GLOC	Glue Logic Controller - GLOC
0xFFFF6800	AW	aWire - AW

6.4 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 6-3. Local Bus Mapped GPIO Registers

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

7. Boot Sequence

This chapter summarizes the boot sequence of the UC3D. The behavior after power-up is controlled by the Power Manager. For specific details, refer to the Power Manager chapter.

7.1 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.

On system start-up, 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.

7.2 Fetching of Initial Instructions

After reset has been released, the AVR32UC 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 divide the frequency of the clock routed to some of the peripherals, and to gate the clocks to unused peripherals.

8. Flash Controller (FLASHCDW)

Rev: 1.2.0.0

8.1 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**
- **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**

8.2 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.

8.3 Product Dependencies

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

8.3.1 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.

8.3.2 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.

8.3.3 Interrupts

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

8.3.4 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.

8.4 Functional Description

8.4.1 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.

8.4.2 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

8.4.3 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 8-1 on page 40.

8.4.4 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.

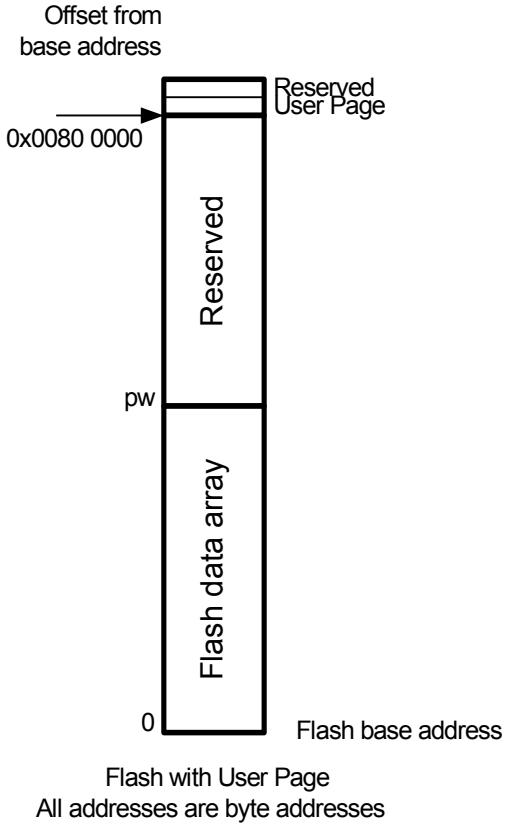
The Flash Controller provides two sets of buffers that can be enabled in order to speed up instruction fetching. These buffers can be enabled by writing a one to the FCR.SEQBUF and FCR.BRBUF bits. The SEQBUF bit enables buffering hardware optimizing sequential instruction fetches. The BRBUF bit enables 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 8-1. 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 8-1. User Page Addresses

Memory type	Start address, byte sized	Size
Main array	0	*pw* bytes
User	0x00800000	w bytes

Figure 8-1. Memory Map for the Flash Memories



8.4.5 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.

8.4.6 Quick User Page Read

A dedicated command, Quick User Page Read (QPRUP), is provided to read all words in the user 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 QPRUP command is useful to check that a page is in an erased state. The QPRUP instruction is much faster than performing the erased-page check using a regular software subroutine.

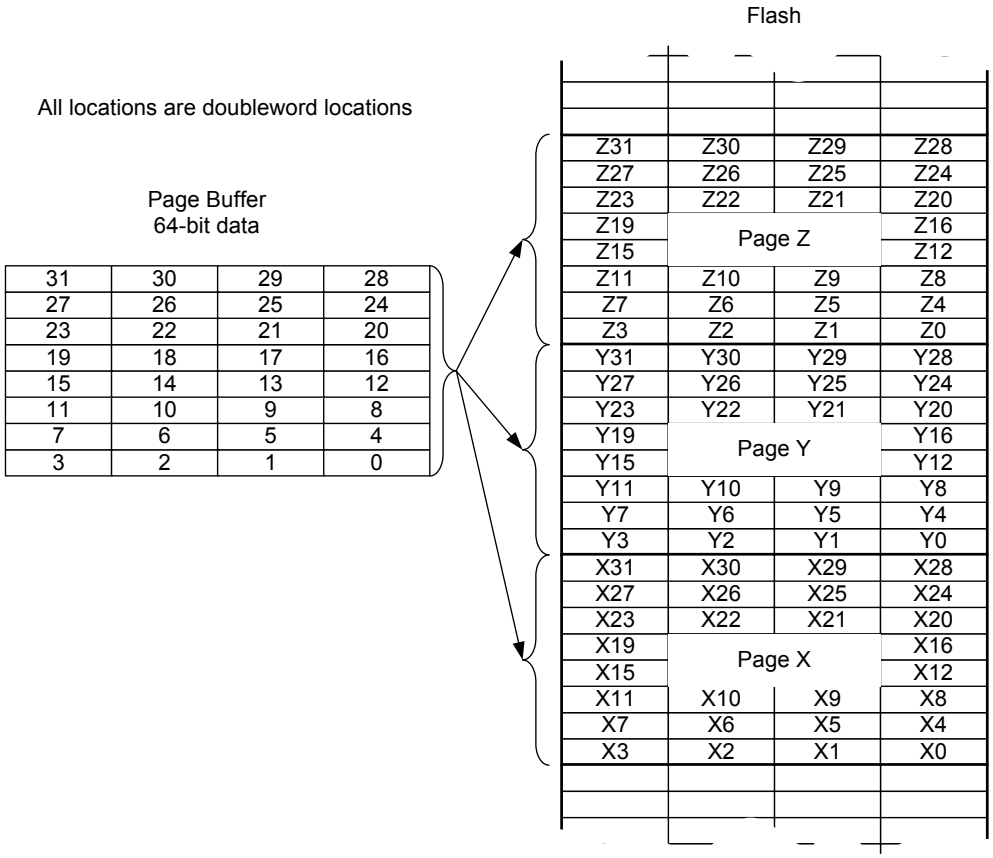
8.4.7 Page Buffer Operations

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 8.5.1 on page 43.

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 8-2. 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 8-2 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.

8.5 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 8.8.2 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) and Quick User Page Read (QPRUP) 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.

8.5.1 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 8.4.7 on page 40.
- 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.

8.5.2 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.

8.5.3 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 8.6. 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 8.6.

8.6 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 8-2. 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.

The BOOTPROT fuses protects the following address space for the Boot Loader:

Table 8-4. 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	1Kbyte
4	0-7	2Kbyte
3	0-15	4Kbyte
2	0-31	8Kbyte
1	0-63	16Kbyte
0	0-127	32Kbyte

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 fuse bits was attempted while the flash is locked by the security bit.

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 8.5.3.

8.7 Security Bit

The security bit allows the entire device 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.

8.8 User Interface

Table 8-5. 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-only	-(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.

8.8.1 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.

8.8.2 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 8-6. 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 8-6. 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 8-7. 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
RESERVED	16-31	

8.8.3 Flash Status Register

Name: FSR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8
23	22	21	20	19	18	17	16
LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	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.
- **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.

8.8.4 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 8-8. 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 8-9. 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

8.8.5 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.

8.8.6 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.

8.8.7 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.

8.9 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.

8.9.1 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]	Reserved	Reserved	Reserved	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 716.

BOOTPROT, EPFL, LOCK

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

Reserved

These fuses should never be programmed.

8.9.1.1 *Default Fuse Value*

The devices are shipped with the FGPFRLO register value: 0xFFF5FFFF:

- BODEN fuses set to 11.
- BODHYST fuse set to 1. The BOD hysteresis is enabled.
- BODLEVEL fuses set to 11111. BOOTPROT fuses set to 010. The bootloader protected size is 8KBytes.
- EPFL fuse set to 1. External privileged fetch is not locked.
- Reserved fuses set to 1.
- LOCK fuses set to 1111111111111111. No region locked.

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

8.9.2 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.

8.9.2.1 *Default user page first word value*

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

- WDTAUTO set to 1, WDT disabled.

8.10 Bootloader Configuration

The bootloader uses two words in the flash User page to store its configuration:

- Configuration word 1 at address 0x808000FC is read first at boot time to know if it should start the ISP process inconditionally and whether it should use the configuration word 2 where further configuration is stored.
- Configuration word 2 at address 0x808000F8 stores the I/O conditions that determine which of the ISP and the application to start at the end of the boot process. Please refer to the bootloader documentation for more information.
- The default value of the bootloader flash User page configuration word1 is 0xE11EFFD7.
- The default value of the bootloader flash User page configuration word 2 is 0x929E0D6B.

8.11 Serial Number

Each device has a unique 120 bits serial number readable from address 0x80800114 to 0x80800122.

8.12 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 8-10. Module Configuration

Feature	ATUC256D	ATUC128D	ATUC64D
Flash size	256Kbytes	128Kbytes	64Kbytes
Number of pages	512	512	256
Page size	512 bytes	256 bytes	256 bytes
FPR register value	0x00000409	0x00000307	0x00000305

Table 8-11. Module Clock Name

Module Name	Clock Name	Clock Name
FLASHCDW	CLK_FLASHCDW_HSB	CLK_FLASHCDW_PB

9. HSB Bus Matrix (HMATRIXB)

Rev: 1.3.0.3

9.1 Features

- **User Interface on peripheral bus**
- **Configurable number of masters (up to 16)**
- **Configurable number of slaves (up to 16)**
- **One decoder for each master**
- **Programmable arbitration for each slave**
  - **Round-Robin**
  - **Fixed priority**
- **Programmable default master for each slave**
  - **No default master**
  - **Last accessed default master**
  - **Fixed default master**
- **One cycle latency for the first access of a burst**
- **Zero cycle latency for default master**
- **One special function register for each slave (not dedicated)**

9.2 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.

9.3 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.

9.3.1 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.

9.4 Functional Description

9.4.1 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.

9.4.1.1 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.

9.4.1.2 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.

9.4.1.3 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).

9.4.2 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” .

9.4.2.1 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.

9.4.2.2 *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.

9.4.2.3 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).

9.4.3 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.

9.5 User Interface

Table 9-1. 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 9-1. 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 9-1. 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	–

9.5.1 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 9-2. 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.

9.5.2 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.

9.5.3 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.

9.5.4 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.

9.5.5 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.

9.6 Module Configuration

9.6.1 Bus Matrix Connections

The bus matrix has 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 9-3. High Speed Bus Masters

Master 0	CPU Data
Master 1	CPU Instruction
Master 2	CPU SAB
Master 3	PDCA
Master 4	USBC Built-in DMA

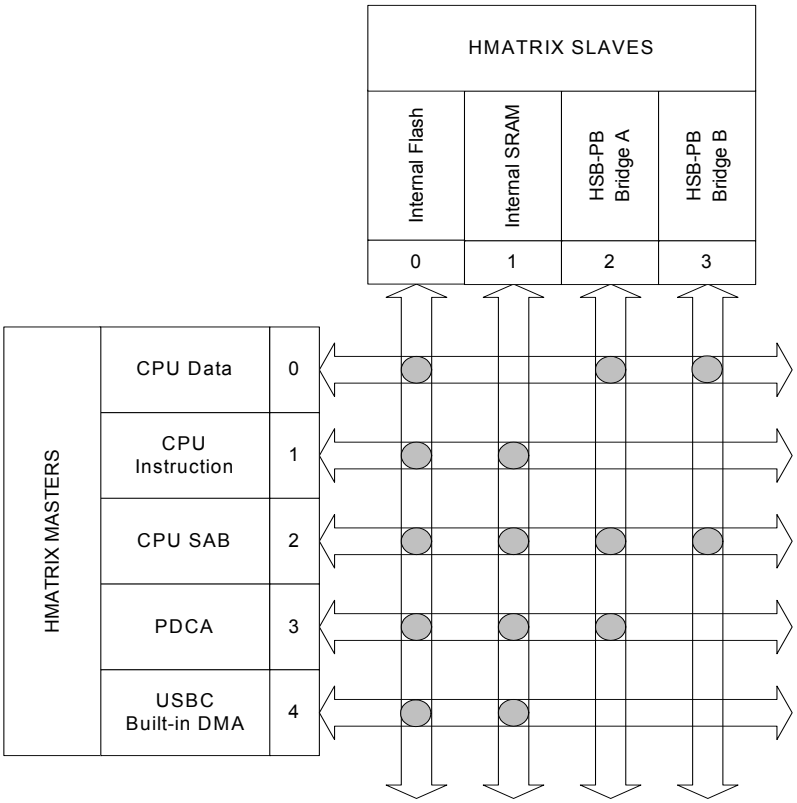
Each slave has its own arbiter, thus allowing a different arbitration per slave. The slave number in the table below can be used to index the HMATRIX control registers. For example, SCFG1 is associated with the Internal SRAM Slave Interface.

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

Table 9-4. High Speed Bus Slaves

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

Figure 9-1. HMatrix Master / Slave Connections



10. Peripheral DMA Controller (PDCA)

Rev: 1.2.3.1

10.1 Features

- **Multiple channels**
- **Generates transfers between memories and peripherals such as USART and SPI**
- **Two address pointers/counters per channel allowing double buffering**
- **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:

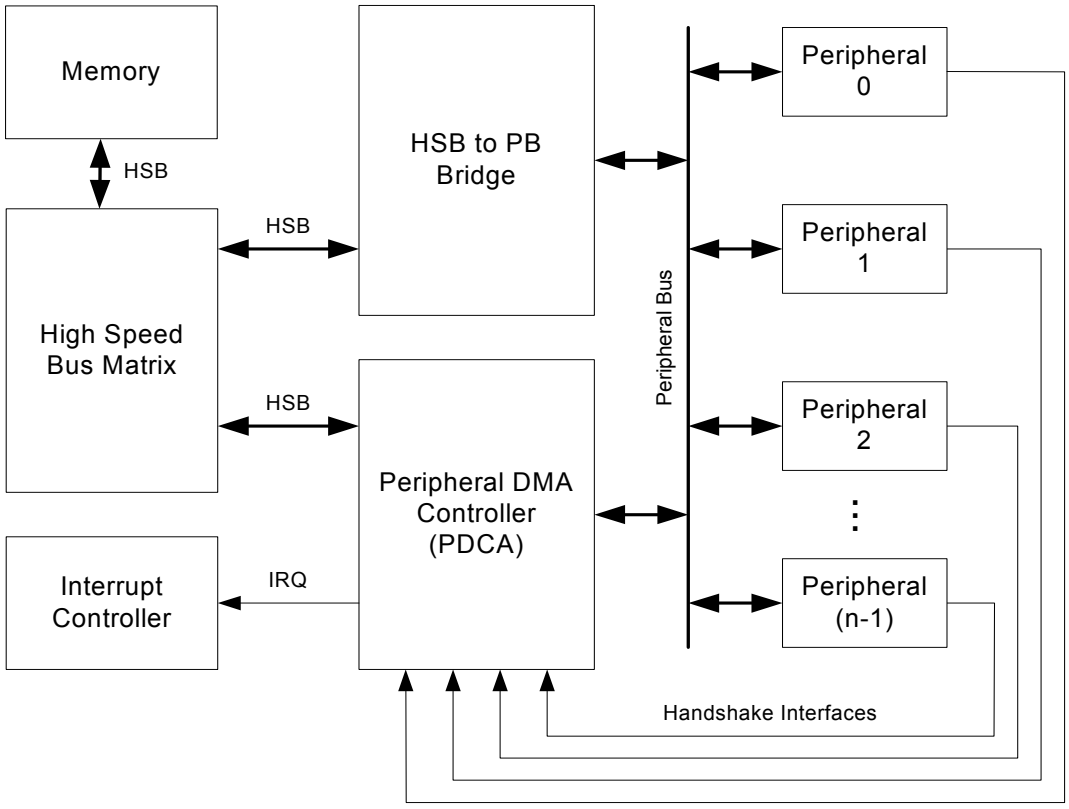
- A Peripheral Select Register
- A 32-bit memory pointer
- A 16-bit transfer counter
- A 32-bit memory pointer reload value
- A 16-bit transfer counter reload value

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 in 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.

10.5 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 written 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 PSR register. Writing the PID will both select the direction of the transfer (memory to peripheral or peripheral to memory), which handshake interface to use, and the address of the peripheral holding register. Refer to the Peripheral Identity (PID) table in the Module Configuration section for the peripheral PID values.

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 different 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 a transfer error occurs.

10.6 User Interface

10.6.1 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
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.

10.6.2 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.

10.6.3 Version Register Memory Map

Table 10-3. 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.

10.6.4 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.

10.6.5 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.

10.6.6 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.

10.6.7 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.

10.6.8 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.

10.6.9 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.

10.6.10 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	-	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.
- **SIZE: Size of Transfer**

Table 10-4. Size of Transfer

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

10.6.11 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.

10.6.12 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.

10.6.13 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.

10.6.14 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.

10.6.15 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.

10.6.16 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.

10.7 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 10-5. PDCA Configuration

Feature	PDCA
Number of channels	7

Table 10-6. Module Clock Name

Module name	PB Clock Name	HSB Clock Name
PDCA	CLK_PDCA_PB	CLK_PDCA_HSB

Table 10-7. Register Reset Values

Register	Reset Value
PSR CH n	n
VERSION	123

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 10-8. Peripheral Identity Values

PID	Direction	Peripheral Instance	Peripheral Register
0	RX	USART0	RHR
1	RX	USART1	RHR
2	RX	USART2	RHR
3	RX	SPI	RDR
4	RX	TWIM	RHR
5	RX	TWIS	RHR
6	RX	IISC	RHR
7	RX	IISC	RHR
8	RX	ADCIFD	LCV
9	RX	CAT	ACOUNT
10	RX	CAT	DMATSR
11	RX	AW	RHR
12	TX	USART0	THR
13	TX	USART1	THR
14	TX	USART2	THR
15	TX	SPI	TDR
16	TX	TWIM	THR

Table 10-8. Peripheral Identity Values

PID	Direction	Peripheral Instance	Peripheral Register
17	TX	TWIS	THR
18	TX	IISC	THR
19	TX	IISC	THR
20	TX	CAT	DMATSW
21	TX	AW	THR

11. Interrupt Controller (INTC)

Rev: 1.0.2.5

11.1 Features

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

11.2 Overview

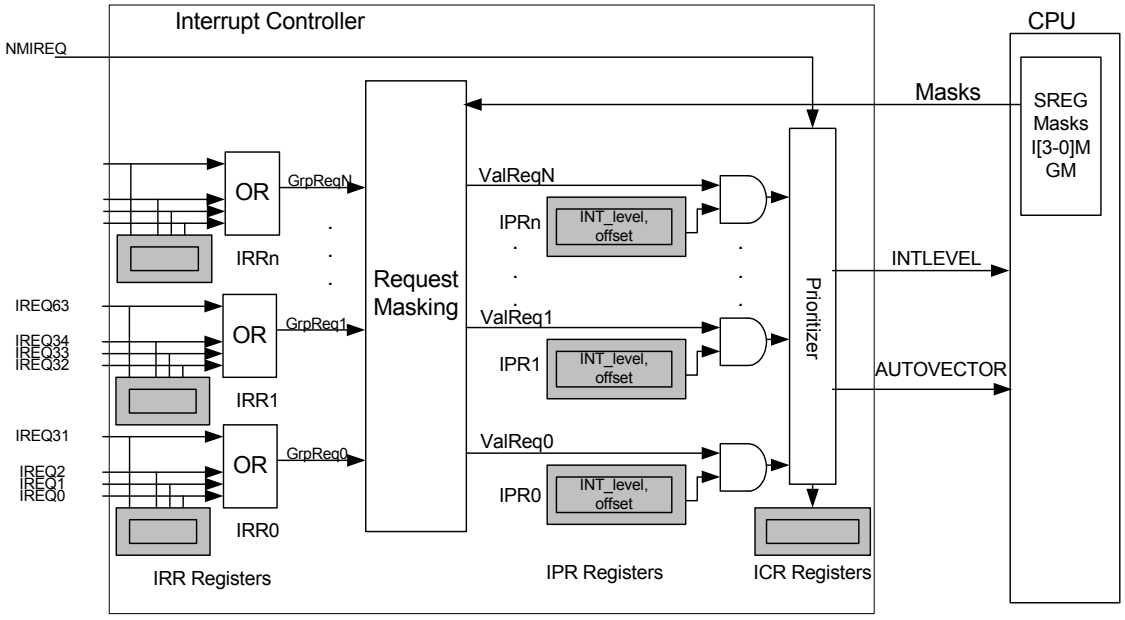
The INTC collects interrupt requests from the peripherals, prioritizes them, and delivers an interrupt request and an autovector to the CPU. The AVR32 architecture supports 4 priority levels for regular, maskable interrupts, and a Non-Maskable Interrupt (NMI).

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

11.3 Block Diagram

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

Figure 11-1. INTC Block Diagram



11.4 Product Dependencies

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

11.4.1 Power Management

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

11.4.2 Clocks

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

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

11.4.3 Debug Operation

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

11.5 Functional Description

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

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

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

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

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

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

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

11.5.1 Non-Maskable Interrupts

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

11.5.2 CPU Response

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

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

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

11.5.3 Clearing an Interrupt Request

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

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

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

11.6 User Interface

Table 11-1. INTC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Interrupt Priority Register 0	IPR0	Read/Write	0x00000000
0x004	Interrupt Priority Register 1	IPR1	Read/Write	0x00000000
...	...	...	...	...
0x0FC	Interrupt Priority Register 63	IPR63	Read/Write	0x00000000
0x100	Interrupt Request Register 0	IRR0	Read-only	N/A
0x104	Interrupt Request Register 1	IRR1	Read-only	N/A
...	...	...	...	...
0x1FC	Interrupt Request Register 63	IRR63	Read-only	N/A
0x200	Interrupt Cause Register 3	ICR3	Read-only	N/A
0x204	Interrupt Cause Register 2	ICR2	Read-only	N/A
0x208	Interrupt Cause Register 1	ICR1	Read-only	N/A
0x20C	Interrupt Cause Register 0	ICR0	Read-only	N/A

11.6.1 Interrupt Priority Registers

Name: IPR0...IPR63

Access Type: Read/Write

Offset: 0x000 - 0x0FC

Reset Value: 0x00000000

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

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

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

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

- **INTLEVEL: Interrupt Level**
  Indicates the EVBA-relative offset of the interrupt handler of the corresponding group:
  00: INT0: Lowest priority
  01: INT1
  10: INT2
  11: INT3: Highest priority
- **AUTOVECTOR: Autovector Address**
  Handler offset is used to give the address of the interrupt handler. The least significant bit should be written to zero to give halfword alignment.

11.6.2 Interrupt Request Registers

Name: IRR0...IRR63

Access Type: Read-only

Offset: 0x0FF - 0x1FC

Reset Value: N/A

31	30	29	28	27	26	25	24
IRR[32*x+31]	IRR[32*x+30]	IRR[32*x+29]	IRR[32*x+28]	IRR[32*x+27]	IRR[32*x+26]	IRR[32*x+25]	IRR[32*x+24]
23	22	21	20	19	18	17	16
IRR[32*x+23]	IRR[32*x+22]	IRR[32*x+21]	IRR[32*x+20]	IRR[32*x+19]	IRR[32*x+18]	IRR[32*x+17]	IRR[32*x+16]
15	14	13	12	11	10	9	8
IRR[32*x+15]	IRR[32*x+14]	IRR[32*x+13]	IRR[32*x+12]	IRR[32*x+11]	IRR[32*x+10]	IRR[32*x+9]	IRR[32*x+8]
7	6	5	4	3	2	1	0
IRR[32*x+7]	IRR[32*x+6]	IRR[32*x+5]	IRR[32*x+4]	IRR[32*x+3]	IRR[32*x+2]	IRR[32*x+1]	IRR[32*x+0]

- **IRR: Interrupt Request line**

  This bit is cleared when no interrupt request is pending on this input request line.

  This bit is set when an interrupt request is pending on this input request line.

  The are 64 IRRs, one for each group. Each IRR has 32 bits, one for each possible interrupt request, for a total of 2048 possible input lines. The IRRs are read by the software interrupt handler in order to determine which interrupt request is pending. The IRRs are sampled continuously, and are read-only.

11.6.3 Interrupt Cause Registers

Name: ICR0...ICR3

Access Type: Read-only

Offset: 0x200 - 0x20C

Reset Value: N/A

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

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

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

7	6	5	4	3	2	1	0
-	-	CAUSE					

- **CAUSE: Interrupt Group Causing Interrupt of Priority n**
  ICRn identifies the group with the highest priority that has a pending interrupt of level n. This value is only defined when at least one interrupt of level n is pending.

11.7 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 11-2. INTC Clock Name

Module Name	Clock Name
INTC	CLK_INTC

11.7.1 Interrupt Request Signal Map

The various modules may output Interrupt request signals. These signals are routed to the Interrupt Controller (INTC), described in a later chapter. The Interrupt Controller supports up to 64 groups of interrupt requests. Each group can have up to 32 interrupt request signals. All interrupt signals in the same group share the same autovector address and priority level. Refer to the documentation for the individual submodules for a description of the semantics of the different interrupt requests.

The interrupt request signals are connected to the INTC as follows.

Table 11-3. Interrupt Request Signal Map

Group	Line	Module	Signal
0	0	AVR32UC3 CPU	SYSREG COMPARE
1	0	External Interrupt Controller	EIC 1
	1	External Interrupt Controller	EIC 2
	2	External Interrupt Controller	EIC 3
	3	External Interrupt Controller	EIC 4
	4	External Interrupt Controller	EIC 5
	5	External Interrupt Controller	EIC 6
	6	External Interrupt Controller	EIC 7
	7	External Interrupt Controller	EIC 8
	8	Asynchronous Timer	AST ALARM
	9	Power Manager	PM
2	0	General Purpose Input/Output Controller	GPIO 0
	1	General Purpose Input/Output Controller	GPIO 1
	2	General Purpose Input/Output Controller	GPIO 2
	3	General Purpose Input/Output Controller	GPIO 3
	4	General Purpose Input/Output Controller	GPIO 4
	5	General Purpose Input/Output Controller	GPIO 5
	6	General Purpose Input/Output Controller	GPIO 6

Table 11-3. Interrupt Request Signal Map

3	0	Peripheral DMA Controller	PDCA 0
	1	Peripheral DMA Controller	PDCA 1
	2	Peripheral DMA Controller	PDCA 2
	3	Peripheral DMA Controller	PDCA 3
	4	Peripheral DMA Controller	PDCA 4
	5	Peripheral DMA Controller	PDCA 5
	6	Peripheral DMA Controller	PDCA 6
4	0	Flash Controller	FLASHCDW
5	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART0
6	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART1
7	0	Universal Synchronous/Asynchronous Receiver/Transmitter	USART2
8	0	Frequency Meter	FREQM
9	0	Serial Peripheral Interface	SPI
10	0	Two-wire Master Interface	TWIM
11	0	Two-wire Slave Interface	TWIS
12	0	Basic Pulse Width Modulation Controller	PWMA
13	0	Inter-IC Sound (I2S) Controller	IISC
14	0	Timer/Counter	TC0
	1	Timer/Counter	TC1
	2	Timer/Counter	TC2
15	0	ADC controller interface	ADCIFD SEQ
	1	ADC controller interface	ADCIFD TIMING
	2	ADC controller interface	ADCIFD WINDOW
17	0	USB 2.0 Interface	USBC
18	0	AVR32UC3 CPU	OCD DCEMU_DIRTY
	1	AVR32UC3 CPU	OCD DCCPU_READ
19	0	System Control Interface	SCIF
20	0	Asynchronous Timer	AST CLKREADY
	1	Asynchronous Timer	AST OVF
	2	Asynchronous Timer	AST PER
	3	Asynchronous Timer	AST READY
21	0	Capacitive Touch Module	CAT
22	0	aWire	AW

12. Power Manager (PM)

Rev: 4.1.2.4

12.1 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**

12.2 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).

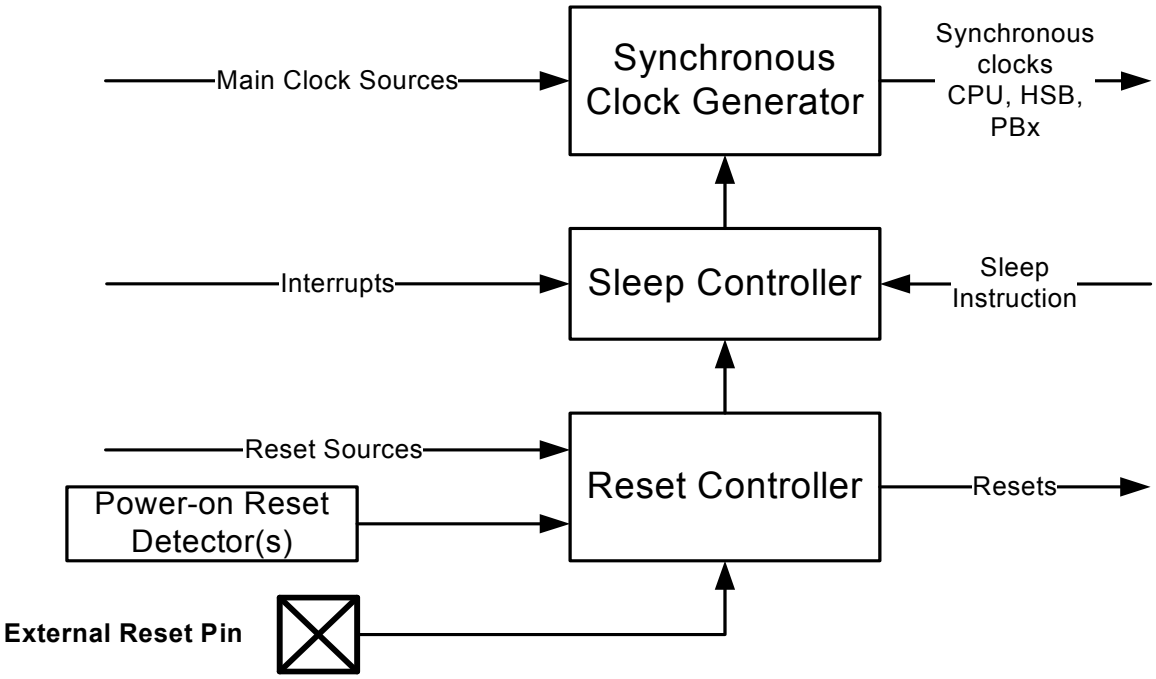
The PM contains advanced power-saving features, allowing the user to optimize the power consumption for an application. The synchronous clocks are divided into a number of clock domains, one for the CPU and HSB, and one for each PBx. The clocks can run at different speeds, allowing the user to save power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

Each module has a separate clock, enabling the user to save power by switching off clocks to inactive modules. Clocks and oscillators can be automatically switched off during idle periods by the CPU sleep instruction. The system will return to normal operation when interrupts occur.

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.

12.3 Block Diagram

Figure 12-1. PM Block Diagram



12.4 I/O Lines Description

Table 12-1. I/O Lines Description

Name	Description	Type	Active Level
RESET_N	Reset	Input	Low

12.5 Product Dependencies

12.5.1 Interrupt

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

12.5.2 Clock Implementation

In UC3D, the HSB shares source clock with the CPU. Write attempts to the HSB Clock Select register (HSBSEL) will be ignored, and it will always read the same as the CPU Clock Select register (CPUSEL).

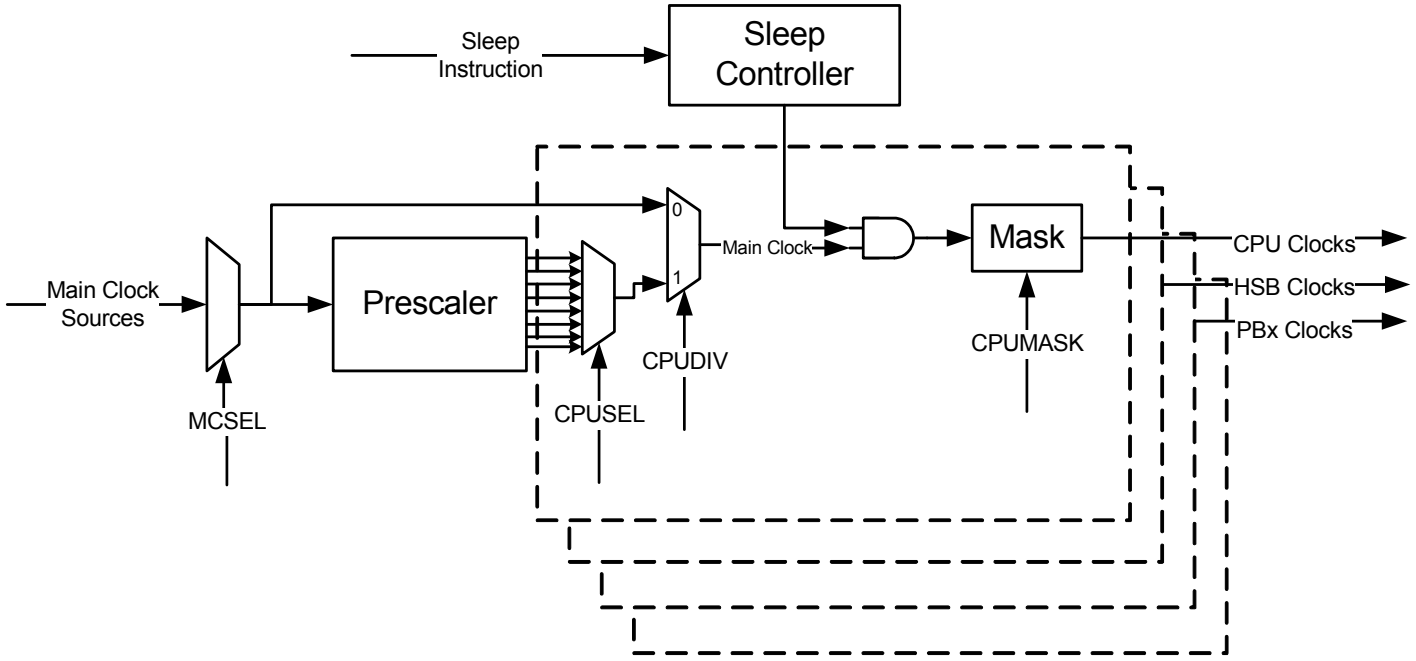
The PM bus interface clock (CLK_PM) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. If disabled it can only be re-enabled by a reset.

12.6 Functional Description

12.6.1 Synchronous Clocks

The System RC Oscillator (RCSYS) and a selection of other clock sources can provide the source for the main clock, which is the origin for the synchronous CPU/HSB and PBx module clocks. For details about the other main clock sources, please refer to the Main Clock Control (MCCTRL) register description. The synchronous clocks can run of the main clock and all the 8-bit prescaler settings as long as $f_{CPU} \geq f_{PBx}$. The synchronous clock source can be changed on-the fly, according to variations in application load. The clock domains can be shut down in sleep mode, as described in Section 12.6.3. The module clocks in every synchronous clock domain can be individually masked to minimize power consumption in inactive modules.

Figure 12-2. Synchronous Clock Generation



12.6.1.1 *Selecting the main clock source*

The common main clock can be connected to RCSYS or a selection of other clock sources. For details about the other main clock sources, please refer to the MCCTRL register description. By default, the main clock will be connected to RCSYS. The user can connect the main clock to another source by writing to the Main Clock Select (MCCTRL.MCSEL) field. The user must first assure that the source is enabled and ready in order to avoid a deadlock. Care should also be taken so that the new synchronous clock frequencies do not exceed the maximum frequency for each clock domain.

12.6.1.2 *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 a one to the CPU Division bit in the CPU Clock Select register (CPUSEL.CPUDIV), and a value to the CPU Clock Select field (CPUSEL.CPUSEL), resulting in a CPU clock frequency:

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

Similarly, the PBx clocks can be divided by writing their respective Clock Select (PBxSEL) registers to get the divided PBx frequency:

$f_{PBx} = f_{main} / 2^{(PBSEL+1)}$

The PBx clock frequency can not exceed the CPU clock frequency. The user must select a PBxSEL.PBSEL value greater than or equal to the CPUSEL.CPUSEL value, so that $f_{CPU} \geq f_{PBx}$. If the user selects division factors that will result in $f_{CPU} < f_{PBx}$, the Power Manager will automatically change the PBxSEL.PBSEL/PBDIV values to ensure correct operation ($f_{CPU} \geq f_{PBx}$).

The HSB clock will always be forced to the same division as the CPU clock.

To ensure correct operation, the frequencies must never exceed the specified maximum frequency for each clock domain.

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

12.6.1.3 Clock Ready flag

There is a slight delay from CPUSEL and PBxSEL being written to the new clock setting taking effect. During this interval, the Clock Ready bit in the Status Register (SR.CKRDY) will read as zero. When the clock settings change is completed, the bit will read as one. The Clock Select registers (CPUSEL, PBxSEL) must not be written to while SR.CKRDY is zero, or the system may become unstable or hang.

The Clock Ready bit in the Interrupt Status Register (ISR.CKRDY) is set on a SR.CKRDY zero-to-one transition. If the Clock Ready bit in the Interrupt Mask Register (IMR.CKRDY) is set, an interrupt request is generated. IMR.CKRDY is set by writing a one to the corresponding bit in the Interrupt Enable Register (IER.CKRDY).

12.6.2 Peripheral Clock Masking

By default, the clocks 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 a zero to the corresponding bit in the corresponding Clock Mask (CPUMASK/HSBMASK/PBxMASK) register. When a module is not clocked, it will cease operation, and its registers cannot be read nor written. The module can be re-enabled later by writing a one to the corresponding mask bit. A module may be connected to several clock domains, in which case it will have several mask bits. The Maskable Module Clocks table in the Clock Mask register description contains a list of implemented maskable clocks.

12.6.2.1 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, which contains the mask registers, or the corresponding PBx bridge, will make it impossible to write to the mask registers again. In this case, they can only be re-enabled by a system reset.

12.6.3 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 it and other (optional) clock domains off to save power. This is done by the sleep instruction, which takes the sleep mode index number from Table 12-2 on page 109 as argument.

12.6.3.1 *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 in 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.

12.6.3.2 *Supported sleep modes*

The following sleep modes are supported. These are detailed in Table 12-2 on page 109.

- Idle: The CPU is stopped, the rest of the device is operational.
- Frozen: The CPU and HSB modules are stopped, peripherals are operational.
- Standby: All synchronous clocks are stopped, and the clock sources are running, allowing for a quick wake-up to normal mode.
- Stop: As Standby, but oscillators, and other clock sources are also stopped. 32KHz Oscillator OSC32K(2), RCSYS, AST, and WDT will remain operational.
- DeepStop: All synchronous clocks and clock sources are stopped. Bandgap voltage reference and BOD are turned off. OSC32K(2) and RCSYS remain operational.
- Static: All clock sources, including RCSYS are stopped. Bandgap voltage reference and BOD are turned off. OSC32K(2) remains operational. (2)

Table 12-2. Sleep Modes

Index(1)	Sleep Mode	CPU	HSB	PBx, GCLK	Clock Sources(3), SYSTIMER(4)	OSC32K(2)	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

Notes:
1. The sleep mode index is used as argument for the sleep instruction.
2. OSC32K will only remain operational if pre-enabled.
3. Clock sources other than those specifically listed in the table.
4. SYSTIMER is the clock for the CPU COUNT and COMPARE registers.

The internal voltage regulator is also adjusted according to the sleep mode in order to reduce its power consumption.

12.6.3.3 *Waking from sleep modes*

There are two types of wake-up sources from sleep mode, synchronous and asynchronous. Synchronous wake-up sources are all non-masked interrupts. Asynchronous wake-up sources are AST, WDT, external interrupts from EIC, external reset, and all asynchronous wake-ups enabled in the Asynchronous Wake Up Enable (AWEN) register. The valid wake-up sources for each sleep mode are detailed in Table 12-3 on page 110.

Table 12-3. Wake-up Sources

Index[1]	Sleep Mode	Wake-up Sources
0	**Idle**	Synchronous, Asynchronous
1	**Frozen**	Synchronous[2], Asynchronous
2	**Standby**	Asynchronous
3	**Stop**	Asynchronous
4	**DeepStop**	Asynchronous
5	**Static**	Asynchronous[3]

Notes:
1. The sleep mode index is used as argument for the sleep instruction.
2. Only PB modules operational, as HSB module clocks are stopped.
3. WDT only available if clocked from pre-enabled OSC32K.

12.6.3.4 *SleepWalking*

In all sleep modes where the PBx clocks are stopped, the device can partially wake up if a PBx module asynchronously discovers that it needs its clock. Only the requested clocks and clock sources needed will be started, all other clocks will remain 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. Generic clocks can also 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 device is SleepWalking any asynchronous wake-up can wake the device up 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, it will wake the whole device up.

12.6.3.5 *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 will prevent erratic behavior caused by entering or exiting sleep modes. Please refer to the relevant module documentation for recommended actions.

Communication between the synchronous clock domains is disturbed when entering and exiting sleep modes. Bus transactions over clock domains affected by the sleep mode are therefore not recommended. The system may hang if the bus clocks are stopped during 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. In order to let potential PBx write operations complete, the user should let the CPU perform a PBx register read operation before issuing the sleep instruction. This will stall the CPU until pending PBx operations have completed.

12.6.4 Divided PB Clocks

The clock generator in the Power Manager provides divided PBx clocks for use by peripherals that require a prescaled PBx clock. This is described in the documentation for the relevant modules. The divided clocks are directly maskable, and are stopped in sleep modes where the PBx clocks are stopped.

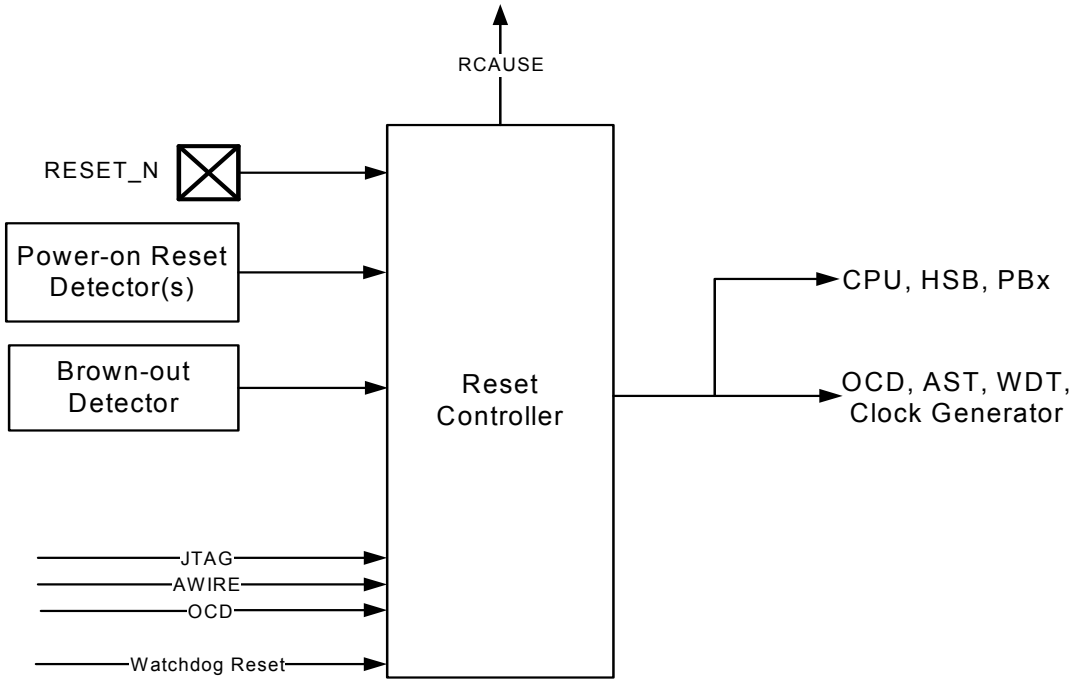
12.6.5 Reset Controller

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

The device contains a Power-on Reset (POR) 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 pulling the RESET_N pin low. This pin has an internal pull-up, and does not need to be driven externally during normal operation. Table 12-4 on page 112 lists these and other reset sources supported by the Reset Controller.

Figure 12-3. 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 12-4. Reset Description

Reset Source	Description
Power-on Reset	Supply voltage below the Power-on Reset detector threshold voltage V_{POT}
External Reset	RESET_N pin asserted
Brown-out Reset	VDDCORE supply voltage below the Brown-out detector threshold voltage
Watchdog Timer	See Watchdog Timer documentation
OCD	See On-Chip Debug documentation

Depending on the reset source, when a reset occurs, some parts of the device are not always reset. Only the Power-on Reset (POR) will force a whole device reset. Refer to the table in the Module Configuration section at the end of this chapter for further details. The latest reset cause can be read in the RCAUSE register, and can be read during the applications boot sequence in order to determine proper action.

12.6.5.1 Power-on Reset Detector

The Power-on Reset 1.8V (POR18) detector monitors the VDDCORE supply pin and generates a Power-on Reset (POR) when the device is powered on. The POR is active until the VDDCORE voltage is above the power-on threshold level (V_{POT}). The POR will be re-generated if the voltage drops below the power-on threshold level. See Electrical Characteristics for parametric details.

The Power-on Reset 3.3V (POR33) detector monitors the internal regulator supply pin and generates a Power-on Reset (POR) when the device is powered on. The POR is active until the internal regulator supply voltage is above the regulator power-on threshold level (V_{POT}). The POR will be re-generated if the voltage drops below the regulator power-on threshold level. See Electrical Characteristics for parametric details.

12.6.5.2 External Reset

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

12.6.6 Clock Failure Detector

This mechanism automatically switches the main clock source to the safe RCSYS clock when the main clock source fails. This may happen when an external crystal is selected as a source for the main clock and the crystal is not mounted on the board. The main clock is compared with RCSYS, and if no rising edge of the main clock is detected during one RCSYS period, the clock is considered to have failed.

The detector is enabled by writing a one to the Clock Failure Detection Enable bit in the Clock Failure Detector Control Register (CFDCTRL.CFDEN). As soon as the detector is enabled, the clock failure detector will monitor the divided main clock. Note that the detector does not monitor the main clock if 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 Clock Failure Detected (CFD)

interrupt is generated if enabled. The MCCTRL register is also changed by hardware to indicate that the main clock comes from RCSYS.

12.6.7 Interrupts

The PM has a number of interrupt sources:

- AE - Access Error,
  - A lock protected register is written to without first being unlocked.
- CKRDY - Clock Ready:
  - New Clock Select settings in the CPUSEL/PBxSEL registers have taken effect. (A zero-to-one transition on SR.CKRDY is detected).
- CFD - Clock Failure Detected:
  - The system detects that the main clock is not running.

The Interrupt Status Register contains one bit for each interrupt source. A bit in this register is set on a zero-to-one transition of the corresponding bit in the Status Register (SR), and cleared by writing a one to the corresponding bit in the Interrupt Clear Register (ICR). 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 Power Manager 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 Power Manager will remain active until all the bits in ISR are cleared.

12.7 User Interface

Table 12-5. 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
0x020	CPU Mask	CPUMASK	Read/Write	0x00000003
0x024	HSB Mask	HSBMASK	Read/Write	0x0000001F
0x028	PBA Mask	PBAMASK	Read/Write	0x003FFFFF
0x02C	PBB Mask	PBBMASK	Read/Write	0x00000007
0x040	PBA Divided Mask	PBADIVMASK	Read/Write	0x0000007F
0x054	Clock Failure Detector Control	CFDCTRL	Read/Write	0x00000000
0x058	Unlock Register	UNLOCK	Write-only	0x00000000
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
0x160	Peripheral Power Control Register	PPCR	Read/Write	0x00000018
0x180	Reset Cause Register	RCAUSE	Read-only	-(1)
0x184	Wake Cause Register	WCAUSE	Read-only	-(2)
0x188	Asyncronous Wake Enable	AWEN	Read/Write	0x00000000
0x3F8	Configuration Register	CONFIG	Read-only	0x00000003
0x3FC	Version Register	VERSION	Read-only	0x00000412

Notes: 1. Latest Reset Source.
2. Latest Wake Source.

12.7.1 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 12-6. Main clocks in .

MCSEL[2:0]	Main clock source
0	System RC oscillator (RCSYS)
1	Oscillator0 (OSC0)
2	PLL0
3	PLL1
4	RC120[1]
others	reserved

Note: 1. If the 120MHz 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.

12.7.2 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.

12.7.3 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

12.7.4 PBx Clock Select

Name: PBxSEL

Access Type: Read/Write

Offset: 0x00C-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
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.

12.7.5 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 12-7.

Table 12-7. Maskable Module Clocks in UC3D.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK
0	-	FLASHCDW	PDCA	USBC
1	OCD[1]	PBA bridge	INTC	HMATRIX
2	-	PBB bridge	PM	FLASHCDW
3	-	USBC	AST	-
4	-	PDCA	WDT	-
5	-	-	EIC	-
6	-	-	GPIO	-
7	-	-	USART0	-
8	-	-	USART1	-
9	-	-	USART2	-
10	-	-	SPI	-
11	-	-	TWIM	-

Table 12-7. Maskable Module Clocks in UC3D.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK
12	-	-	TWIS	-
13	-	-	PWMA	-
14	-	-	IISC	-
15	-	-	TC	-
16	SYSTIMER (COMPARE/COUNT REGISTERS CLK)	-	ADCIFD	-
17	-	-	SCIF	-
18	-	-	FREQM	-
19	-	-	CAT	-
20	-	-	GLOC	-
21	-	-	AW	-
22	-	-	-	-
23	-	-	-	-
24	-	-	-	-
25	-	-	-	-
31:25	-	-	-	-

Note: 1. This bit must be one if the user wishes to debug the device with a JTAG debugger.

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.

12.7.6 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.

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.

12.7.7 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
-	-	-	-	-	-	OCPEN	CFDEN

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

12.7.8 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.

12.7.9 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x0C0

Reset Value: -

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

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

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

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

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

12.7.10 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x0C4

Reset Value: -

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

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

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

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

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

12.7.11 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x0C8

Reset Value: 0x00000000

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

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

12.7.12 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x0CC

Reset Value: 0x00000000

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

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

12.7.13 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x0D0

Reset Value: -

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

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

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

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

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

12.7.14 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.
- **OCP: Over Clock**
  - 0: Main clock is running at a legal frequency.
  - 1: Illegal frequency detected on main clock. Main clock is now running on RC osc.
- **CFD: Clock Failure Detected**
  - 0: Main clock is running correctly.
  - 1: Failure on main clock detected. Main clock is now running on RC osc.

12.7.15 Peripheral Power Control Register

Name: PPCR

Access Type: Read/Write

Offset: 0x160

Reset Value: 0x00000018

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

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

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

7	6	5	4	3	2	1	0
-	-	-	ASTRCMASK	TWISRCMASK	RSTTM	-	RSTPUN

- **ASTRCMASK : AST Request Clock Mask**
  0: AST Request Clock is disabled
  1: AST Request Clock is enabled
- **TWISRCMASK : TWIS0 Request Clock Mask**
  0: TWIS Request Clock is disabled
  1: TWIS Request Clock is enabled
- **RSTPUN: Reset Pullup, active low**
  0: Pull-up for external reset on
  1: Pull-up for external reset off
- **RSTTM : Reset test mode**
  0: External reset not in test mode
  1: External reset in test mode

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.

12.7.16 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		-	OCDRST

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

- **AWIRE: AWIRE reset**
  The CPU was reset by tthe AWIRE
- **OCDRST: OCD Reset**
  The CPU was reset because the RES strobe in the OCD Development Control register has been written to one.
- **JTAG: JTAG reset**
  The CPU was reset by the JTAG system reset.
- **SLEEP:**
  The CPU was reset because it went to SHUTDOWN or STATIC sleep mode.
- **CPUERR: CPU Error**
  The CPU was reset because it had detected an illegal access.
- **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.

12.7.17 Wake Cause Register

Name: WCAUSE

Access Type: Read-only

Offset: 0x184

Reset Value: Latest Wake Source

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

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

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

7	6	5	4	3	2	1	0
-	-	-	-	-	-	TWIS	USBC

A bit in this register is set on wake up caused by the corresponding peripheral.

12.7.18 Asynchronous Wake Up Enable Register

Name: AWEN

Access Type: Read/Write

Offset: 0x188

Reset Value: 0x00000000

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

Each bit in this register corresponds to an asynchronous wake up.
0: The correcponding wake up is disabled.
1: The corresponding wake up is enabled.

12.7.19 Configuration Register

Name: CONFIG

Access Type: Read-Only

Offset: 0x3F8

Reset Value: 0x000000003

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
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	OCP	-	-	PBD	PBC	PBB	PBA

This register shows the configuration of the PM.

- **HSBPEVC:HSB PEVC Clock Implemented**
  0: HSBPEVC not implemented.
  1: HSBPEVC implemented.
- **OCP: Over Clock Protection Implemented**
  0: OCP not implemented.
  1: OCP 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.

12.7.20 Version Register

Name: VERSION

Access Type: Read-Only

Offset: 0x3FC

Reset Value: 0x00000412

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.

12.8 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 12-8. Power Manager Clock Name

Module Name	Clock Name
PM	CLK_PM

Table 12-9. Register Reset Values

Register	Reset Value
VERSION	0x00000412

Table 12-10. Effect of the Different Reset Events

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

13. System Control Interface (SCIF)

Rev: 1.0.2.0

13.1 Features

- **Controls integrated oscillators and Phase Locked Loops (PLLs)**
- **Supports crystal oscillator 0.4-20 MHz (OSC0)**
- **Supports two Phase Locked Loop 80-240 MHz (PLL)**
- **Supports 32 KHz oscillator (OSC32K)**
- **Integrated 115 KHz RC oscillator (RCSYS)**
- **Generic clocks (GCLK) with wide frequency range provided**
- **Controls bandgap voltage reference through control and calibration registers**
- **Controls Brown-out detector (BOD)**
- **Controls Voltage Regulator (VREG) behavior and calibration**
- **Controls 120 MHz integrated RC Oscillator (RC120M)**

13.2 Overview

The System Control Interface (SCIF) controls the Oscillators, PLLs, Generic Clocks, BODs, and Voltage Regulator.

13.3 I/O Lines Description

Table 13-1. I/O Lines Description

Pin Name	Pin Description	Type
XIN0	Crystal 0 Input	Analog/Digital
XIN32	Crystal 32 Input (primary location)	Analog/Digital
XOUT0	Crystal 0 Output	Analog
XOUT32	Crystal 32 Output (primary location)	Analog
GCLK2-GCLK0	Generic Clock Output	Output

13.4 Product Dependencies

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

13.4.1 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.

13.4.2 Interrupt

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

13.4.3 Debug Operation

The SCIF module does not interact with debug operations.

13.4.4 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.)

13.5 Functional Description

13.5.1 Oscillator (OSC0) Operation

The main oscillator (OSC0) is designed to be used with an external 4 to 20 MHz crystal and two biasing capacitors, as shown in Figure 13-1. The oscillator can be used for the main clock in the device, as described in the Power Manager chapter. The oscillator can be used as source for the generic clocks, as described in ”Generic clocks” on page 140.

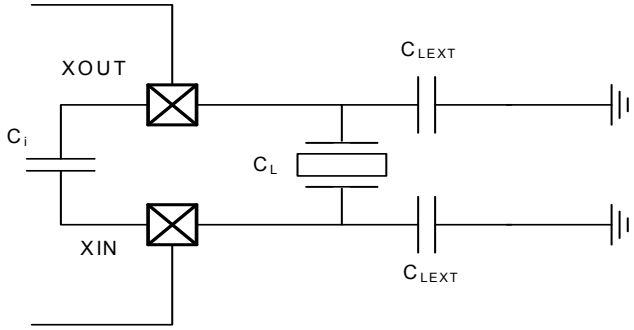
The oscillator is disabled by default after reset. When the oscillator is disabled, the XIN and XOUT pins can be used as general purpose I/Os. When the oscillator is configured to use an external clock, the clock must be applied to the XIN pin while the XOUT pin can be used as a general purpose I/O.

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 bit in PCLKSR is automatically set and cleared according to the status of the oscillator. A zero to one transition on this bit can also be configured to generate an interrupt, as described in Section 13.6.1.

Figure 13-1. Oscillator connections.



13.5.2 32KHz Oscillator Operation

The 32KHz oscillator operates as described for the Oscillator above. The 32KHz oscillator is used as source clock for the Asynchronous Timer and the Watchdog Timer. The 32KHz oscillator can be used as source for the generic clocks.

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.

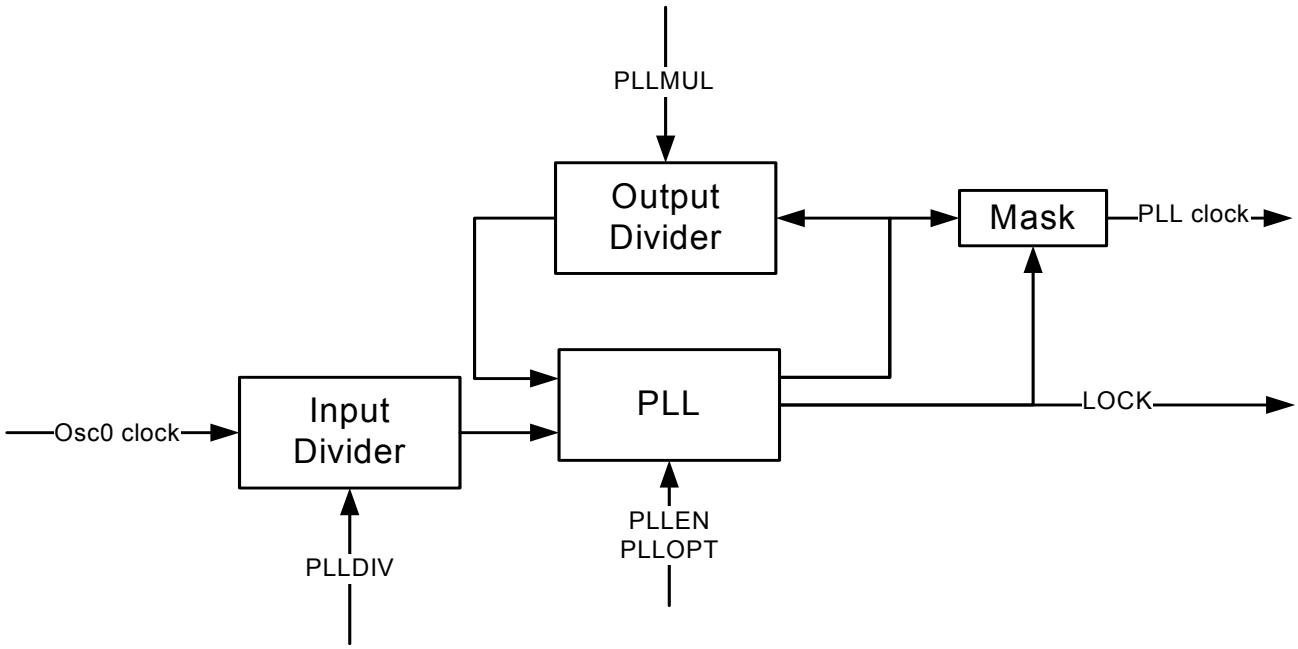
The 32KHz oscillator is not controlled by the sleep controller, and will run in all sleep modes if enabled.

13.5.3 PLL Operation

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

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

Figure 13-2. PLL with control logic and filters



13.5.3.1 Enabling the PLL

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

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

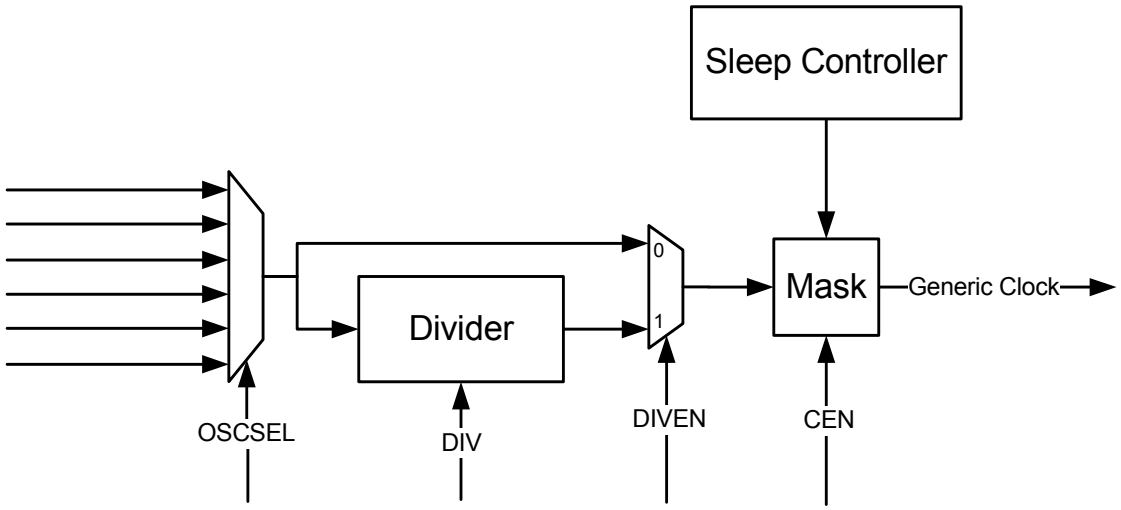
The lock signal for each PLL is available as a LOCKn flag in POSCSR. An interrupt can be generated on a 0 to 1 transition of these bits.

13.5.4 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 13-9 on page 166. 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 13-3. Generic clock generation



13.5.4.1 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:

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

13.5.4.2 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.

13.5.4.3 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.

13.5.4.4 Generic clock implementation

In UC3D, there are nine generic clocks. These are allocated to different functions as shown in Table 13-2. Note that only GCLK2-0 are routed out.

Table 13-2. Generic clock allocation

Clock number	Function
0	GCLK0, GLOC
1	GCLK1
2	GCLK2

Table 13-2. Generic clock allocation

Clock number	Function
3	USB clock (48 MHz)
4	PWMA
5	IISC
6	AST
7	-
8	ADCIFD

13.5.5 Brown Out Detection (BOD)

The Brown-Out Detector (BOD) monitors the internal voltage regulator output and compares the 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.BODET 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 RCSYS clock cycle and two main clocks cycles to avoid false results

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.

13.5.6 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

13.5.7 Voltage Regulator (VREG)

The embedded voltage regulator can be used to provide the internal logic voltage from the VDDIN. It is controlled by the VREGCR register. The voltage regulator is turned off by default at startup and automatically turned on if an external 3.3V power is provided on the VDDIN.

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.

13.5.8 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.

13.5.9 120 MHz RC Oscillator (RC120M)

The 120 MHz 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 140. 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 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.

13.5.10 General Purpose Low Power Registers

The GPLP registers are 32-bit registers that are reset only by power-on-reset. User software can

use these registers to save context variables in a very low power mode.

13.5.11 Interrupts

The SCIF has 8 separate interrupt sources. Refer to the PCLKSR register description.

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

13.6 User Interface

Table 13-3. SCIF Register Memory Map

Offset	Register	Register Name	Access	Reset
0x0000	Interrupt Enable Register	IER	Write-only	0x00000000
0x0004	Interrupt Disable Register	IDR	Write-only	0x00000000
0x0008	Interrupt Mask Register	IMR	Read-only	0x00000000
0x000C	Interrupt Status Register	ISR	Read-only	0x00000000
0x0010	Interrupt Clear Register	ICR	Write-only	0x00000000
0x0014	Power and Clocks Status Register	PCLKSR	Read-only	0x00000000
0x0018	Unlock Register	UNLOCK	Write-only	0x00000000
0x001C	PLL0 Control Register	PLL0CR	Read/Write	0x00000000
0x0020	PLL1 Control Register	PLL1CR	Read/Write	0x00000000
0x0024	Oscillator Control Register	OSCCTRL0	Read/Write	0x00000000
0x0028	BOD Level Register	BOD	Read/Write	0x00000000
0x002C	Bandgap Calibration Register	BGCR	Read/Write	0x00000000
0x0030	Voltage Regulator Calibration Register	VREGCR	Read/Write	0x00000000
0x0034	Voltage Regulator Control Register	VREGCTRL	Read/Write	0x00000000
0x0038	System RC Oscillator Calibration Register	RCCR	Read/Write	0x00000000
0x003C	32K Oscillator Control Register	OSCCTRL32	Read/Write	0x00000004
0x0044	120MHz RC Oscillator Control Register	RC120MCR	Read/Write	0x00000000
0x0048	General Purpose Low Power Register 0	GPLP0	Read/Write	0x00000000
0x004C	General Purpose Low Power Register 1	GPLP1	Read/Write	0x00000000
0x0060 - 0x080	Generic Clock Control	GCCTRL	Read/Write	0x00000000
0x03D4	PLL Interface Version Register	PLLVERSION	Read-only	(1)
0x03D8	Oscillator Interface Version Register	OSCVERSION	Read-only	(1)
0x03DC	BOD Interface Version Register	BODVERSION	Read-only	(1)
0x03E0	Voltage Regulator Interface Version Register	VREGVERSION	Read-only	(1)
0x03E4	System RC Oscillator Interface Version Register	RCCRVERSION	Read-only	(1)
0x03E8	32K Oscillator Interface Version Register	OSC32VERSION	Read-only	(1)
0x03F0	120MHz RC Oscillator Interface Version Register	RC120MVERSION	Read-only	(1)
0x03F4	GPLP Version Register	GPLPVERSION	Read-only	(1)
0x03F8	Generic Clock Interface 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.

13.6.1 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x0000

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	PLL1_LOCK LOST	PLL0_LOCK LOST	BODDET	PLL1_LOCK	PLL0_LOCK	OSC32RDY	OSC0RDY

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

13.6.2 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x0004

Reset Value: 0x00000000

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

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

13.6.3 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x0008

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	PLL1_LOCK LOST	PLL0_LOCK LOST	BODDET	PLL1_LOCK	PLL0_LOCK	OSC32RDY	OSC0RDY

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

13.6.4 Interrupt Status Register

Name: ISR

Access Type: Read-only

Offset: 0x000C

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	PLL1_LOCK LOST	PLL0_LOCK LOST	BODDET	PLL1_LOCK	PLL0_LOCK	OSC32RDY	OSC0RDY

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

13.6.5 Interrupt Clear Register

Name: ICR

Access Type: Write-only

Offset: 0x0010

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	PLL1_LOCK LOST	PLL0_LOCK LOST	BODDET	PLL1_LOCK	PLL0_LOCK	OSC32RDY	OSC0RDY

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

13.6.6 Power and Clocks Status Register

Name: PCLKSR

Access Type: Read-only

Offset: 0x0014

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	PLL1_LOCK LOST	PLL0_LOCK LOST	BODDET	PLL1_LOCK	PLL0_LOCK	OSC32RDY	OSC0RDY

- **AE: SCIF Access Error value**
  0: No access error has occurred on the SCIF.
  1: An access error has occurred on the SCIF.
- **PLLL1_LOCKLOST: PLL1 lock lost value**
  0: PLL1 has not lost its lock or has never been enabled.
  1: PLL1 has lost its lock, either by disabling the PLL1 or due to faulty operation.
- **PLLL0_LOCKLOST: PLL0 lock lost value**
  0: PLL1 has not lost its lock or has never been enabled.
  1: PLL1 has lost its lock, either by disabling the PLL1 or due to faulty operation.
- **BODDET: 1.8V Brown out detection**
  0: 1.8V BOD not enabled or the 1.8V power supply is above the 1.8V BOD threshold.
  1: 1.8V BOD enabled and the 1.8V power supply is going below 1.8V BOD threshold.
- **PLL1_LOCK: PLL1 Locked on Accurate value**
  0: PLL1 is unlocked on Accurate value.
  1: PLL1 is locked on Accurate value, and is ready to be selected as clock source with an accurate output clock.
- **PLL0_LOCK: PLL0 Locked on Accurate value**
  0: PLL0 is unlocked on Accurate value.
  1: PLL0 is locked on Accurate value, and is ready to be selected as clock source with an accurate output clock.
- **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.
- **OSC0RDY: OSC0Ready**
  0: Oscillator not enabled or not ready.
  1: Oscillator is stable and ready to be used as clock source.

13.6.7 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.

13.6.8 PLL Control Register

Name: PLL0,1

Access Type: Read/Write

Offset: 0x001C-0x0020

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	PLLCOUNT					

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

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

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

- **PLLCOUNT: PLL Count**

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

  These field determine the ratio of the PLL output frequency (voltage controlled oscillator frequency f_{VCO}) to the source oscillator frequency:

  $f_{VCO} = (PLLMUL+1)/(PLLDIV) \cdot f_{OSC}$ if PLLDIV > 0.

  $f_{VCO} = 2*(PLLMUL+1) \cdot f_{OSC}$ if PLLDIV = 0.

  If PLLOPT[1] field is set to 0:

  $f_{PLL} = f_{VCO}$.

  If PLLOPT[1] field is set to 1:

  $f_{PLL} = f_{VCO} / 2$.

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

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

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

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

- **PLLOPT: PLL Option**

  Select the operating range for the PLL.

  PLLOPT[0]: Select the VCO frequency range.

  PLLOPT[1]: Enable the extra output divider.

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

Table 13-4. PLLOPT Fields Description

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

- **PLLOSC: PLL Oscillator Select**
  0: Oscillator 0 is the source for the PLL.
  others: Reserved.
- **PLLEN: PLL Enable**
  0: PLL is disabled.
  1: PLL is enabled.

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

13.6.9 Oscillator Control Register

Name: OSCCTRL0

Access Type: Read/Write

Offset: 0x0024

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 13-5. Startup time for oscillators 0

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

Table 13-5. Startup time for oscillators 0

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

- **AGC: Automatic Gain Control**
  For test purposes
- **GAIN: Gain**
  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 (Use in noisy environment to get better margin with respect to e.g. jitter)
- **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.

13.6.10 1.8V BOD Control Register

Name: BOD

Access Type: Read/Write

Offset: 0x0028

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: BOD Fuse Calibration Done**
  This bit is set to 1 when the CTRL, HYST and LEVEL fields have been updated by the flash fuses after a reset.
  0: The flash calibration will be redone after any reset.
  1: The flash calibration will not be redone after a BOD reset.
- **CTRL: BOD Control**

Table 13-6. Operation mode for BOD

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

13.6.11 Bandgap Calibration Register

Name: BGCR

Access Type: Read/Write

Offset: 0x002C

Reset Value: 0x00000000

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

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

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

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 power-on reset or when the Flash fuses are reprogrammed. The CALIB field will not be updated again by the Flash fuses until a new power-on reset or the FCD field is written to zero.
- **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.

13.6.12 Voltage Regulator Calibration Register

Name: VREGCR

Access Type: Read/Write

Offset: 0x0030

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
SFV	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD
15	14	13	12	11	10	9	8
-	-	-	-	-	SLEEPCALIB		
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 and SLEEPCALIB fields have been updated by the Flash fuses after power-on reset or when the Flash fuses are reprogrammed. The CALIB and SLEEPCALIB fields will not be updated again by the Flash fuses until a new power-on reset or the FCD field is written to zero.
- **SLEEPCALIB: SLEEP Calibration value**
  Calibration value for Voltage Regulator in Sleep Static and DeepStop modes, and also in Stop mode if the VREGCTRL.DMD is set to 0.
- **CALIB: Calibration value**
  Calibration value for 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.

13.6.13 Voltage Regulator Control Register

Name: VREGCTRL

Access Type: Read/Write

Offset: 0x0034

Reset Value: 0x00000000

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

- **SFV: Store Final Value**
  0: The register is read/write
  1: The register is read-only, to protect against further accidental writes.
- **DMD: Deep Mode Disable**
  0: The Voltage Regulator Deep Mode is enabled in Sleep Stop mode, the voltage Regulator enters in low-power mode to decrease the chip power consumption.
  1: The Voltage Regulator Deep Mode is disabled in Sleep Stop mode, the Voltage Regulator stays in normal mode.

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.

13.6.14 RCSYS Calibration Register

Name: RCCR

Access Type: Read/Write

Offset: 0x0038

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.

13.6.15 32KHz Oscillator Control Register

Name: OSCCTRL32

Access Type: Read/Write

Offset: 0x003C

Reset Value: 0x00000000

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

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

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

Table 13-7. Startup time for 32 KHz oscillator

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

- **MODE: Oscillator Mode**

Table 13-8. Operation mode for 32 KHz oscillator

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

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

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

13.6.16 120MHz RC Oscillator Configuration Register

Name: RC120MCR

Access Type: Read/Write

Offset: 0x0044

Reset Value: 0x00000000

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

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

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.

13.6.17 General Purpose Low-power Register 0/1

Name: GPLP0,1

Access Type: Read/Write

Offset: 0x0048, 0x004C

Reset Value: 0x00000000

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

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

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

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

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

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

Please refer to the UNLOCK register description for details.

13.6.18 Generic Clock Control

Name: GCCTRL

Access Type: Read/Write

Offset: 0x0060-0x0080

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: Generic Clock Source Selection**

Table 13-9. Generic Clock Sources

OSCSEL	Clock	Description
0	RCSYS	System RC oscillator clock
1	32 KHz clock	Output clock from OSC32
2	OSC0 out	Output clock from Oscillator 0
3	PLL0 out	Output from PLL 0
4	PLL1 out	Output from PLL 1
5	CPU clock	The clock the CPU runs on
6	HSB clock	High Speed Bus clock
7	PBA clock	Peripheral Bus A clock
8	PBB clock	Peripheral Bus B clock
9	RC120M	Output clock from Oscillator 120Mhz
10-15	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.

13.6.19 PLL Interface Version Register

Name: PLLVERSION

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.

13.6.20 Oscillator 0 Interface Version Register

Name: OSCVERSION

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.

13.6.21 1.8V BOD Interface Version Register

Name: BODVERSION

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.

13.6.22 Voltage Regulator Interface Version Register

Name: VREGVERSION

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.

13.6.23 RCSYS Interface Version Register

Name: RCCRVERSION

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.

13.6.24 32KHz Oscillator Interface Version Register

Name: OSC32VERSION

Access Type: Read-only

Offset: 0x03E8

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.

13.6.25 120MHz RC Oscillator Interface Version Register

Name: RC120MVERSION

Access Type: Read-only

Offset: 0x03F0

Reset Value: -

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

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

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

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

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

13.6.26 GPLP Version Register

Name: GPLPVERSION

Access Type: Read-only

Offset: 0x03F4

Reset Value: -

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

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

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

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

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

13.6.27 Generic Clock Interface 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.

13.6.28 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.

13.7 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 13-10. SCIF Clock Name

Module Name	Clock Name
SCIF	CLK_SCIF

Table 13-11. Register Reset Values

Register	Reset Value
PLLVERSION	0x00000101
OSCVERSION	0x00000110
BODVERSION	0x00000120
VREGVERSION	0x00000110
RCCRVERSION	0x00000110
OSC32VERSION	0x00000100
RC120MVERSION	0x00000110
GPLPVERSION	0x00000200
GCLKVERSION	0x00000101
VERSION	0x00000102

14. Asynchronous Timer (AST)

Rev: 3.1.0.1

14.1 Features

- **32-bit counter with 32-bit prescaler**
- **Clocked Source**
  - **System RC oscillator (RCSYS)**
  - **32KHz crystal oscillator (OSC32K)**
  - **PB clock**
  - **Generic clock (GCLK)**
- **Optional calendar mode supported**
- **Periodic interrupt(s) supported**
- **Alarm interrupt(s) supported**
  - **Optional clear on alarm**

14.2 Overview

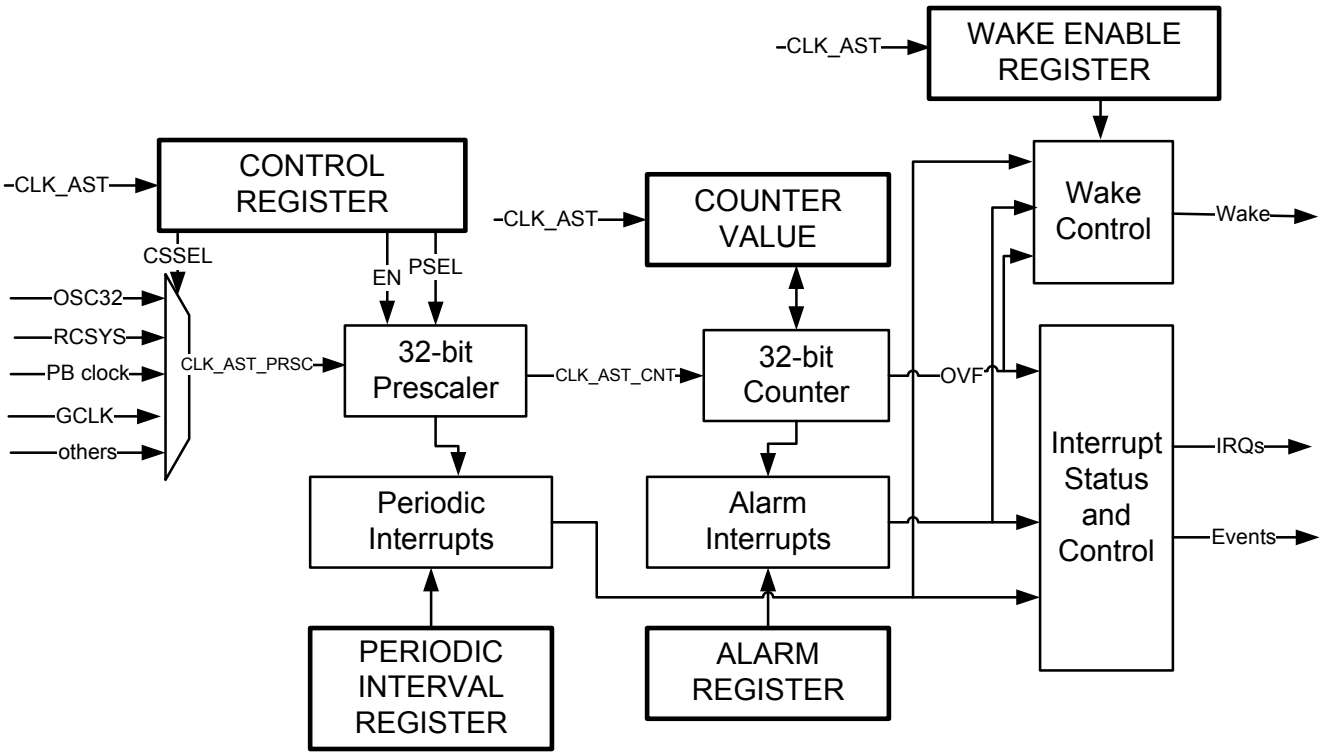
The Asynchronous Timer (AST) enables periodic interrupts, as well as interrupts at a specified time in the future. The AST consists of a 32-bit prescaler which feeds a 32-bit up-counter. The prescaler can be clocked from five different clock sources, including the low-power 32KHz oscillator, which allows 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 from output from the prescaler, as well as alarm interrupts, which can trigger at any counter value. Additionally, the timer can trigger an overflow interrupt, and be reset on the occurrence of any alarm. This allows periodic interrupts at very long and accurate intervals.

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

14.3 Block Diagram

Figure 14-1. Asynchronous Timer Block diagram



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 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.

14.4.2 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.

14.4.3 Interrupts

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

14.4.4 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.

14.5 Functional Description

14.5.1 Initialization

Before enabling the AST, the internal AST clock CLK_AST_PRSC must be enabled, following the procedure specified in Section 14.5.1.1. 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).

14.5.1.1 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

14.5.1.2 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

14.5.2 Basic Operation

14.5.2.1 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.

14.5.2.2 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 14.5.3.2 on page 184. 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.

14.5.2.3 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 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.

14.5.3 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 14.5.3.1., and the generation of the ALARM interrupt is described in Section 14.5.3.2. 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 14.5.5 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.

14.5.3.1 Periodic interrupt

The AST can generate periodic interrupts. If the PERn bit in the Interrupt Mask Register (IMR) is one, the AST will generate an interrupt request on the 0-to-1 transition of the selected bit in the prescaler when the AST is enabled. The bit is selected by the Interval Select field in the corresponding Periodic Interval Register (PIRn.INSEL), resulting in a periodic interrupt frequency of

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

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

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

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

14.5.3.2 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.

14.5.4 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 no 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.

14.5.5 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, PIRn and ARn 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 and reading will return a zero value.

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

14.6 User Interface

Table 14-1. 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
0x54	Calendar Value	CALV	Read/Write	0x00000000
0xF0	Parameter Register	PARAMETER	Read-only	-(1)
0xFC	Version Register	VERSION	Read-only	-(1)

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

14.6.1 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

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

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

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

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

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

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

14.6.2 Counter Value

Name: CV

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

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

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

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

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

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

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

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

14.6.4 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

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

Writing a zero to a bit in this register has no effect.

Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

14.6.5 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.

14.6.6 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.

14.6.7 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.

14.6.8 Wake Enable Register

Name: WER

Access Type: Read/Write

Offset: 0x1C

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	PER1	PER0

15	14	13	12	11	10	9	8
-	-	-	-	-	-	ALARM1	ALARM0

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

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

This register enables the wakeup signal from the AST.

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

14.6.9 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]							

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

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

14.6.10 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]							

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

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

14.6.11 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				

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

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

14.6.12 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				

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

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

14.6.13 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

When writing to this register, follow the sequence in Section 14.5.1 on page 181.

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

Table 14-2. Clock Source Selection

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

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

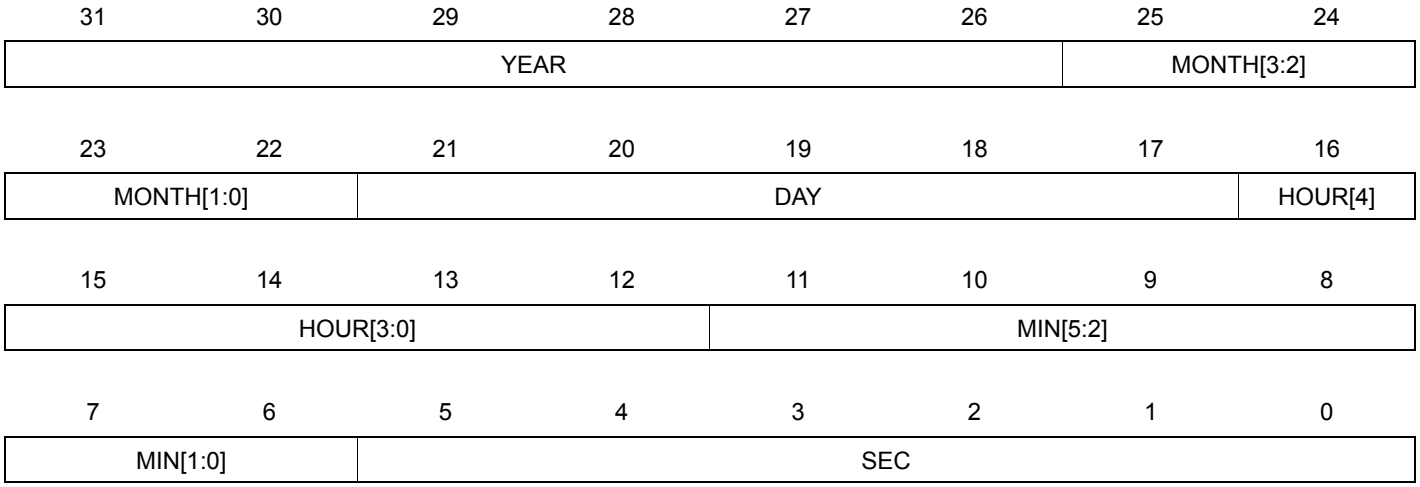
14.6.14 Calendar Value

Name: CALV

Access Type: Read/Write

Offset: 0x54

Reset Value: 0x00000000



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

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

14.6.15 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
-	-					-	-

- **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.

14.6.16 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[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.

14.7 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 14-3. Module Configuration

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

Table 14-4. AST Clocks

Clock Name	Description
CLK_AST	Clock for the AST bus interface
GCLK_AST	The generic clock used for the AST is GCLK6
PB clock	Peripheral Bus Clock, PB clock = CLK_AST

Table 14-5. Register Reset Values

Register	Reset Value
VERSION	0x00000310
PARAMETER	0x00004100

15. Watchdog Timer (WDT)

Rev: 4.1.0.0

15.1 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**

15.2 Overview

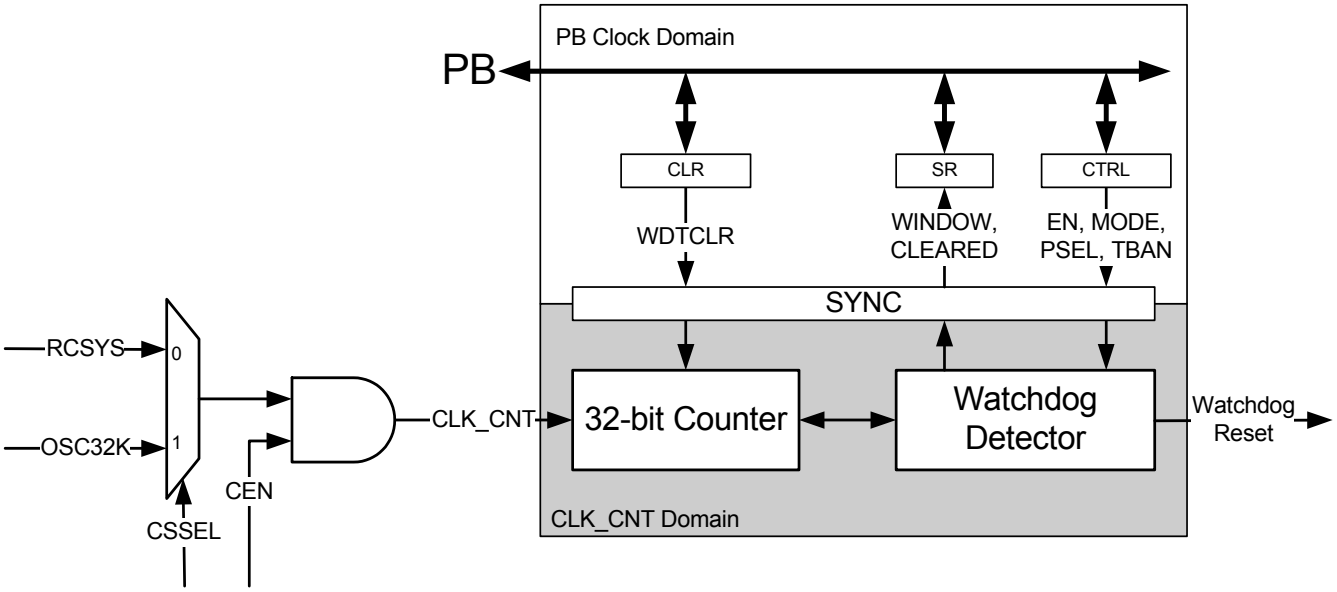
The Watchdog Timer (WDT) will reset the device unless it is periodically serviced by the software. This allows the device to recover from a condition that has caused the system to be unstable.

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.

15.3 Block Diagram

Figure 15-1. WDT 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 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 device 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.

15.4.2 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.

15.4.3 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.

15.4.4 Fuses

The WDT can be enabled at reset. This is controlled by the WDTAUTO fuse, see Section 15.5.4 for details. Please refer to the Fuse Settings section in the Flash Controller chapter for details about WDTAUTO and how to program the fuses.

15.5 Functional Description

15.5.1 Basic Mode

15.5.1.1 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.

15.5.1.2 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.

15.5.1.3 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:

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

15.5.1.4 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.

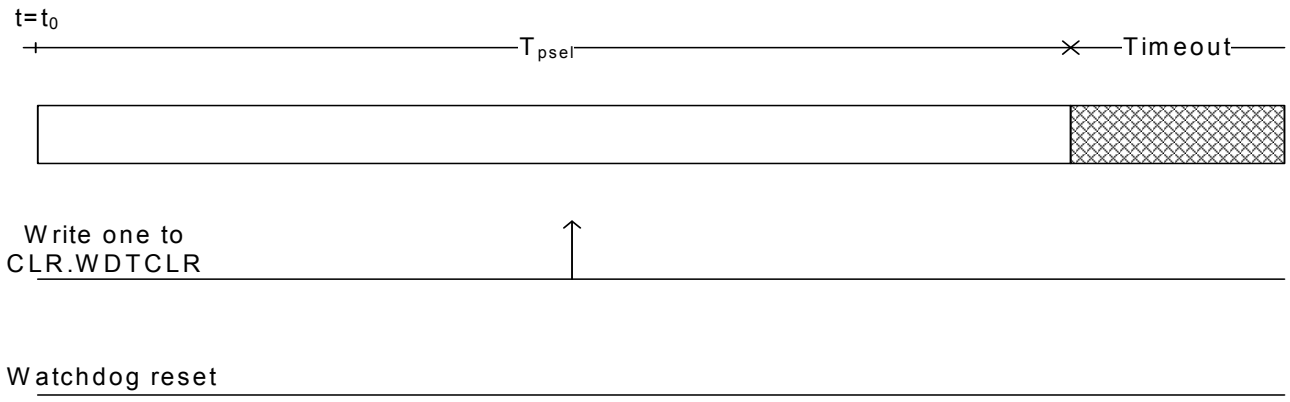
15.5.1.5 Clearing the WDT Counter

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 device is reset. In basic mode the CLR.WDTCLR can be written at any time when the WDT Counter Cleared (CLEARED) bit in the Status Register (SR) is one. Due to internal synchronization, clearing the WDT counter takes some time. The SR.CLEARED bit is cleared when writing to CLR.WDTCLR bit and set when the clearing is done. Any write to the CLR.WDTCLR bit while SR.CLEARED is zero will not clear the counter.

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

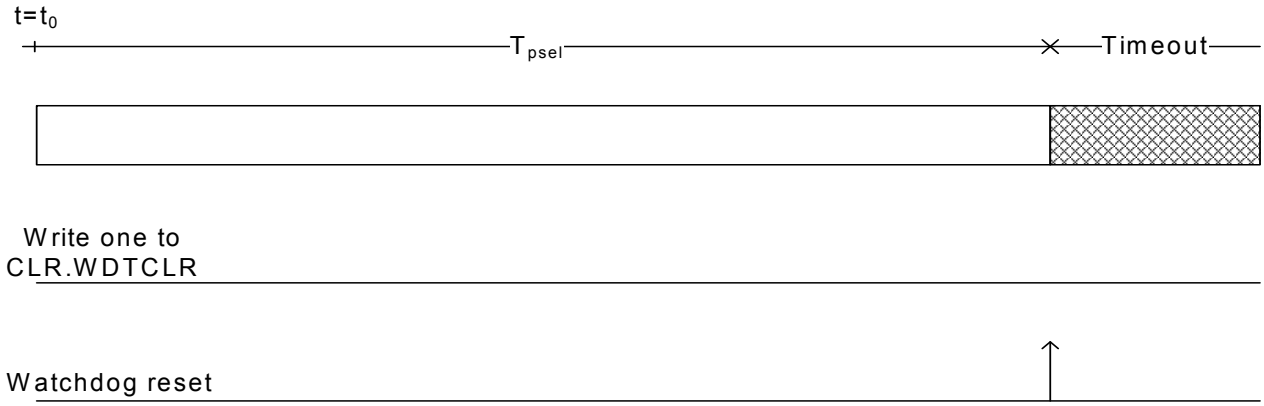
If the WDT counter is periodically cleared within T_{psel} no watchdog reset will be issued, see Figure 15-2 on page 206.

Figure 15-2. 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 15-3 on page 206.

Figure 15-3. Basic Mode WDT Timing Diagram, no clear within T_{psel}.



15.5.1.6 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.

15.5.2 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 15-5 on page 207. 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 15-4 on page 207.

If the value in the CTRL Register is changed, the WDT counter will be cleared without a watch-dog 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 15-4. Window Mode WDT Timing Diagram

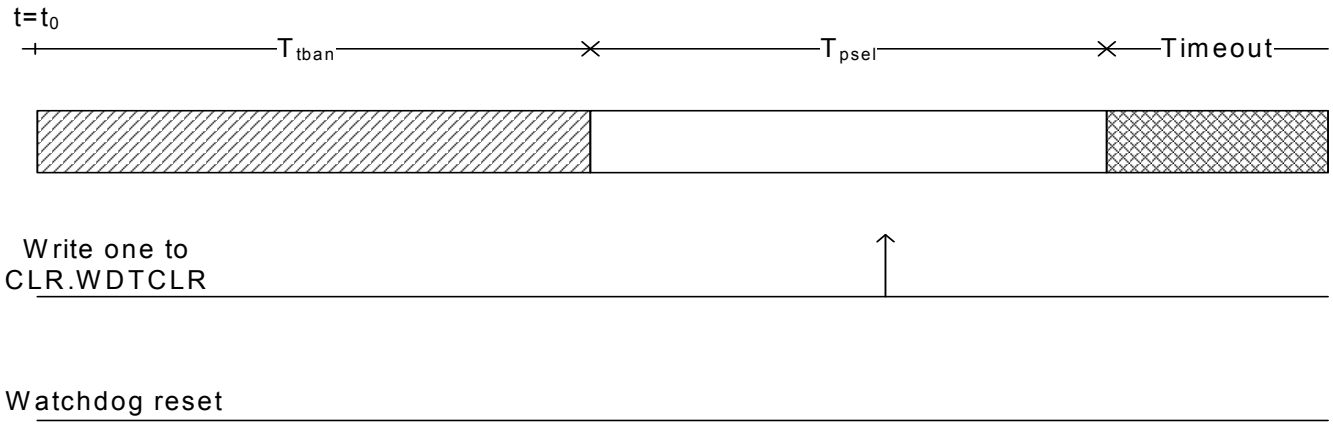
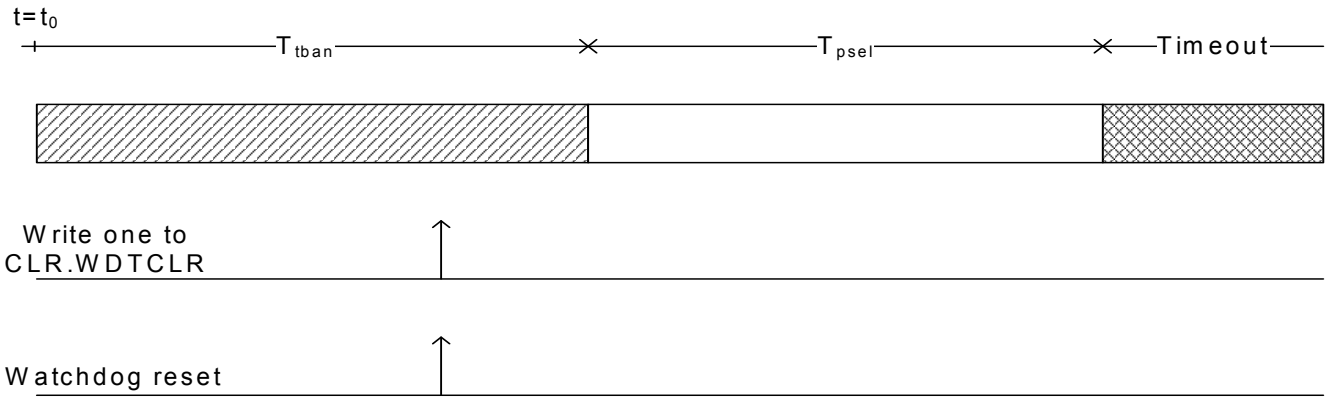


Figure 15-5. Window Mode WDT Timing Diagram, clearing within T_{tban}, resulting in watchdog reset.



15.5.3 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.

15.5.4 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.

15.5.5 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 device. Otherwise a watchdog reset will reset the device 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).

15.6 User Interface

Table 15-1. 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.

15.6.1 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.

15.6.2 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.

15.6.3 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.

15.6.4 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.

15.7 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 15-2. Module clock name

Module name	Clock name
WDT	CLK_WDT

Table 15-3. Register Reset Values

Register	Reset Value
VERSION	0x00000410

16. External Interrupt Controller (EIC)

Rev: 3.0.2.0

16.1 Features

- **Dedicated interrupt request for each interrupt**
- **Individually maskable interrupts**
- **Interrupt on rising or falling edge**
- **Interrupt on high or low level**
- **Asynchronous interrupts for sleep modes without clock**
- **Filtering of interrupt lines**
- **Non-Maskable NMI interrupt**

16.2 Overview

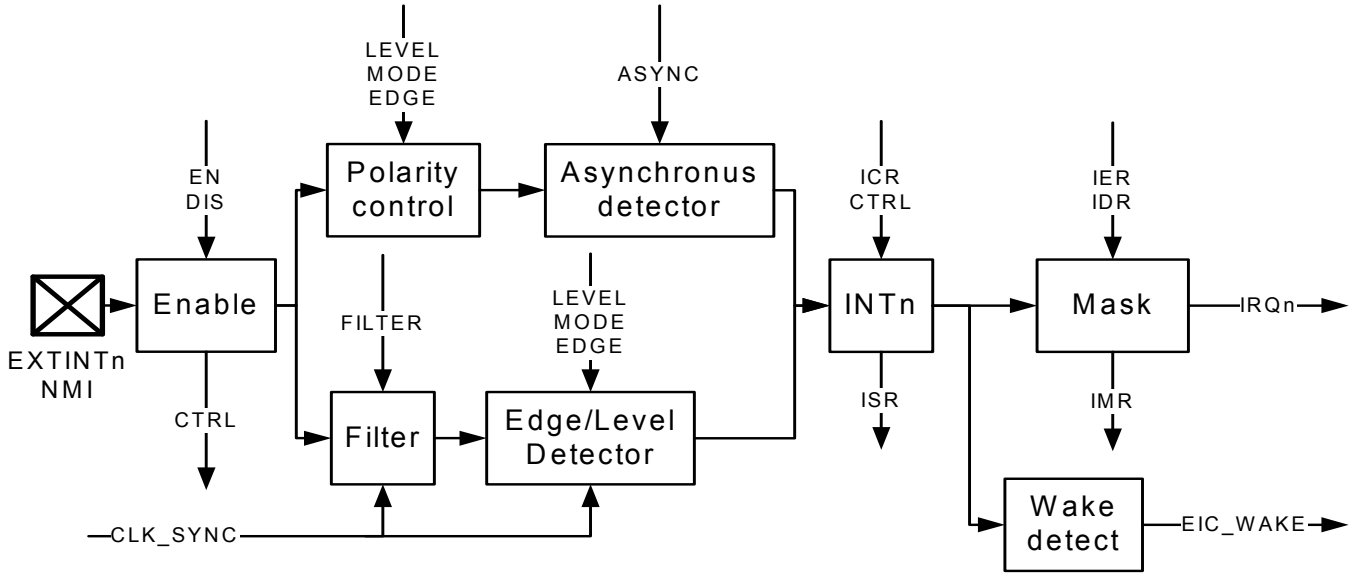
The External Interrupt Controller (EIC) allows pins to be configured as external interrupts. Each external interrupt has its own interrupt request and can be individually masked. Each external interrupt can generate an interrupt on rising or falling edge, or high or low level. Every interrupt input has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

16.3 Block Diagram

Figure 16-1. EIC Block Diagram



16.4 I/O Lines Description

Table 16-1. I/O Lines Description

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

16.5 Product Dependencies

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

16.5.1 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.

16.5.2 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.

16.5.3 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.

16.5.4 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.

16.5.5 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.

16.6 Functional Description

16.6.1 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.

16.6.2 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 16-2 and Figure 16-3 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 16-2 and Figure 16-3 for examples (FILTER on).

Figure 16-2. Timing Diagram, Synchronous Interrupts, High Level or Rising Edge

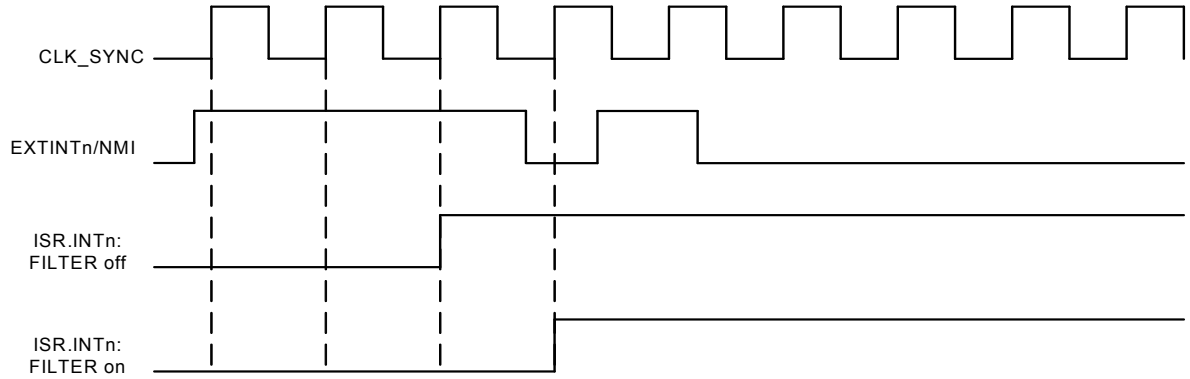
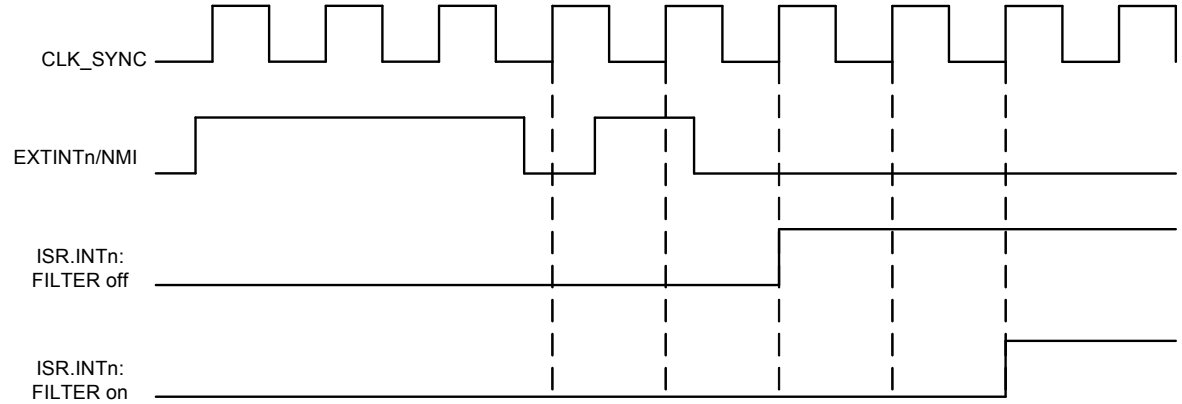


Figure 16-3. Timing Diagram, Synchronous Interrupts, Low Level or Falling Edge



16.6.3 Non-Maskable Interrupt

The NMI supports the same features as the external interrupts, and is accessed through the same registers. The description in Section 16.6.1 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.

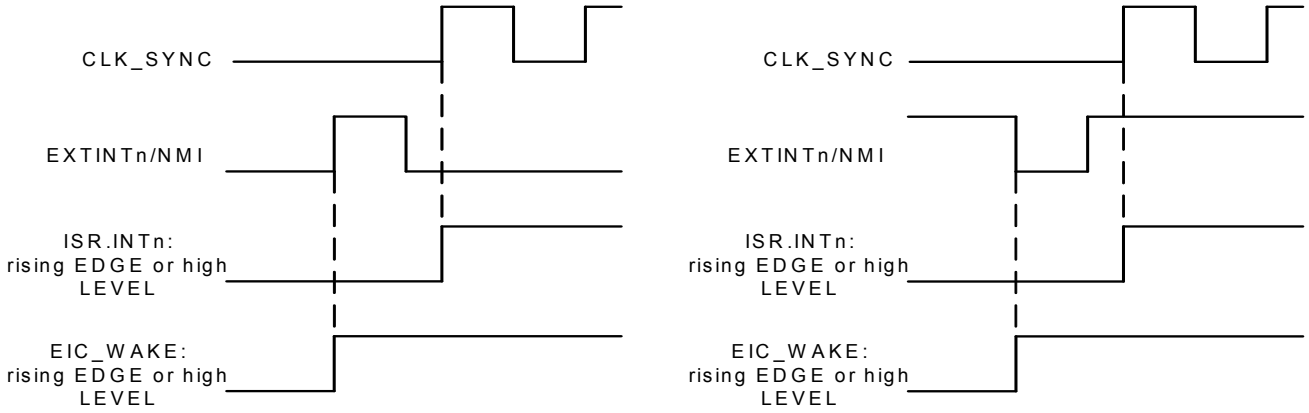
16.6.4 Asynchronous Interrupts

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 16-4 on page 220 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 16-4. Timing Diagram, Asynchronous Interrupts



16.6.5 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.

16.7 User Interface

Table 16-2. EIC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Interrupt Enable Register	IER	Write-only	0x00000000
0x004	Interrupt Disable Register	IDR	Write-only	0x00000000
0x008	Interrupt Mask Register	IMR	Read-only	0x00000000
0x00C	Interrupt Status Register	ISR	Read-only	0x00000000
0x010	Interrupt Clear Register	ICR	Write-only	0x00000000
0x014	Mode Register	MODE	Read/Write	0x00000000
0x018	Edge Register	EDGE	Read/Write	0x00000000
0x01C	Level Register	LEVEL	Read/Write	0x00000000
0x020	Filter Register	FILTER	Read/Write	0x00000000
0x024	Test Register	TEST	Read/Write	0x00000000
0x028	Asynchronous Register	ASYNC	Read/Write	0x00000000
0x030	Enable Register	EN	Write-only	0x00000000
0x034	Disable Register	DIS	Write-only	0x00000000
0x038	Control Register	CTRL	Read-only	0x00000000
0x3FC	Version Register	VERSION	Read-only	- (1)

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

16.7.1 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.

16.7.2 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.

16.7.3 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.

16.7.4 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.

16.7.5 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.

16.7.6 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.

16.7.7 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.

16.7.8 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.

16.7.9 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.

16.7.10 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.

16.7.11 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.

16.7.12 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.

16.7.13 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.

16.7.14 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.

16.7.15 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.

16.8 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 16-3. Module Configuration

Feature	EIC
Number of external interrupts, including NMI	9

Table 16-4. Module Clock Name

Module Name	Clock Name
EIC	CLK_EIC

Table 16-5. Register Reset Values

Register	Reset Value
VERSION	0x00000302

17. Frequency Meter (FREQM)

Rev: 3.1.0.1

17.1 Features

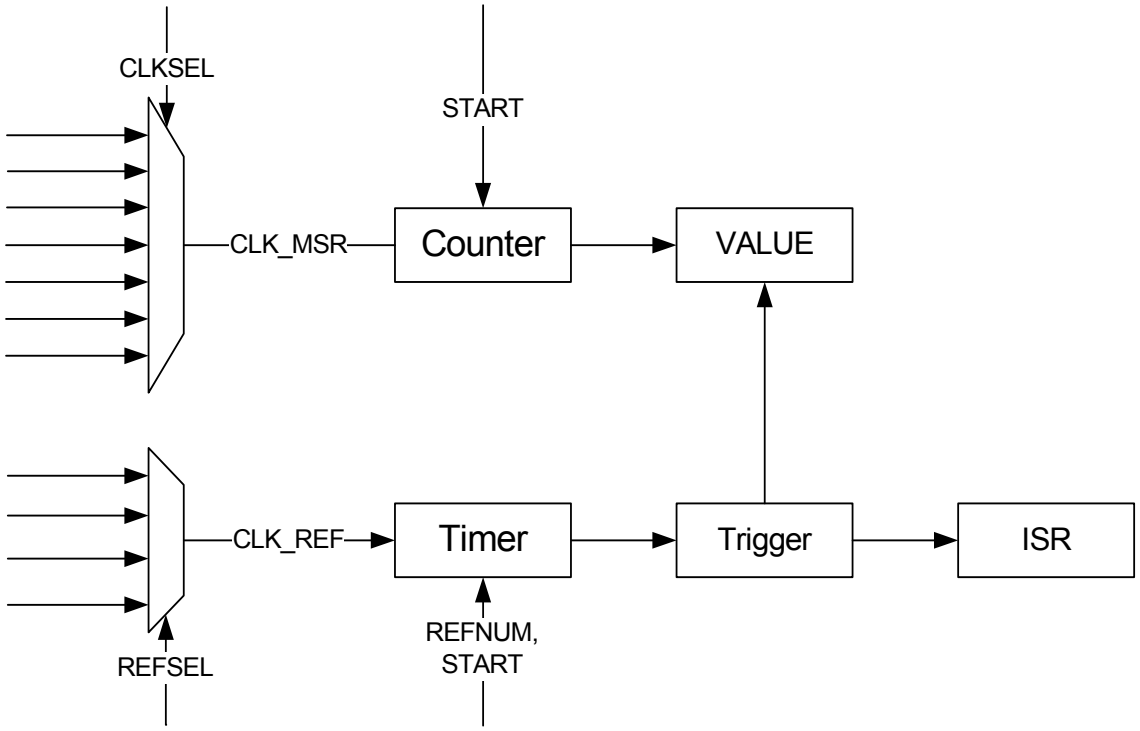
- **Accurately measures a clock frequency**
- **Selectable reference clock**
- **A selectable clock can be measured**
- **Ratio can be measured with 24-bit accuracy**

17.2 Overview

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

17.3 Block Diagram

Figure 17-1. Frequency Meter Block Diagram



17.4 Product Dependencies

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

17.4.1 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.

17.4.2 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.

17.4.3 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.

17.4.4 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.

17.5 Functional Description

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

$$f_{CLK\_MSR} = (VALUE/REFNUM)*f_{CLK\_REF}$$

17.5.1 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.

17.5.1.1 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.

17.5.2 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:

$$f_{CLK\_MSR} = (VALUE/REFNUM)*f_{CLK\_REF}$$

17.5.3 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.

17.6 User Interface

Table 17-1. 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.

17.6.1 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.

17.6.2 Mode Register

Name: MODE

Access Type: Read/Write

Offset: 0x004

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
REFNUM							

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

- **REFCEN: Reference Clock Enable**
  0: The reference clock is disabled
  1: The reference clock is enabled
- **CLKSEL: Clock Source Selection**
  Selects the source for CLK_MSR. See table in Module Configuration chapter for details.
- **REFNUM: Number of Reference Clock Cycles**
  Selects the duration of a measurement, given in number of CLK_REF cycles.
- **REFSEL: Reference Clock Selection**
  Selects the source for CLK_REF. See table in Module Configuration chapter for details.

17.6.3 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.

17.6.4 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.

17.6.5 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.

17.6.6 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.

17.6.7 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.

17.6.8 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.

17.6.9 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.

17.6.10 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.

17.7 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 17-2. Module Clock Name

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

Table 17-3. Register Reset Values

Register	Reset Value
VERSION	0x00000310

Table 17-4. 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	PLL0	Output clock from PLL0
8	PLL1	Output clock from PLL1
10-18	GCLK0-8	Generic clocks
19	RC120M AW clock	Output clock from RC120M to AW
20-31	Reserved	

Table 17-5. Clock Sources for CLK_REF

REFSEL	Clock/Oscillator	Description
0	RCSYS	System RC oscillator clock
1	OSC32K	Output clock from OSC32K
2	OSC0	Output clock from Oscillator O
3	GCLK0	Generic Clock 0
4-7	Reserved	

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

Rev: 2.1.2.5

18.1 Features

- **Configurable pin-change, rising-edge, or falling-edge interrupt**
- **Glitch filter providing rejection of pulses shorter than one clock cycle**
- **Input visibility and output control**
- **Multiplexing of peripheral functions on I/O pins**
- **Programmable internal pull-up resistor**
- **Optional locking of configuration to avoid accidental reconfiguration**

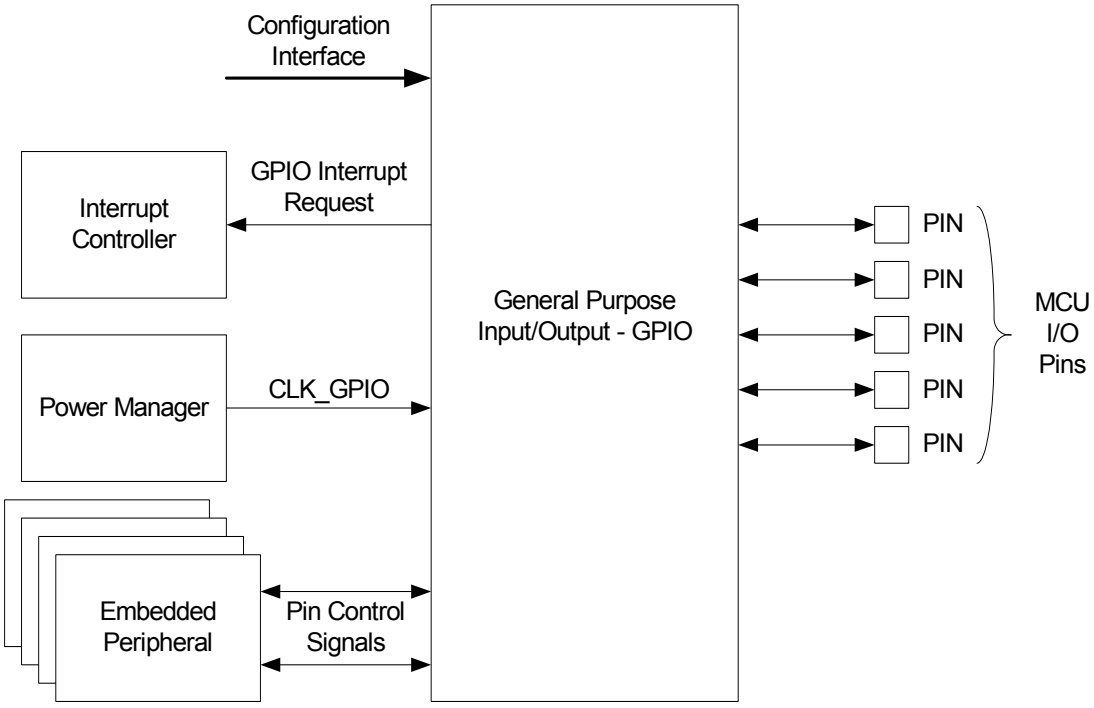
18.2 Overview

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 18.6.2.7 for details.

18.3 Block Diagram

Figure 18-1. GPIO Block Diagram



18.4 I/O Lines Description

Pin Name	Description	Type
GPIOn	GPIO pin n	Digital

18.5 Product Dependencies

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

18.5.1 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.

18.5.2 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.

18.5.3 Interrupts

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

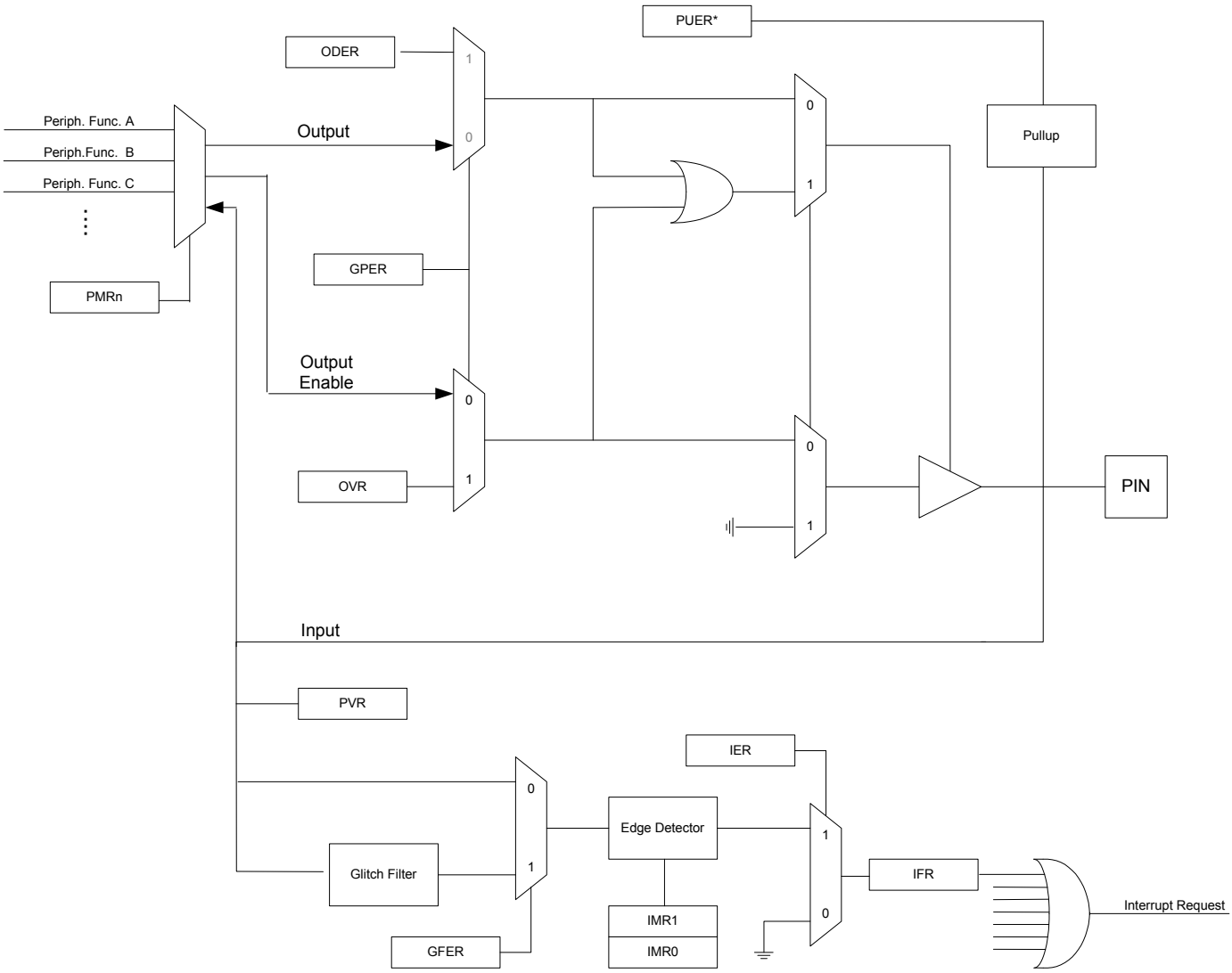
18.5.4 Debug Operation

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

18.6 Functional Description

The GPIO controls the I/O pins of the microcontroller. The control logic associated with each pin is shown in the figure below.

Figure 18-2. Overview of the GPIO



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

18.6.1 Basic Operation

18.6.1.1 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 18.7.

18.6.1.2 Available Features

The GPIO features implemented are device dependent, and not all functions are implemented on all pins. The user must refer to the Module Configuration section and the GPIO Function Multiplexing section in the Package and Pinout chapter for the device specific settings used in the UC3D.

Device specific settings includes:

- Number of GPIO pins
- Functions implemented on each pin
- Peripheral function(s) multiplexed on each GPIO pin
- Reset state of registers

18.6.1.3 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.

18.6.1.4 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).

18.6.1.5 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.

18.6.2 Advanced Operation

18.6.2.1 *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 18-1, 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 18-1. 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

18.6.2.2 *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.

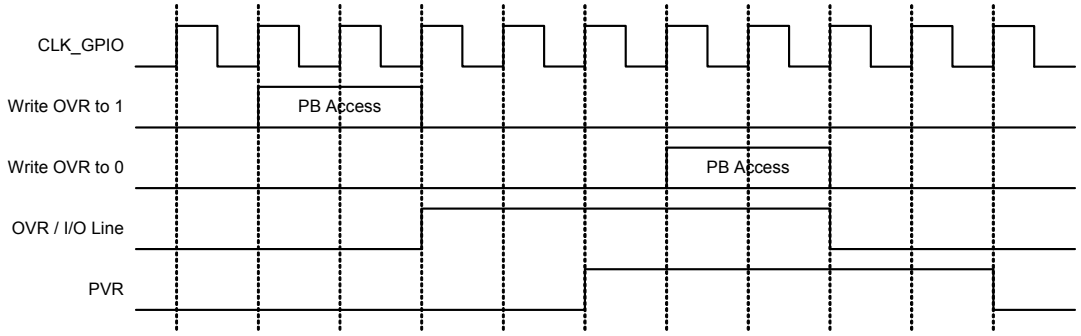
Pull-up is useful for detecting if a pin is unconnected or if a mechanical button is pressed, for various communication protocols and to keep unconnected pins from floating.

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).

18.6.2.3 *Output Pin Timings*

Figure 18-3 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 18-3. Output Pin Timings



18.6.2.4 *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.

18.6.2.5 *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.

18.6.2.6 *Interrupt Timings*

Figure 18-4 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 18-4. Interrupt Timing with Glitch Filter Disabled

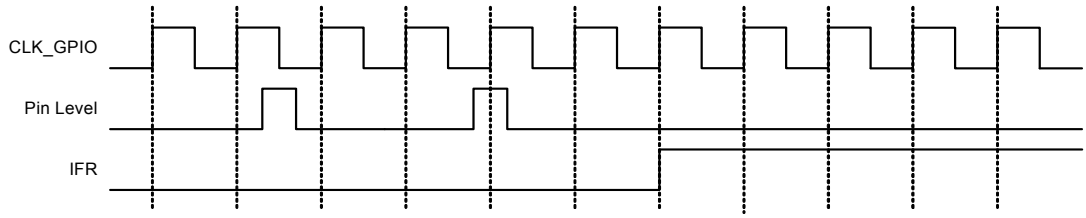
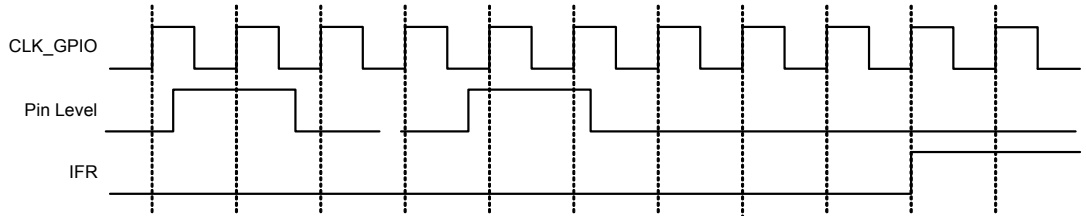


Figure 18-5 shows the timing for rising edge (or pin-change) interrupts when the glitch filter is enabled. For the pulse to be registered, it must be sampled on two subsequent rising edges. In the example, the first pulse is rejected while the second pulse is accepted and causes an interrupt request.

Figure 18-5. Interrupt Timing with Glitch Filter Enabled



18.6.2.7 *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).

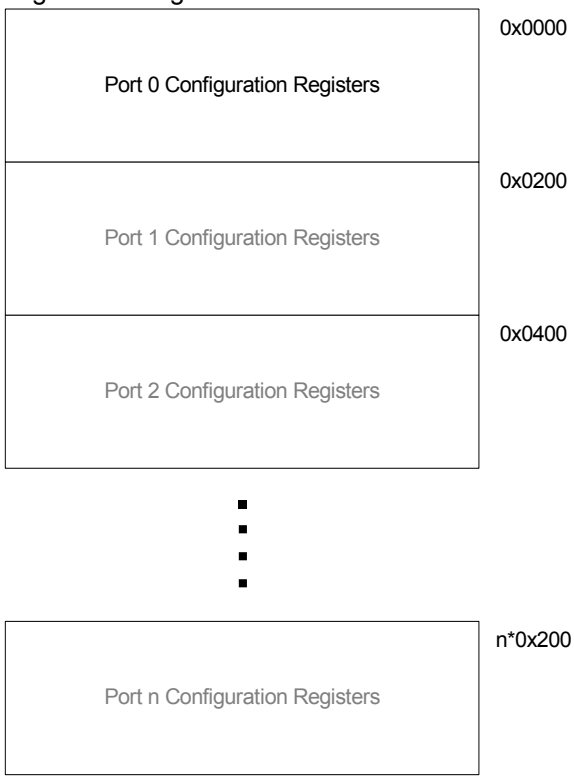
To avoid data loss when using the CPU Local Bus, the CLK_GPIO must run at the same frequency as the CLK_CPU. See Section 18.5.2 for details.

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.

18.7 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 18-6. 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 18-2 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.

18.7.1 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.

18.7.2 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 18.7.3, ensuring the LOCK mechanism itself is robust against software failure.

18.7.3 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 18-2. 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 18-2. 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 18-2. 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 18-2. 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
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.

18.7.4 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.

18.7.5 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**

18.7.6 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**

18.7.7 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

18.7.8 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.

18.7.9 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.

18.7.10 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.

18.7.11 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.

18.7.12 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.

18.7.13 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**

18.7.14 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

18.7.15 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.

18.7.16 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.

18.7.17 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.

18.7.18 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.

18.7.19 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.

18.7.20 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.

18.7.21 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.

18.8 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 18-3. Module Configuration

Feature	GPIO
Number of GPIO ports	2
Number of peripheral functions	4

Table 18-4. Implemented Pin Functions

Pin Function	Implemented	Notes
Pull-up	On all pins	Controlled by PUER or peripheral
Glitch Filter	No	

Table 18-5. Module Clock Name

Module Name	Clock Name
GPIO	**CLK_GPIO**

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

Table 18-6. Register Reset Values

Port	Register	Reset Value
0	GPER	0xFFFFFFFF
0	PUER	0x00000000
0	PARAMETER	0xFFFFFFFF
0	VERSION	0x00000212
1	GPER	0x0007FFFF
1	PUER	0x00001000
1	PARAMETER	0x0007FFFF
1	VERSION	0x00000212

19. USB Interface (USBC)

Rev: 2.0.0.15

19.1 Features

- **Compatible with the USB 2.0 specification**
- **Supports full (12Mbit/s) and low (1.5Mbit/s) speed communication**
- **7 physical pipes/endpoints in ping-pong mode**
- **Flexible pipe/endpoint configuration and reallocation of data buffers in embedded RAM**
- **Up to two memory banks per pipe/endpoint**
- **Built-in DMA with multi-packet support through ping-pong mode**
- **On-chip transceivers with built-in pull-ups and pull-downs**

19.2 Overview

The Universal Serial Bus interface (USBC) module complies with the Universal Serial Bus (USB) 2.0 specification.

Each pipe/endpoint can be configured into one of several transfer types. It can be associated with one or more memory banks (located inside the embedded system or CPU RAM) used to store the current data payload. If two banks are used (“ping-pong” mode), then one bank is read or written by the CPU (or any other HSB master) while the other is read or written by the USBC core.

Table 19-1 describes the hardware configuration of the USBC module.

Table 19-1. Description of USB pipes/endpoints

pipe/endpoint	Mnemonic	Max. size	Number of available banks	Type
0	PEP0	1023 bytes	1	Control/Isochronous/Bulk/Interrupt
1	PEP1	1023 bytes	2	Control/Isochronous/Bulk/Interrupt
2	PEP2	1023 bytes	2	Control/Isochronous/Bulk/Interrupt
...	...	...	...	...
6	PEP6	1023 bytes	2	Control/Isochronous/Bulk/Interrupt

19.3 Block Diagram

The USBC interfaces a USB link with a data flow stored in the embedded ram (CPU or HSB).

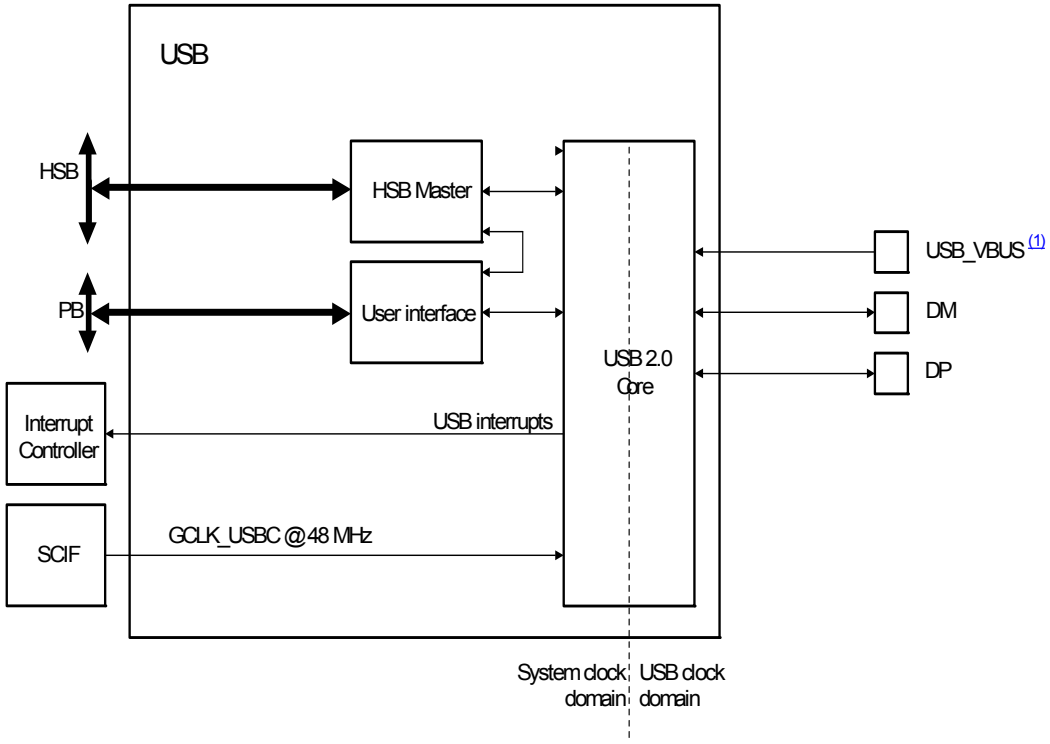
The USBC requires a 48MHz ± 0.25% reference clock, which is the USB generic clock. For more details see ”Clocks” on page 286. The 48MHz clock is used to generate either a 12MHz full-speed or a 1.5MHz low-speed bit clock from the received USB differential data, and to transmit data according to full- or low-speed USB device tolerances. Clock recovery is achieved by a digital phase-locked loop (a DPLL, not represented) in the USBC module, which complies with the USB jitter specifications.

The USBC module consists of:

- HSB master interface

- User interface
- USB Core
- Transceiver pads

Figure 19-1. USBC Block Diagram



Note: is 5V tolerant

19.4 I/O Lines Description

Table 19-2. I/O Lines Description

PIn Name	Pin Description	Type	Active Level
USB_VBUS	VBUS: Bus Power Measurement Port	Input	
DM	Data -: Differential Data Line - Port	Input/Output	
DP	Data +: Differential Data Line + Port	Input/Output	

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 I/O Lines

The USBC pins may be multiplexed with the I/O Controller lines. The user must first configure the I/O Controller to assign the desired USBC pins to their peripheral functions.

19.5.2 Power Management

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

19.5.3 Clocks

The USBC has two bus clocks connected: One High Speed Bus clock (CLK_USBC_HSB) and one Peripheral Bus clock (CLK_USBC_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by the Power Manager. It is recommended to disable the USBC before disabling the clocks, to avoid freezing the USBC in an undefined state.

The 48 MHz USB clock is generated by a dedicated generic clock from the SCIF module. Before using the USB, the user must ensure that the USB generic clock (GCLK_USBC) is enabled at 48 MHz in the SCIF module.

19.5.4 Interrupts

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

The USBC asynchronous interrupts can wake the CPU from any sleep mode:

- The VBUS Transition Interrupt (VBUSTI)
- The Wakeup Interrupt (WAKEUP)

19.6 Functional Description

19.6.1 USB General Operation

19.6.1.1 Initialization

After a hardware reset, the USBC is in the Reset state. In this state:

- The module is disabled. The USBC Enable bit in the General Control register (USBCON.USBE) is reset.
- The module clock is stopped in order to minimize power consumption. The Freeze USB Clock bit in USBCON (USBCON.FRZCLK) is set.
- The USB pad is in suspend mode.
- The internal states and registers of the device are reset.
- The VBUS Level bit (USBSTA.VBUS) reflects the states of the USB_VBUS input pins.
- The Freeze USB Clock (FRZCLK), USBC Enable (USBE), in USBCON and the Low-Speed mode bit in the Device General Control register (UDCON.LS) can be written to by software, so that the user can configure pads and speed before enabling the module. These values are only taken into account once the module has been enabled and unfrozen.

After writing a one to USBCON.USBE, the USBC enters device mode in idle state.

Refer to Section 19.6.2 for the basic operation of the device mode.

The USBC can be disabled at any time by writing a zero to USBCON.USBE, this acts as a hardware reset, except that the FRZCLK,bit in USBCON, and the LS bits in UDCON are not reset.

19.6.1.2 Interrupts

One interrupt vector is assigned to the USBC.

See Section 19.6.2.18 for further details about device interrupts.

See Section 19.5.4 for asynchronous interrupts.

19.6.1.3 Frozen clock

When the USB clock is frozen, it is still possible to access the following bits: FRZCLK, and USBE in the USBCON register, and LS in the UDCON register.

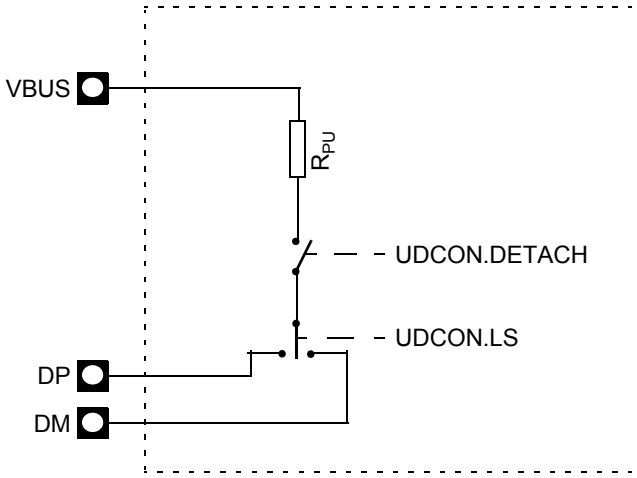
When FRZCLK is set, only the asynchronous interrupts can trigger a USB interrupt (see Section 19.5.4).

19.6.1.4 Speed control

- *Device mode*

When the USBC interface is in device mode, the speed selection is done by the UDCON.LS bit, connecting an internal pull-up resistor to either DP (full-speed mode) or DM (low-speed mode). The LS bit shall be written before attaching the device, which can be simulated by clearing the UDCON.DETACH bit.

Figure 19-2. Speed Selection in device mode



19.6.1.5 Data management

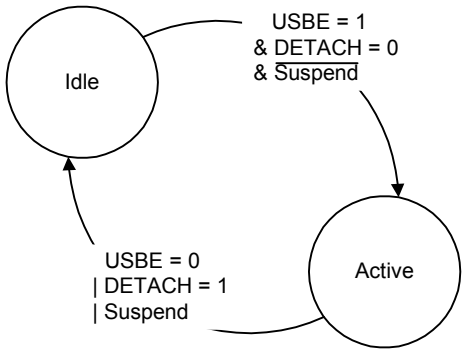
Endpoints and pipe buffers can be allocated anywhere in the embedded memory (CPU RAM or HSB RAM).

See ”RAM management” on page 292.

19.6.1.6 Pad Suspend

Figure 19-3 illustrates the behavior of the USB pad in device mode.

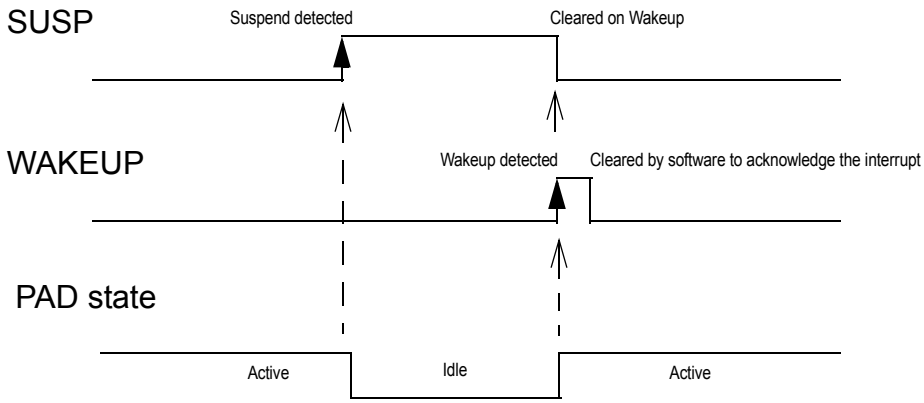
Figure 19-3. Pad Behavior



- In Idle state, the pad is in low power consumption mode.
- In Active state, the pad is working.

Figure 19-4 illustrates the pad events leading to a PAD state change.

Figure 19-4. Pad events



The Suspend Interrupt bit in the Device Global Interrupt register (UDINT.SUSP) is set and the Wakeup Interrupt (UDINT.WAKEUP) bit is cleared when a USB Suspend state has been detected on the USB bus. This event automatically puts the USB pad in the Idle state. The detection of a non-idle event sets WAKEUP, clears SUSP, and wakes the USB pad.

The pad goes to the Idle state if the module is disabled or if UDCON.DETACH is written to one. It returns to the Active state when USBCON.USBE is written to one and DETACH is written to zero.

19.6.2 USBC Device Mode Operation

19.6.2.1 Device Enabling

In device mode, the USBC supports full- and low-speed data transfers.

Including the default control endpoint, a total of None endpoints are provided. They can be configured as isochronous, bulk or interrupt types, as described in Table 19-1 on page 283

After a hardware reset, the USBC device mode is in the reset state (see Section 19.6.1.1). In this state, the endpoint banks are disabled and neither DP nor DM are pulled up (DETACH is one).

DP or DM will be pulled up according to the selected speed as soon as the DETACH bit is written to zero and VBUS is present. See "Device mode" for further details.

When the USBC is enabled (USBE is one) in device mode, it enters the Idle state, minimizing power consumption. Being in Idle state does not require the USB clocks to be activated.

The USBC device mode can be disabled or reset at any time by disabling the USBC (by writing a zero to USBE).

19.6.2.2 USB reset

The USB bus reset is initiated by a connected host and managed by hardware.

When a USB reset state is detected on the USB bus, the following operations are performed by the controller:

- UDCON register is reset except for the DETACH and SPDCONF bits.
- Device Frame Number Register (UDFNUM), Endpoint n Configuration Register (UECFGn), and Endpoint n Control Register (UECONn) registers are cleared.
- The data toggle sequencing in all the endpoints are cleared.
- At the end of the reset process, the End of Reset (EORST) bit in the UDINT register is set.

19.6.2.3 Endpoint activation

When an endpoint is disabled (UERST.EPENn = 0) the data toggle sequence, Endpoint n Status Set (UESTAn), and UECONn registers will be reset. The controller ignores all transactions to this endpoint as long as it is inactive.

To complete an endpoint activation, the user should fill out the endpoint descriptor: see Figure 19-5 on page 293.

19.6.2.4 Data toggle sequence

In order to respond to a CLEAR_FEATURE USB request without disabling the endpoint, the user can clear the data toggle sequence by writing a one to the Reset Data Toggle Set bit in the Endpoint n Control Set register (UECONnSET.RSTDTS)

19.6.2.5 Busy bank enable

In order to make an endpoint bank look busy regardless of its actual state, the user can write a one to the Busy Bank Enable bit in the Endpoint n Control Register (UECONnSET.BUSY0/1ES).

If a BUSYnE bit is set, any transaction to this bank will be rejected with a NAK reply.

19.6.2.6 Address setup

The USB device address is set up according to the USB protocol.

- After all kinds of resets, the USB device address is 0.
- The host starts a SETUP transaction with a SET_ADDRESS(addr) request.
- The user writes this address to the USB Address field (UDCON.UADD), and writes a zero to the Address Enable bit (UDCON.ADDEN), resulting in the address remaining zero.
- The user sends a zero-length IN packet from the control endpoint.
- The user enables the stored USB device address by writing a one to ADDEN.

Once the USB device address is configured, the controller filters the packets to only accept those targeting the address stored in UADD.

UADD and ADDEN should not be written to simultaneously. They should be written sequentially, UADD field first.

If UADD or ADDEN is cleared, the default device address 0 is used. UADD and ADDEN are cleared:

- On a hardware reset.
- When the USBC is disabled (USBE written to zero).
- When a USB reset is detected.

19.6.2.7 *Suspend and Wakeup*

When an idle USB bus state has been detected for 3 ms, the controller sets the Suspend (SUSP) interrupt bit in UDINT. In this case, the transceiver is suspended, reducing power consumption.

To further reduce power consumption it is recommended to freeze the USB clock by writing a one to the Freeze USB Clock (FRZCLK) bit in USBCON when the USB bus is in suspend mode. The MCU can also enter the idle or frozen sleep mode to further lower power consumption.

To recover from the suspend mode, the user shall wait for the Wakeup (WAKEUP) interrupt bit, which is set when a non-idle event is detected, and then write a zero to FRZCLK.

As the WAKEUP interrupt bit in UDINT is set when a non-idle event is detected, it can occur regardless of whether the controller is in the suspend mode or not. The SUSP and WAKEUP interrupts are thus independent of each other except for that one bit is cleared when the other is set.

19.6.2.8 *Detach*

The reset value of the DETACH bit located in the UDCON register, is one.

It is possible to initiate a device re-enumeration simply by writing a one and then a zero to DETACH.

DETACH acts on the pull-up connections of the DP and DM pads. See “Device mode” for further details.

19.6.2.9 *Remote wakeup*

The remote wakeup request (also known as upstream resume) is the only request the device may send on its own initiative. This should be preceded by a DEVICE_REMOTE_WAKEUP request from the host.

- First, the USBC must have detected a “Suspend” state on the bus, i.e. the remote wakeup request can only be sent after a SUSP interrupt has been set.

- The user may then write a one to the remote wakeup (RMWKUP) bit in UDCON to send an Upstream Resume to the host initiating the wakeup. This will automatically be done by the controller after 5ms of inactivity on the USB bus.
- When the controller sends the Upstream Resume, the Upstream Resume (UPRSM) interrupt is set and SUSP is cleared.
- RMWKUP is cleared at the end of the transmitting Upstream Resume.
- In case of a rebroadcast resume initiated by the host, the End of Resume (EORSM) interrupt is set when the rebroadcast resume is completed.

19.6.2.10 RAM management

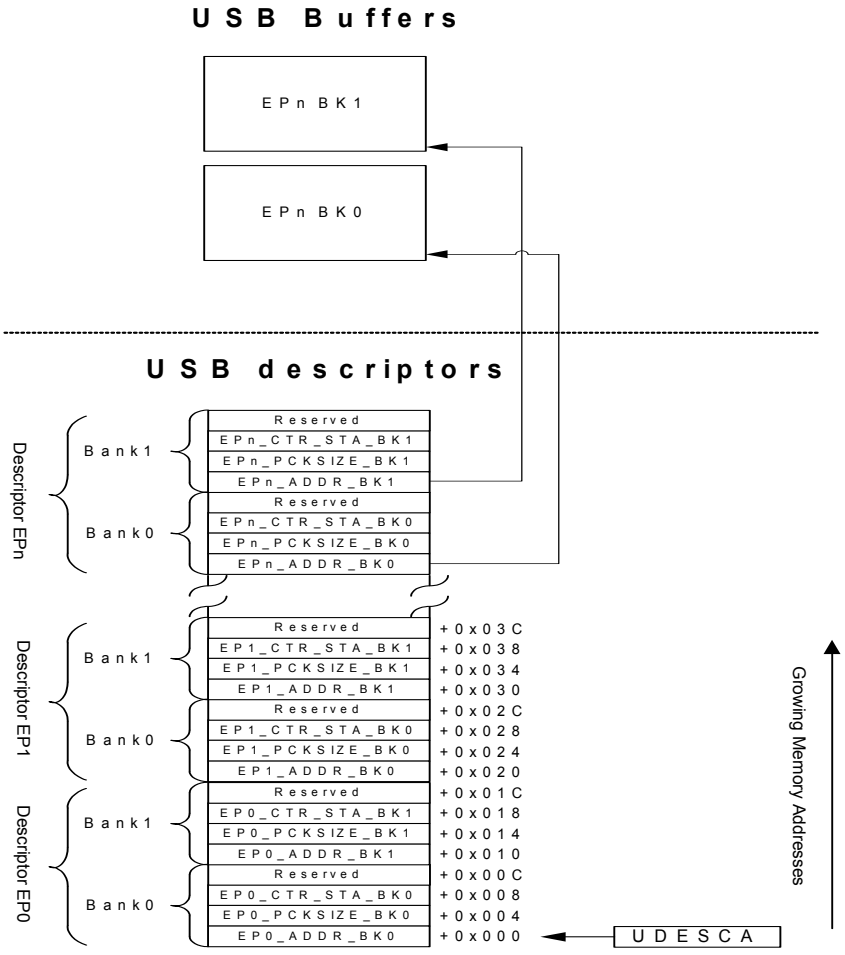
Endpoint data can be physically allocated anywhere in the embedded RAM. The USBC controller accesses these endpoints directly through the HSB master (built-in DMA).

The USBC controller reads the USBC descriptors to know where each endpoint is located. The base address of the USBC descriptor (UDESC.UDESCA) needs to be written by the user. The descriptors can also be allocated anywhere in the embedded RAM.

Before using an endpoint, the user should setup the endpoint address for each bank. Depending on the direction, the type, and the packet-mode (single or multi-packet), the user should also initialize the endpoint packet size, and the endpoint control and status fields, so that the USBC controller does not compute random values from the RAM.

When using an endpoint the user should read the UESTAX.CURRBK field to know which bank is currently being processed.

Figure 19-5. Memory organization



Each descriptor of an endpoint n consists of four words.

- The address of the endpoint and the bank used (EPn_ADDR_BK0/1).
- The packet size information for the endpoint and bank (EPn_PCKSIZE_BK0/1):

Table 19-3. EPn_PCKSIZE_BK0/1 structure

31	30:16	15	14:0
AUTO_ZLP	MULTI_PACKET_SIZE	-	BYTE_COUNT

– AUTO_ZLP: Auto zero length packet, see ”Multi packet mode for IN endpoints” on page 298.

– MULTI_PACKET_SIZE: see ”Multi packet mode and single packet mode.” on page 295.

– BYTE_COUNT: see ”Multi packet mode and single packet mode.” on page 295.

- The control and status fields for the endpoint and bank (EPn_CTR_STA_BK0/1):

Table 19-4. EPn_CTR_STA_BK0/1 structure

31:19	18	17	16	15:1	0
Status elements				Control elements	
-	UNDERF	OVERF	CRCERR	-	STALLRQ_NEXT

  - UNDERF: Underflow status for isochronous IN transfer. See ”Data flow error” on page 301.
  - OVERF: Overflow status for isochronous OUT transfer. See ”Data flow error” on page 301.
  - CRCERR: CRC error status for isochronous OUT transfer. See ”CRC error” on page 301.
  - STALLRQ_NEXT: Stall request for the next transfer. See ”STALL request” on page 294.

19.6.2.11 STALL request

For each endpoint, the STALL management is performed using:

- The STALL Request (STALLRQ) bit in UECONn is set to initiate a STALL request.
- The STALLed Interrupt (STALLEDI) bit in UESTAn is set when a STALL handshake has been sent.

To answer requests with a STALL handshake, STALLRQ has to be set by writing a one to the STALL Request Set (STALLRQS) bit. All following requests will be discarded (RXOUTI, etc. will not be set) and handshaked with a STALL until the STALLRQ bit is cleared, by receiving a new SETUP packet (for control endpoints) or by writing a one to the STALL Request Clear (STALLRQC) bit.

Each time a STALL handshake is sent, the STALLEDI bit is set by the USBC and the EPnINT interrupt is set.

The user can use the descriptor to manage STALL requests. The USBC controller reads the EPn_CTR_STA_BK0/1.STALLRQ_NEXT bit after successful transactions and if it is one the USBC controller will set UECON.STALLRQ. The STALL_NEXT bit will be cleared upon receiving a SETUP transaction and the USBC controller will then clear the STALLRQ bit.

- *Special considerations for control endpoints*

  If a SETUP packet is received at a control endpoint where a STALL request is active, the Received SETUP Interrupt (RXSTPI) bit in UESTAn is set, and the STALLRQ and STALLEDI bits are cleared. It allows the SETUP to be always ACKed as required by the USB standard.

  This management simplifies the enumeration process management. If a command is not supported or contains an error, the user requests a STALL and can return to the main task, waiting for the next SETUP request.

- *STALL handshake and retry mechanism*

  The retry mechanism has priority over the STALL handshake. A STALL handshake is sent if the STALLRQ bit is set and if there is no retry required.

19.6.2.12 *Multi packet mode and single packet mode.*

Single packet mode is the default mode where one USB packet is managed per bank.

The multi-packet mode allows the user to manage data exceeding the maximum endpoint size (UECFGn.EPSIZE) for an endpoint bank across multiple packets without software intervention. This mode can also be coupled with the ping-pong mode.

- For an OUT endpoint, the EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE field should be configured correctly to enable the multi-packet mode. See ”Multi packet mode for OUT endpoints” on page 300. For single packet mode, the MULTI_PACKET_SIZE should be initialized to 0.
- For an IN endpoint, the EPn_PCKSIZE_BK0/1.BYTE_COUNT field should be configured correctly to enable the multi-packet mode. See”Multi packet mode for IN endpoints” on page 298. For single packet mode, the BYTE_COUNT should be less than EPSIZE.

19.6.2.13 *Management of control endpoints*

- *Overview*

A SETUP request is always ACKed. When a new SETUP packet is received, the RXSTPI is set, but not the Received OUT Data Interrupt (RXOUTI) bit.

The FIFO Control (FIFOCON) bit in UECONn is irrelevant for control endpoints. The user should therefore never use it for these endpoints. When read, this value is always zero.

Control endpoints are managed using:

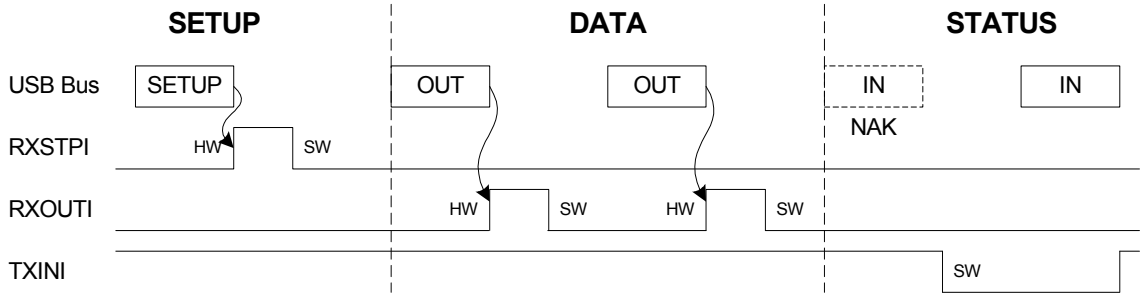
- The RXSTPI bit: is set when a new SETUP packet is received. This has to be cleared by firmware in order to acknowledge the packet and to free the bank.
- The RXOUTI bit: is set when a new OUT packet is received. This has to be cleared by firmware in order to acknowledge the packet and to free the bank.
- The Transmitted IN Data Interrupt (TXINI) bit: is set when the current bank is ready to accept a new IN packet. This has to be cleared by firmware in order to send the packet.

- *Control write*

Figure 19-6 on page 296 shows a control write transaction. During the status stage, the controller will not necessarily send a NAK on the first IN token:

- If the user knows the exact number of descriptor bytes that will be read, the status stage can be predicted, and a zero-length packet can be sent after the next IN token.
- Alternatively the bytes can be read until the NAKed IN Interrupt (NAKINI) is triggered, notifying that all bytes are sent by the host and that the transaction is now in the status stage.

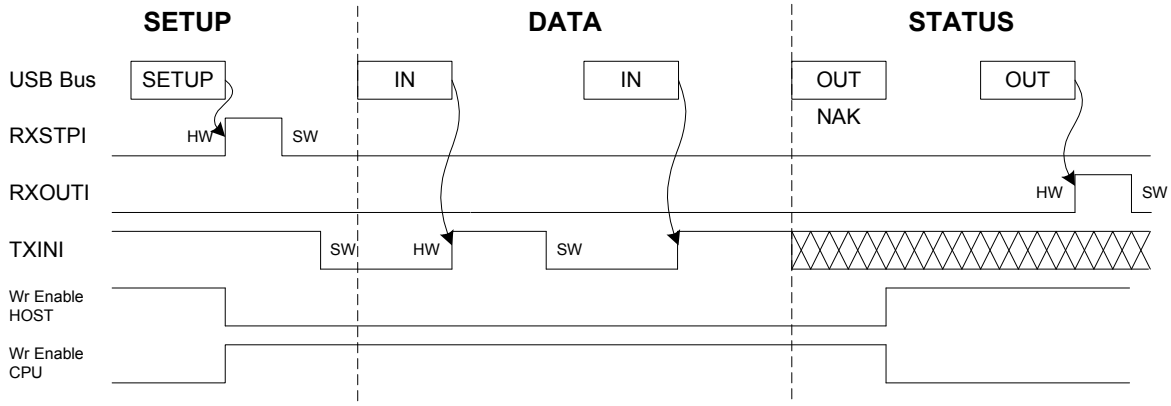
Figure 19-6. Control Write



• *Control read*

Figure 19-7 on page 296 shows a control read transaction. The USBC has to manage the simultaneous write requests from the CPU and USB host.

Figure 19-7. Control Read



A NAK handshake is always generated as the first status stage command. The UESTAn.NAKINI bit is set. It allows the user to know that the host aborts the IN data stage. As a consequence, the user should stop processing the IN data stage and should prepare to receive the OUT status stage by checking the UESTAn.RXOUTI bit.

The OUT retry is always ACKed. This OUT reception sets RXOUTI. Handle this with the following software algorithm:

```
// process the IN data stage
set TXINI
wait for RXOUTI (rising) OR TXINI (falling)
if RXOUTI is high, then process the OUT status stage
if TXINI is low, then return to process the IN data stage
```

Once the OUT status stage has been received, the USBC waits for a SETUP request. The SETUP request has priority over all other requests and will be ACKed.

19.6.2.14 Management of IN endpoints

- *Overview*

IN packets are sent by the USBC device controller upon IN requests from the host.

The endpoint and its descriptor in RAM must be pre configured (see section "RAM management" on page 292 for more details).

When the current bank is clear, the TXINI and FIFO Control (UECONn.FIFOCON) bits will be set simultaneously. This triggers an EPnINT interrupt if the Transmitted IN Data Interrupt Enable (TXINE) bit in UECONn is one.

TXINI shall be cleared by software (by writing a one to the Transmitted IN Data Interrupt Enable Clear bit in the Endpoint n Control Clear register (UECONnCLR.TXINIC)) to acknowledge the interrupt. This has no effect on the endpoint FIFO.

The user writes the IN data to the bank referenced by the EPn descriptor and allows the USBC to send the data by writing a one to the FIFO Control Clear (UECONnCLR.FIFOCONC) bit. This will also cause a switch to the next bank if the IN endpoint is composed of multiple banks. The TXINI and FIFOCON bits will be updated accordingly.

TXINI should always be cleared before clearing FIFOCON to avoid missing an TXINI event.

Figure 19-8. Example of an IN endpoint with one data bank

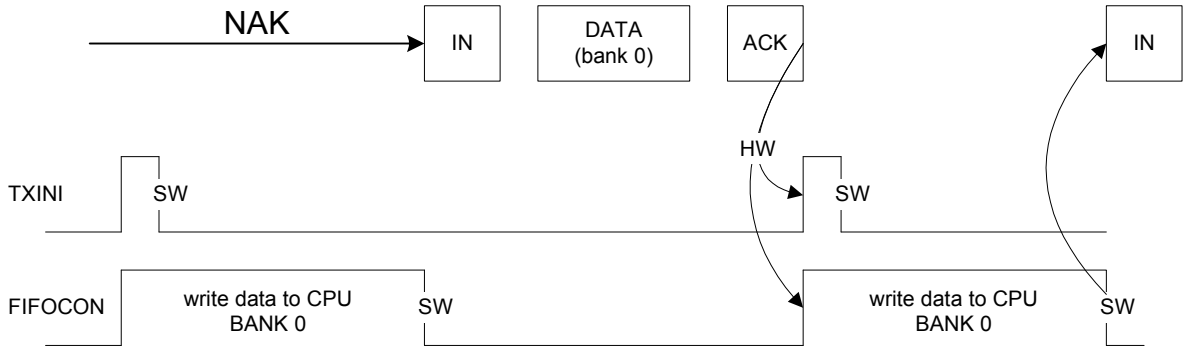
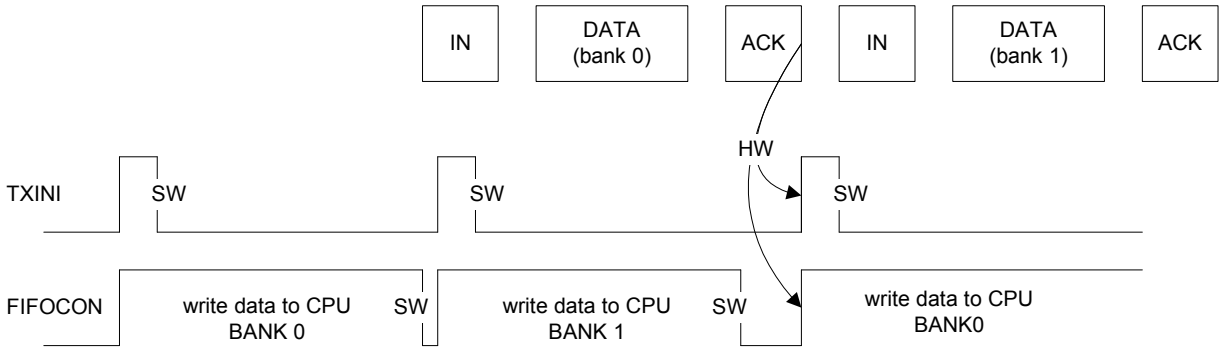


Figure 19-9. Example of an IN endpoint with two data banks



• *Detailed description*

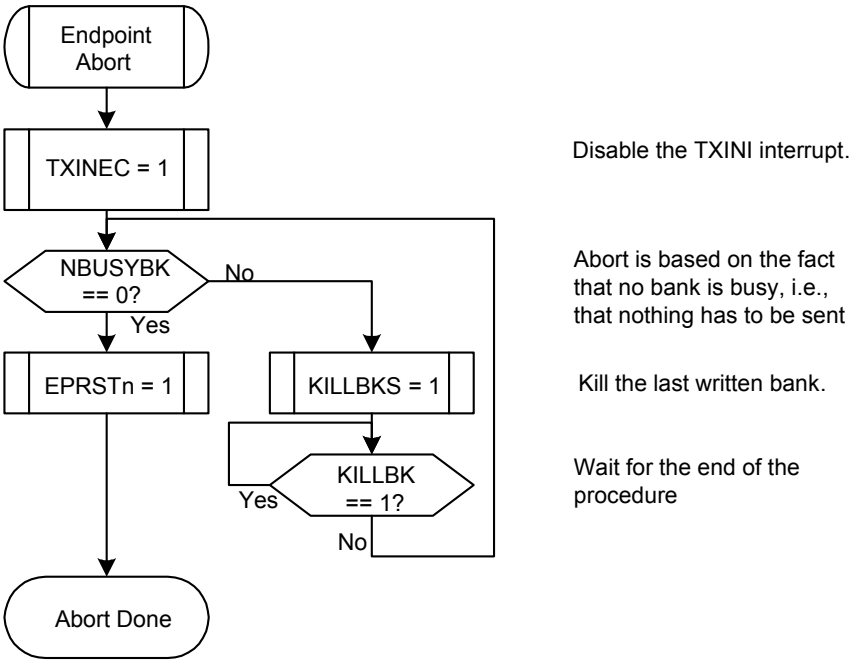
The data is written according to this sequence:

- When the bank is empty, TXINI and FIFOCON are set, which triggers an EPnINT interrupt if TXINE is one.
- The user acknowledges the interrupt by clearing TXINI.
- The user reads the UESTAX.CURRBK field to see which the current bank is.
- The user writes the data to the current bank, located in RAM as described by its descriptor: EPn_ADDR_BK0/1.
- The user should write the size of the IN packet into the USB descriptor: EPn_PCKSIZE_BK0/1.BYTE_COUNT.
- The user allows the controller to send the bank contents and switches to the next bank (if any) by clearing FIFOCON.

If the endpoint uses several banks, the current one can be written while the previous one is being read by the host. When the user clears FIFOCON, the next current bank may already be clear and TXINI is set immediately.

An “Abort” stage can be produced when a zero-length OUT packet is received during an IN stage of a control or isochronous IN transaction. The Kill IN Bank (KILLBK) bit in UECONn is used to kill the last written bank. The best way to manage this abort is to apply the algorithm represented on Figure 19-10 on page 298. See ”Endpoint n Control Register” on page 332 for more details about the KILLBK bit.

Figure 19-10. Abort Algorithm



• *Multi packet mode for IN endpoints*

In multi packet mode, the user can prepare n USB packets in the bank to be sent on a multiple IN transaction. The packet sizes will equal UECFGn.EPSIZE unless the AUTO_ZLP option is

set, or if the total byte count is not an integral multiple of EPSIZE, whereby the last packet should be short.

To enable the multi packet mode, the user should configure the endpoint descriptor (EPn_PCKSIZE_BK0/1.BYTE_COUNT) to the total size of the multi packet, which should be larger than the endpoint size (EPSIZE).

Since the EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE is incremented (by the transmitted packet size) after each successful transaction, it should be set to zero when setting up a new multi packet transfer.

The EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE is cleared by hardware when all the bank contents have been sent. The bank is considered as ready and the TX_IN flag is set when:

- A short packet (smaller than EPSIZE) has been transmitted.
- A packet has been successfully transmitted, the updated MULTI_PACKET_SIZE equals the BYTE_COUNT, and the AUTO_ZLP field is not set.
- An extra zero length packet has been automatically sent for the last transfer of the current bank, if BYTE_COUNT is a multiple of EPSIZE and AUTO_ZLP is set.

19.6.2.15 *Management of OUT endpoints*

- *Overview*

The endpoint and its descriptor in RAM must be pre configured, see section ”RAM management” on page 292 for more details.

When the current bank is full, the RXOUTI and FIFO Control (UECONn.FIFOCON) bits will be set simultaneously. This triggers an EPnINT interrupt if the Received OUT Data Interrupt Enable (RXOUTE) bit in UECONn is one.

RXOUTI shall be cleared by software (by writing a one to the Received OUT Data Interrupt Clear (RXOUTIC) bit) to acknowledge the interrupt. This has no effect on the endpoint FIFO.

The user reads the OUT data from the RAM and clears the FIFOCON bit to free the bank. This will also cause a switch to the next bank if the OUT endpoint is composed of multiple banks.

RXOUTI should always be cleared before clearing FIFOCON to avoid missing an RXOUTI event.

Figure 19-11. Example of an OUT endpoint with one data bank

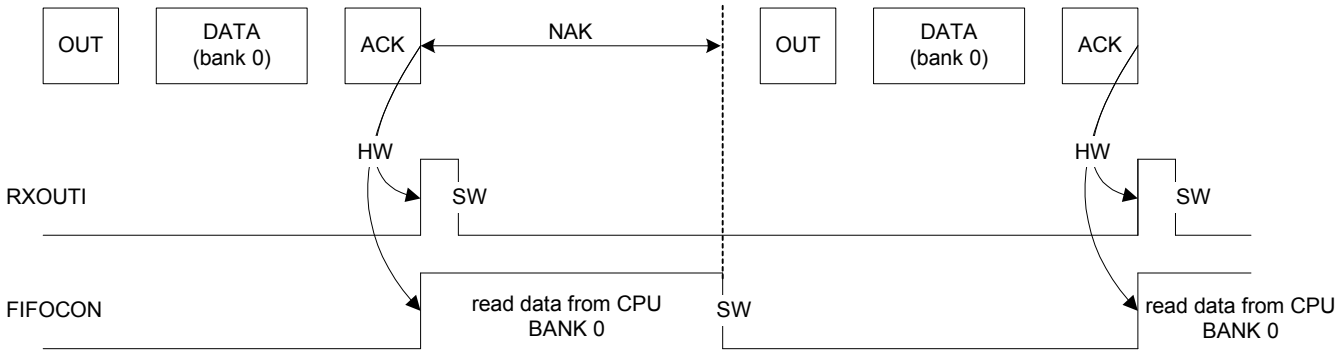
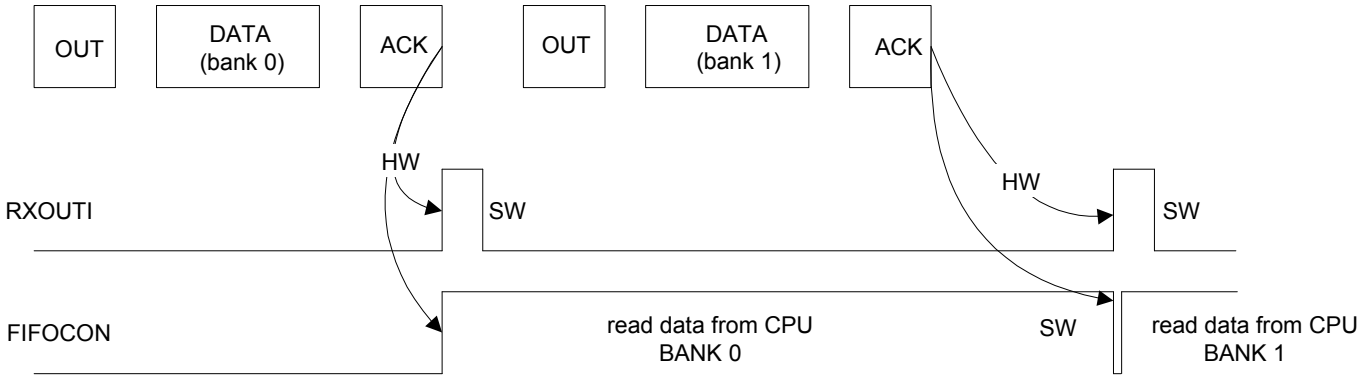


Figure 19-12. Example of an OUT endpoint with two data banks



• *Detailed description*

Before using the OUT endpoint, one should properly initialize its descriptor for each bank. See Figure 19-5 on page 293.

The data is read, according to this sequence:

- When the bank is full, RXOUTI and FIFOCON are set, which triggers an EPnINT interrupt if RXOUTE is one.
- The user acknowledges the interrupt by writing a one to RXOUTIC in order to clear RXOUTI.
- The user reads the UESTAX.CURRBK field to know the current bank number.
- The user reads the byte count of the current bank from the descriptor in RAM (EPn_PCKSIZE_BK0/1.BYTE_COUNT) to know how many bytes to read.
- The user reads the data in the current bank, located in RAM as described by its descriptor: EPn_ADDR_BK0/1.
- The user frees the bank and switches to the next bank (if any) by clearing FIFOCON.

If the endpoint uses several banks, the current one can be read while the next is being written by the host. When the user clears FIFOCON, the following bank may already be ready and RXOUTI will be immediately set.

• *Multi packet mode for OUT endpoints*

In multi packet mode, the user can extend the size of the bank allowing the storage of n USB packets in the bank.

To enable the multi packet mode, the user should configure the endpoint descriptor (EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE) to match the size of the multi packet.This value should be a multiple of the endpoint size (UECFGn.EPSIZE).

Since the EPn_PCKSIZE_BK0/1.BYTE_COUNT is incremented (by the received packet size) after each successful transaction, it should be set to zero when setting up a new multi packet transfer.

As for single packet mode, the number of received data bytes is stored in the BYTE_CNT field.

The bank is considered as “valid” and the RX_OUT flag is set when:

- A packet has been successfully received and the updated BYTE_COUNT equals the MULTI_PACKET_SIZE.
- A short packet (smaller than EPSIZE) has been received.

19.6.2.16 *Data flow error*

This error exists only for isochronous IN/OUT endpoints. It sets the Errorflow Interrupt (ERRORFI) bit in UESTAn, which triggers an EPnINT interrupt if the Errorflow Interrupt Enable (ERRORFE) bit is one. The user can check the EPn_CTR_STA_BK0/1.UNDERF and OVERF bits in the endpoint descriptor to see which current bank has been affected.

- An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBC. The endpoint descriptor EPn_CTR_STA_BK0/1.UNDERF points out the bank from which the IN data should have originated. If a new successful transaction occurs, the UNDERF bit is overwritten to 0 only if the UESTAn.ERRORFI is cleared.
- An overflow can occur during the OUT stage if the host tries to send a packet while the bank is full. Typically this occurs when a CPU is not fast enough. The packet data is not written to the bank and is lost. The endpoint descriptor EPn_CTR_STA_BK0/1.OVERF points out which bank the OUT data was destined to. If the UESTAn.ERRORFI bit is cleared and a new transaction is successful, the OVERF bit will be overwritten to zero.

19.6.2.17 *CRC error*

This error exists only for isochronous OUT endpoints. It sets the CRC Error Interrupt (CRCERRI) bit in UESTAn, which triggers an EPnINT interrupt if the CRC Error Interrupt Enable (CRCERRE) bit is one.

A CRC error can occur during an isochronous OUT stage if the USBC detects a corrupted received packet. The OUT packet is stored in the bank as if no CRC error had occurred (RXOUTI is set).

The user can also check the endpoint descriptor to see which current bank is impacted by the CRC error by reading EPn_CTR_STA_BK0/1.CRCERR.

19.6.2.18 *Interrupts*

There are two kinds of device interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors not related to CPU exceptions.

- *Global interrupts*

The processing device global interrupts are:

- The Suspend (SUSP) interrupt
- The Start of Frame (SOF) interrupt with no frame number CRC error (the Frame Number CRC Error (FNCERR) bit in the Device Frame Number (UDFNUM) register is zero)
- The End of Reset (EORST) interrupt
- The Wakeup (WAKEUP) interrupt
- The End of Resume (EORSM) interrupt
- The Upstream Resume (UPRSM) interrupt
- The Endpoint n (EPnINT) interrupt

The exception device global interrupts are:

- The Start of Frame (SOF) interrupt with a frame number CRC error (FNCERR is one)

- *Endpoint interrupts*

  The processing device endpoint interrupts are:

  - The Transmitted IN Data Interrupt (TXINI)
  - The Received OUT Data Interrupt (RXOUTI)
  - The Received SETUP Interrupt (RXSTPI)
  - The Number of Busy Banks (NBUSYBK) interrupt

  The exception device endpoint interrupts are:

  - The Errorflow Interrupt (ERRORFI)
  - The NAKed OUT Interrupt (NAKOUTI)
  - The NAKed IN Interrupt (NAKINI)
  - The STALLed Interrupt (STALLEDI)
  - The CRC Error Interrupt (CRCERRI)

19.7 User Interface

Table 19-5. USBC Register Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Device General Control Register	UDCON	Read/Write	0x00000100
0x0004	Device Global Interrupt Register	UDINT	Read-Only	0x00000000
0x0008	Device Global Interrupt Clear Register	UDINTCLR	Write-Only	0x00000000
0x000C	Device Global Interrupt Set Register	UDINTSET	Write-Only	0x00000000
0x0010	Device Global Interrupt Enable Register	UDINTE	Read-Only	0x00000000
0x0014	Device Global Interrupt Enable Clear Register	UDINTECLR	Write-Only	0x00000000
0x0018	Device Global Interrupt Enable Set Register	UDINTESET	Write-Only	0x00000000
0x001C	Endpoint Enable/Reset Register	UERST	Read/Write	0x00000000
0x0020	Device Frame Number Register	UDFNUM	Read-Only	0x00000000
0x0100 + n*4	Endpoint n Configuration Register	UECFGn	Read/Write	0x00000000
0x0130 + n*4	Endpoint n Status Register	UESTAn	Read-Only	0x00000100
0x0160 + n*4	Endpoint n Status Clear Register	UESTAnCLR	Write-Only	0x00000000
0x0190 + n*4	Endpoint n Status Set Register	UESTAnSET	Write-Only	0x00000000
0x01C0 + n*4	Endpoint n Control Register	UECONn	Read-Only	0x00000000
0x01F0 + n*4	Endpoint n Control Set Register	UECONnSET	Write-Only	0x00000000
0x0220 + n*4	Endpoint n Control Clear Register	UECONnCLR	Write-Only	0x00000000
0x0800	General Control Register	USBCON	Read/Write	0x00004000
0x0804	General Status Register	USBSTA	Read-Only	0x00000000
0x0808	General Status Clear Register	USBSTACLR	Write-Only	0x00000000
0x080C	General Status Set Register	USBSTASET	Write-Only	0x00000000
0x0818	IP Version Register	UVERS	Read-Only	-(1)
0x081C	IP Features Register	UFEATURES	Read-Only	-(1)
0x0820	IP PB Address Size Register	UADDRSIZE	Read-Only	-(1)
0x0824	IP Name Register 1	UNAME1	Read-Only	-(1)
0x0828	IP Name Register 2	UNAME2	Read-Only	-(1)
0x082C	USB Finite State Machine Status Register	USBFSM	Read-Only	0x00000009
0x0830	USB Descriptor address	UDESC	Read/Write	0x00000000

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

19.7.1 USB General Registers

19.7.1.1 General Control Register

Name: USBCON

Access Type: Read/Write

Offset: 0x0800

Reset Value: 0x00004000

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

- **USBE: USBC Enable**
  Writing a zero to this bit will disable the USBC, USB transceiver, and USB clock inputs. This will over-ride FRZCLK settings but not affect the value. Unless explicitly stated, all registers will become reset and read-only.
  Writing a one to this bit will enable the USBC.
  0: The USBC is disabled.
  1: The USBC is enabled.
  This bit can be written to even if FRZCLK is one.
- **FRZCLK: Freeze USB Clock**
  Writing a zero to this bit will enable USB clock inputs.
  Writing a one to this bit will disable USB clock inputs. The resume detection will remain active. Unless explicitly stated, all registers will become read-only.
  0: The clock inputs are enabled.
  1: The clock inputs are disabled.
  This bit can be written to even if USBE is zero.
- **VBUSTE: VBUS Transition Interrupt Enable**
  0: The VBUS Transition Interrupt (VBUSTI) is disabled.
  1: The VBUS Transition Interrupt (VBUSTI) is enabled.

19.7.1.2 General Status Register

Register Name: USBSTA

Access Type: Read-Only

Offset: 0x0804

Reset Value: 0x00000000

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

- **CLKUSABLE: Generic Clock Usable**
  This bit is cleared when the USB generic clock is not usable.
  This bit is set when the USB generic clock (that should be 48 Mhz) is usable.
- **SPEED: Speed Status**
  This field is set according to the controller speed mode.

SPEED	Speed Status
00	full-speed mode
01	Reserved
10	low-speed mode
11	Reserved

- **VBUS: VBUS Level**
  This bit is cleared when the VBUS line level is low, even if USBE is zero.
  This bit is set when the VBUS line level is high, even if USBE is zero.
  This bit can be used in device mode to monitor the USB bus connection state of the application.
- **VBUSTI: VBUS Transition Interrupt**
  This bit is cleared when the USBSTACLR.VBUSTIC bit is written to one.
  This bit is set when a transition (high to low, low to high) has been detected on the USB_VBUS pad. This triggers a USB interrupt if VBUSTE is one.
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.

19.7.1.3 General Status Clear Register

Register Name: USBSTACLR

Access Type: Write-Only

Offset: 0x0808

Reset Value: 0x00000000

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

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

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

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in USBSTA.
These bits always read as zero.

19.7.1.4 General Status Set Register

Register Name: USBSTASET

Access Type: Write-Only

Offset: 0x080C

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in USBSTA.
These bits always read as zero.

19.7.1.5 Version Register

Register Name: UVERS

Access Type: Read-Only

Offset: 0x0818

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.

19.7.1.6 Features Register

Register Name: UFEATURES

Access Type: Read-Only

Offset: 0x081C

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
-	-	-	-	EPTNBRMAX			

- **EPTNBRMAX: Maximal Number of pipes/endpoints**
  This field indicates the number of hardware-implemented pipes/endpoints:

19.7.1.7 Address Size Register

Register Name: UADDRSIZE

Access Type: Read-Only

Offset: 0x0820

Reset Value: -

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

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

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

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

- **UADDRSIZE: IP PB Address Size**
  This field indicates the size of the PB address space reserved for the USBC IP interface.

19.7.1.8 IP Name Register 1

Register Name: UNAME1

Access Type: Read-Only

Offset: 0x0824

Reset Value: -

31	30	29	28	27	26	25	24
UNAME1[31:24]							

23	22	21	20	19	18	17	16
UNAME1[23:16]							

15	14	13	12	11	10	9	8
UNAME1[15:8]							

7	6	5	4	3	2	1	0
UNAME1[7:0]							

- **UNAME1: IP Name Part One**
  This field indicates the first part of the ASCII-encoded name of the USBC IP.

19.7.1.9 IP Name Register 2

Register Name: UNAME2

Access Type: Read-Only

Offset: 0x0828

Reset Value:

31	30	29	28	27	26	25	24
UNAME2[31:24]							

23	22	21	20	19	18	17	16
UNAME2[23:16]							

15	14	13	12	11	10	9	8
UNAME2[15:8]							

7	6	5	4	3	2	1	0
UNAME2[7:0]							

- **UNAME2: IP Name Part Two**
  This field indicates the second part of the ASCII-encoded name of the USBC IP.

19.7.1.10 Finite State Machine Status Register

Register Name: USBFSM

Access Type: Read-Only

Offset: 0x082C

Reset Value: 0x00000009

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

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

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

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

- **DRDSTATE: Dual Role Device State**
  This field indicates the state of the USBC.
  For Device mode it should always read 9.

19.7.1.11 USB Descriptor Address

Register Name: UDESC

Access Type: Read-Write

Offset: 0x0830

Reset Value: -

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

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

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

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

- **UDESCA: USB Descriptor Address**
  This field contains the address of the USB descriptor. The three least significant bits are always zero.

19.7.2 USB Device Registers

19.7.2.1 Device General Control Register

Register Name: UDCON

Access Type: Read/Write

Offset: 0x0000

Reset Value: 0x00000100

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

- **GNAK: Global NAK**
  0: Normal mode.
  1: A NAK handshake is answered for each USB transaction regardless of the current endpoint memory bank status.
- **LS: low-speed mode force**
  0: The full-speed mode is active.
  1: The low-speed mode is active.
  This bit can be written to even if USBE is zero or FRZCLK is one. Disabling the USBC (by writing a zero to the USBE bit) does not reset this bit.
- **RMWKUP: Remote wakeup**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will send an upstream resume to the host for a remote wakeup.
  This bit is cleared when the USBC receives a USB reset or once the upstream resume has been sent.
- **DETACH: Detach**
  Writing a zero to this bit will reconnect the device.
  Writing a one to this bit will physically detach the device (disconnect internal pull-up resistor from DP and DM).
- **ADDEN: Address Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will activate the UADD field (USB address).
  This bit is cleared when a USB reset is received.
- **UADD: USB Address**
  This field contains the device address.
  This field is cleared when a USB reset is received.

19.7.2.2 Device Global Interrupt Register

Register Name: UDINT

Access Type: Read-Only

Offset: 0x0004

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	EP8INT(1)	EP7INT(1)	EP6INT(1)	EP5INT(1)	EP4INT(1)

15	14	13	12	11	10	9	8
EP3INT(1)	EP2INT(1)	EP1INT(1)	EP0INT	-	-	-	-

7	6	5	4	3	2	1	0
-	UPRSM	EORSM	WAKEUP	EORST	SOF	-	SUSP

Note: 1. EPnINT bits are within the range from EP0INT to EP6INT.

- **EPnINT: Endpoint n Interrupt**
  This bit is cleared when the interrupt source is serviced.
  This bit is set when an interrupt is triggered by the endpoint n (UESTAn, UECONn). This triggers a USB interrupt if EPnINTE is one.
- **UPRSM: Upstream Resume Interrupt**
  This bit is cleared when the UDINTCLR.UPRSMC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before).
  This bit is set when the USBC sends a resume signal called “Upstream Resume”. This triggers a USB interrupt if UPRSME is one.
- **EORSM: End of Resume Interrupt**
  This bit is cleared when the UDINTCLR.EORSMC bit is written to one to acknowledge the interrupt.
  This bit is set when the USBC detects a valid “End of Resume” signal initiated by the host. This triggers a USB interrupt if EORSME is one.
- **WAKEUP: Wakeup Interrupt**
  This bit is cleared when the UDINTCLR.WAKEUPC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before) or when the Suspend (SUSP) interrupt bit is set.
  This bit is set when the USBC is reactivated by a filtered non-idle signal from the lines (not by an upstream resume). This triggers an interrupt if WAKEUPE is one.
  This interrupt is generated even if the clock is frozen by the FRZCLK bit.
- **EORST: End of Reset Interrupt**
  This bit is cleared when the UDINTCLR.EORSTC bit is written to one to acknowledge the interrupt.
  This bit is set when a USB “End of Reset” has been detected. This triggers a USB interrupt if EORSTE is one.
- **SOF: Start of Frame Interrupt**
  This bit is cleared when the UDINTCLR.SOFC bit is written to one to acknowledge the interrupt.
  This bit is set when a USB “Start of Frame” PID (SOF) has been detected (every 1 ms). This triggers a USB interrupt if SOFE is one. The FNUM field is updated.

- **SUSP: Suspend Interrupt**

  This bit is cleared when the UDINTCLR.SUSPC bit is written to one to acknowledge the interrupt or when the Wakeup (WAKEUP) interrupt bit is set.

  This bit is set when a USB “Suspend” idle bus state has been detected for 3 frame periods (J state for 3 ms). This triggers a USB interrupt if SUSPE is one.

19.7.2.3 Device Global Interrupt Clear Register

Register Name: UDINTCLR

Access Type: Write-Only

Offset: 0x0008

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	UPRSMC	EORSMC	WAKEUPC	EORSTC	SOFC	-	SUSPC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UDINT.
These bits always read as zero.

19.7.2.4 Device Global Interrupt Set Register

Register Name: UDINTSET

Access Type: Write-Only

Offset: 0x000C

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UDINT, which may be useful for test or debug purposes.
These bits always read as zero.

19.7.2.5 Device Global Interrupt Enable Register

Register Name: UDINTE

Access Type: Read-Only

Offset: 0x0010

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	EP8INTE[1]	EP7INTE[1]	EP6INTE[1]	EP5INTE[1]	EP4INTE[1]
15	14	13	12	11	10	9	8
EP3INTE[1]	EP2INTE[1]	EP1INTE[1]	EP0INTE	-	-	-	-
7	6	5	4	3	2	1	0
-	UPRSME	EORSME	WAKEUPE	EORSTE	SOFE	-	SUSPE

Note: 1. EPnINTE bits are within the range from EP0INTE to EP6INTE.

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

A bit in this register is cleared when the corresponding bit in UDINTECLR is written to one.

A bit in this register is set when the corresponding bit in UDINTESET is written to one.

19.7.2.6 Device Global Interrupt Enable Clear Register

Register Name: UDINTECLR

Access Type: Write-Only

Offset: 0x0014

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	EP8INTEC(1)	EP7INTEC(1)	EP6INTEC(1)	EP5INTEC(1)	EP4INTEC(1)

15	14	13	12	11	10	9	8
EP3INTEC(1)	EP2INTEC(1)	EP1INTEC(1)	EP0INTEC	-	-	-	-

7	6	5	4	3	2	1	0
-	UPRSMEC	EORSMEC	WAKEUPEC	EORSTEC	SOFEC	-	SUSPEC

Note: 1. EPnINTEC bits are within the range from EP0INTEC to EP6INTEC.

Writing a zero to a bit in this register has no effect.

Writing a one to a bit in this register will clear the corresponding bit in UDINTE.

These bits always read as zero.

19.7.2.7 Device Global Interrupt Enable Set Register

Register Name: UDINTESET

Access Type: Write-Only

Offset: 0x0018

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	EP8INTES(1)	EP7INTES(1)	EP6INTES(1)	EP5INTES(1)	EP4INTES(1)

15	14	13	12	11	10	9	8
EP3INTES(1)	EP2INTES(1)	EP1INTES(1)	EP0INTES	-	-	-	-

7	6	5	4	3	2	1	0
-	UPRSMES	EORSMES	WAKEUPES	EORSTES	SOFES	-	SUSPES

Note: 1. EPnINTES bits are within the range from EP0INTES to EP6INTES.
Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UDINTE.
These bits always read as zero.

19.7.2.8 Endpoint Enable/Reset Register

Register Name: UERST

Access Type: Read/Write

Offset: 0x001C

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	EPEN8(1)

7	6	5	4	3	2	1	0
EPEN7(1)	EPEN6(1)	EPEN5(1)	EPEN4(1)	EPEN3(1)	EPEN2(1)	EPEN1(1)	EPEN0

- **EPENn: Endpoint n Enable**

Note: 1. EPENn bits are within the range from EPEN0 to EPEN6.

Writing a zero to this bit will disable the endpoint n (USB requests will be ignored), and resets the endpoints registers (UECFGn, UESTAn, UECONn), but not the endpoint configuration (EPBK, EPSIZE, EPDIR, EPTYPE).

Writing a one to this bit will enable the endpoint n.

0: The endpoint n is disabled.

1: The endpoint n is enabled.

19.7.2.9 Device Frame Number Register

Register Name: UDFNUM

Access Type: Read-Only

Offset: 0x0020

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
FNCERR	-	FNUM[10:5]					

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

- **FNCERR: Frame Number CRC Error**
  This bit is cleared upon receiving a USB reset.
  This bit is set when a corrupted frame number is received. This bit and the SOF interrupt bit are updated at the same time.
- **FNUM: Frame Number**
  This field is cleared upon receiving a USB reset.
  This field contains the 11-bit frame number information, as provided from the last SOF packet.
  FNUM is updated even if a corrupted SOF is received.

19.7.2.10 Endpoint n Configuration Register

Register Name: UECFGn, n in [0..6]

Access Type: Read/Write

Offset: 0x0100 + (n * 0x04)

Reset Value: 0x00000000

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

- **EPTYPE: Endpoint Type**
  This field selects the endpoint type:

EPTYPE		Endpoint Type
0	0	Control
0	1	Isochronous
1	0	Bulk
1	1	Interrupt

  This field is cleared upon receiving a USB reset.
- **EPDIR: Endpoint Direction**
  0: The endpoint direction is OUT.
  1: The endpoint direction is IN (nor for control endpoints).
  This bit is cleared upon receiving a USB reset.
- **EPSIZE: Endpoint Size**
  This field determines the size of each endpoint bank:

EPSIZE			Endpoint Size
0	0	0	8 bytes
0	0	1	16 bytes
0	1	0	32 bytes
0	1	1	64 bytes
1	0	0	128 bytes

EPSIZE			Endpoint Size
1	0	1	256 bytes
1	1	0	512 bytes
1	1	1	1024 bytes

This field is cleared upon receiving a USB reset (except for the endpoint 0).

- **EPBK: Endpoint Banks**

  This bit selects the number of banks for the endpoint:

  0: single-bank endpoint

  1: double-bank endpoint

  For control endpoints, a single-bank endpoint shall be selected.

  This field is cleared upon receiving a USB reset (except for the endpoint 0).

19.7.2.11 Endpoint n Status Register

Register Name: UESTAn, n in [0..6]

Access Type: Read-Only 0x0100

Offset: 0x0130 + (n * 0x04)

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		RAMACERI	-	DTSEQ	

7	6	5	4	3	2	1	0
-	STALLEDI/ CRCERRI	-	NAKINI	NAKOUTI	RXSTPI/ ERRORFI	RXOUTI	TXINI

- **CTRLDIR: Control Direction**
  Writing a zero or a one to this bit has no effect.
  This bit is cleared after a SETUP packet to indicate that the following packet is an OUT packet.
  This bit is set after a SETUP packet to indicate that the following packet is an IN packet.
- **CURRBK: Current Bank**
  This bit is set for non-control endpoints, indicating the current bank:

CURRBK		Current Bank
0	0	Bank0
0	1	Bank1
1	0	Reserved
1	1	Reserved

This field may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

- **NBUSYBK: Number of Busy Banks**
  This field is set to indicate the number of busy banks:

NBUSYBK		Number of Busy Banks
0	0	0 (all banks free)
0	1	1
1	0	2
1	1	Reserved

For IN endpoints, this indicates the number of banks filled by the user and ready for IN transfers. When all banks are free an EPnINT interrupt will be triggered if NBUSYBKE is one.

For OUT endpoints, this indicates the number of banks filled by OUT transactions from the host. When all banks are busy an EPnINT interrupt will be triggered if NBUSYBKE is one.

- **RAMACERI: Ram Access Error Interrupt**

  This bit is cleared when the RAMACERIC bit is written to one, acknowledging the interrupt.

  This bit is set when a RAM access underflow error occurs during an IN data stage.

- **DTSEQ: Data Toggle Sequence**

  This field is set to indicate the PID of the current bank:

DTSEQ		Data Toggle Sequence
0	0	Data0
0	1	Data1
1	X	Reserved

For IN transfers, this indicates the data toggle sequence that will be used for the next packet to be sent.

For OUT transfers, this value indicates the data toggle sequence of the data received in the current bank.

- **STALLEDI: STALLed Interrupt**

  This bit is cleared when the STALLEDIC bit is written to one, acknowledging the interrupt.

  This bit is set when a STALL handshake has been sent and triggers an EPnINT interrupt if STALLEDE is one.

- **CRCERRI: CRC Error Interrupt**

  This bit is cleared when the CRCERRIC bit is written to one, acknowledging the interrupt.

  This bit is set when a CRC error has been detected in an isochronous OUT endpoint bank, and triggers an EPnINT interrupt if CRCERRE is one.

- **NAKINI: NAKed IN Interrupt**

  This bit is cleared when the NAKINIC bit is written to one, acknowledging the interrupt.

  This bit is set when a NAK handshake has been sent in response to an IN request from the host, and triggers an EPnINT interrupt if NAKINE is one.

- **NAKOUTI: NAKed OUT Interrupt**

  This bit is cleared when the NAKOUTIC bit is written to one, acknowledging the interrupt.

  This bit is set when a NAK handshake has been sent in response to an OUT request from the host, and triggers an EPnINT interrupt if NAKOUTE is one.

- **ERRORFI: Isochronous Error flow Interrupt**

  This bit is cleared when the ERRORFIC bit is written to one, acknowledging the interrupt.

  This bit is set, for isochronous IN/OUT endpoints, when an errorflow (underflow or overflow) error occurs, and triggers an EPnINT interrupt if ERRORFE is one.

  An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBC.

  An overflow can also occur during OUT stage if the host sends a packet while the bank is already full, resulting in the packet being lost. This is typically due to a CPU not being fast enough.

  This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints and it means RXSTPI for control endpoints.

- **RXSTPI: Received SETUP Interrupt**

  This bit is cleared when the RXSTPIC bit is written to one, acknowledging the interrupt and freeing the bank.

  This bit is set, for control endpoints, to signal that the current bank contains a new valid SETUP packet, and triggers an EPnINT interrupt if RXSTPE is one.

  This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints and it means UNDERFI for isochronous IN/OUT endpoints.

- **RXOUTI: Received OUT Data Interrupt**

  This bit is cleared when the RXOUTIC bit is written to one, acknowledging the interrupt. For control endpoints, it releases the bank. For other endpoint types, the user should clear the FIFOCON bit to free the bank. RXOUTI shall always be cleared before clearing FIFOCON to avoid missing an interrupt.

This bit is set, for control endpoints, when the current bank contains a bulk OUT packet (data or status stage). This triggers an EPnINT interrupt if RXOUTE is one.

This bit is set for isochronous, bulk and, interrupt OUT endpoints, at the same time as FIFOCON when the current bank is full. This triggers an EPnINT interrupt if RXOUTE is one.

This bit is inactive (cleared) for isochronous, bulk and interrupt IN endpoints.

- **TXINI: Transmitted IN Data Interrupt**

  This bit is cleared when the TXINIC bit is written to one, acknowledging the interrupt. For control endpoints, this will send the packet. For other endpoint types, the user should clear the FIFOCON to allow the USBC to send the data. TXINI shall always be cleared before clearing FIFOCON to avoid missing an interrupt.

  This bit is set for control endpoints, when the current bank is ready to accept a new IN packet. This triggers an EPnINT interrupt if TXINE is one.

  This bit is set for isochronous, bulk and interrupt IN endpoints, at the same time as FIFOCON when the current bank is free. This triggers an EPnINT interrupt if TXINE is one.

  This bit is inactive (cleared) for isochronous, bulk and interrupt OUT endpoints.

19.7.2.12 Endpoint n Status Clear Register

Register Name: UESTAnCLR, n in [0..6]

Access Type: Write-Only

Offset: 0x0160 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	RAMACERIC	-	-	-
7	6	5	4	3	2	1	0
-	STALLEDIC/ CRCERRIC	-	NAKINIC	NAKOUTIC	RXSTPIC/ ERRORFIC	RXOUTIC	TXINIC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UESTA.
These bits always read as zero.

19.7.2.13 Endpoint n Status Set Register

Register Name: UESTAnSET, n in [0..6]

Access Type: Write-Only

Offset: 0x0190 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	NBUSYBKS	RAMACERIS	-		-
7	6	5	4	3	2	1	0
-	STALLEDIS/ CRCERRIS	-	NAKINIS	NAKOUTIS	RXSTPIS/ ERRORFIS	RXOUTIS	TXINIS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UESTA.
These bits always read as zero.

19.7.2.14 Endpoint n Control Register

Register Name: UECONn, n in [0..6]

Access Type: Read-Only

Offset: 0x01C0 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	BUSY1E	BUSY0E
23	22	21	20	19	18	17	16
-	-	-	-	STALLRQ	RSTDT	-	-
15	14	13	12	11	10	9	8
-	FIFOCON	KILLBK	NBUSYBKE	RAMACERE	-	-	
7	6	5	4	3	2	1	0
-	STALLEDE/ CRCERRE	-	NAKINE	NAKOUTE	RXSTPE/ ERRORFE	RXOUTE	TXINE

- **BUSY0E: Busy Bank0 Enable**
  This bit is cleared when the BUSY0C bit is written to one.
  This bit is set when the BUSY0ES bit is written to one. This will set the bank 0 as “busy”. All transactions, except SETUP, destined to this bank will be rejected (i.e: NAK token will be answered).
- **BUSY1E: Busy Bank1 Enable**
  This bit is cleared when the BUSY1C bit is written to one.
  This bit is set when the BUSY1ES bit is written to one. This will set the bank 1 as “busy”. All transactions, except SETUP, destined to this bank will be rejected (i.e: NAK token will be answered).
- **STALLRQ: STALL Request**
  This bit is cleared when a new SETUP packet is received or when the STALLRQC bit is written to zero.
  This bit is set when the STALLRQS bit is written to one, requesting a STALL handshake to be sent to the host.
- **RSTDT: Reset Data Toggle**
  The data toggle sequence is cleared when the RSTDTS bit is written to one (i.e., Data0 data toggle sequence will be selected for the next sent (IN endpoints) or received (OUT endpoints) packet.
  This bit is always read as zero.
- **FIFOCON: FIFO Control**
  For control endpoints:
  The FIFOCON and RWALL bits are irrelevant. The software shall therefore never use them for these endpoints. When read, their value is always 0.
  For IN endpoints:
  This bit is cleared when the FIFOCONC bit is written to one, sending the FIFO data and switching to the next bank.
  This bit is set simultaneously to TXINI, when the current bank is free.
  For OUT endpoints:
  This bit is cleared when the FIFOCONC bit is written to one, freeing the current bank and switching to the next.
  This bit is set simultaneously to RXINI, when the current bank is full.

- **KILLBK: Kill IN Bank**
  This bit is cleared by hardware after the completion of the “kill packet procedure”.
  This bit is set when the KILLBKS bit is written to one, killing the last written bank.
  The user shall wait for this bit to be cleared before trying to process another IN packet.
  Caution: The bank is cleared when the “kill packet” procedure is completed by the USBC core:
  If the bank is really killed, the NBUSYBK field is decremented.
  If the bank sent instead of killed (IN transfer), the NBUSYBK field is decremented and the TXINI flag is set. This specific case can occur if an IN token comes while the user tries to kill the bank.
  Note: If two banks are ready to be sent, the above specific case will not occur, since the first bank is sent (IN transfer) while the last bank is killed.
- **NBUSYBKE: Number of Busy Banks Interrupt Enable**
  This bit is cleared when the NBUSYBKEC bit is written to zero, disabling the Number of Busy Banks interrupt (NBUSYBK).
  This bit is set when the NBUSYBKES bit is written to one, enabling the Number of Busy Banks interrupt (NBUSYBK).
- **RAMACERE: RAMACER Interrupt Enable**
  This bit is cleared when the RAMACEREC bit is written to one, disabling the RAMACER interrupt (RAMACERI).
  This bit is set when the RAMACERES bit is written to one, enabling the RAMACER interrupt (RAMACERI).
- **STALLEDE: STALLed Interrupt Enable**
  This bit is cleared when the STALLEDEC bit is written to one, disabling the STALLed interrupt (STALLEDI).
  This bit is set when the STALLEDES bit is written to one, enabling the STALLed interrupt (STALLEDI).
- **CRCERRE: CRC Error Interrupt Enable**
  This bit is cleared when the CRCERREC bit is written to one, disabling the CRC Error interrupt (CRCERRI).
  This bit is set when the CRCERRES bit is written to one, enabling the CRC Error interrupt (CRCERRI).
- **NAKINE: NAKed IN Interrupt Enable**
  This bit is cleared when the NAKINEC bit is written to one, disabling the NAKed IN interrupt (NAKINI).
  This bit is set when the NAKINES bit is written to one, enabling the NAKed IN interrupt (NAKINI).
- **NAKOUTE: NAKed OUT Interrupt Enable**
  This bit is cleared when the NAKOUTEC bit is written to one, disabling the NAKed OUT interrupt (NAKOUTI).
  This bit is set when the NAKOUTES bit is written to one, enabling the NAKed OUT interrupt (NAKOUTI).
- **RXSTPE: Received SETUP Interrupt Enable**
  This bit is cleared when the RXSTPEC bit is written to one, disabling the Received SETUP interrupt (RXSTPI).
  This bit is set when the RXSTPES bit is written to one, enabling the Received SETUP interrupt (RXSTPI).
- **ERRORFE: Errorflow Interrupt Enable**
  This bit is cleared when the ERRORFEC bit is written to one, disabling the Underflow interrupt (ERRORFI).
  This bit is set when the ERRORFES bit is written to one, enabling the Underflow interrupt (ERRORFI).
- **RXOUTE: Received OUT Data Interrupt Enable**
  This bit is cleared when the RXOUTEC bit is written to one, disabling the Received OUT Data interrupt (RXOUT).
  This bit is set when the RXOUTES bit is written to one, enabling the Received OUT Data interrupt (RXOUT).
- **TXINE: Transmitted IN Data Interrupt Enable**
  This bit is cleared when the TXINEC bit is written to one, disabling the Transmitted IN Data interrupt (TXINI).
  This bit is set when the TXINES bit is written to one, enabling the Transmitted IN Data interrupt (TXINI).

19.7.2.15 Endpoint n Control Clear Register

Register Name: UECONnCLR, n in [0..6]

Access Type: Write-Only

Offset: 0x0220 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	BUSY1EC	BUSY0EC
23	22	21	20	19	18	17	16
-	-	-	-	STALLRQC	-	-	-
15	14	13	12	11	10	9	8
-	FIFOCONC	-	NBUSYBKEC	RAMACEREC	-	-	-
7	6	5	4	3	2	1	0
-	STALLEDEC /CRCERREC	-	NAKINEC	NAKOUTEC	RXSTPEC/ ERRORFEC	RXOUTEC	TXINEC

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in UECONn.
These bits always read as zero.

19.7.2.16 Endpoint n Control Set Register

Register Name: UECONnSET, n in [0..6]

Access Type: Write-Only

Offset: 0x01F0 + (n * 0x04)

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	BUSY1ES	BUSY0ES
23	22	21	20	19	18	17	16
-	-	-	-	STALLRQS	RSTDTS	-	-
15	14	13	12	11	10	9	8
-	-	KILLBKS	NBUSYBKES	RAMACERES	-	-	-
7	6	5	4	3	2	1	0
-	STALLEDES/ CRCERRES	-	NAKINES	NAKOUTES	RXSTPES/ ERRORFES	RXOUTES	TXINES

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in UECONn.
These bits always read as zero.

•
•

19.8 Module Configuration

Table 19-6. USBC Clocks

Clock Name	Description
CLK_USBC	Clock for the USBC bus interface
GCLK_USBC	48Mhz USB clock. This clock frequency must be configured to 48MHz. The generic clock used for the USBC is GCLK3

Table 19-7. Register Reset Values

Register	Reset Value
UVERS	0x00000200
UFEATURES	0x00000007
UADDRSIZE	0x00001000
UNAME1	0x48555342
UNAME2	0x00000000

20. Universal Synchronous Asynchronous Receiver Transmitter (USART)

Rev: 4.4.0.6

20.1 Features

- **Configurable baud rate generator**
- **5- to 9-bit full-duplex, synchronous and asynchronous, serial communication**
  - **1, 1.5, or 2 stop bits in asynchronous mode, and 1 or 2 in synchronous mode**
  - **Parity generation and error detection**
  - **Framing- and overrun error detection**
  - **MSB- or LSB-first**
  - **Optional break generation and detection**
  - **Receiver frequency over-sampling by 8 or 16 times**
  - **Optional RTS-CTS hardware handshaking**
  - **Receiver Time-out and transmitter Timeguard**
  - **Optional Multidrop mode with address generation and detection**
- **SPI Mode**
  - **Master or slave**
  - **Configurable serial clock phase and polarity**
  - **CLK SPI serial clock frequency up to a quarter of the CLK_USART internal clock frequency**
- **Test Modes**
  - **Automatic echo, remote- and local loopback**
- **Supports two Peripheral DMA Controller channels**
  - **Buffer transfers without processor intervention**

20.2 Overview

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides a full duplex, universal, synchronous/asynchronous serial link. Data frame format is widely configurable, including basic length, parity, and stop bit settings, maximizing standards support. The receiver implements parity-, framing-, and overrun error detection, and can handle un-fixed frame lengths with the time-out feature. The USART supports several operating modes, providing an interface to and SPI buses and infrared transceivers. Communication with slow and remote devices is eased by the timeguard. Duplex multidrop communication is supported by address and data differentiation through the parity bit. The hardware handshaking feature enables an out-of-band flow control, automatically managing RTS and CTS pins. The Peripheral DMA Controller connection enables memory transactions, and the USART supports chained buffer management without processor intervention. Automatic echo, remote-, and local loopback -test modes are also supported.

20.3 Block Diagram

Figure 20-1. USART Block Diagram

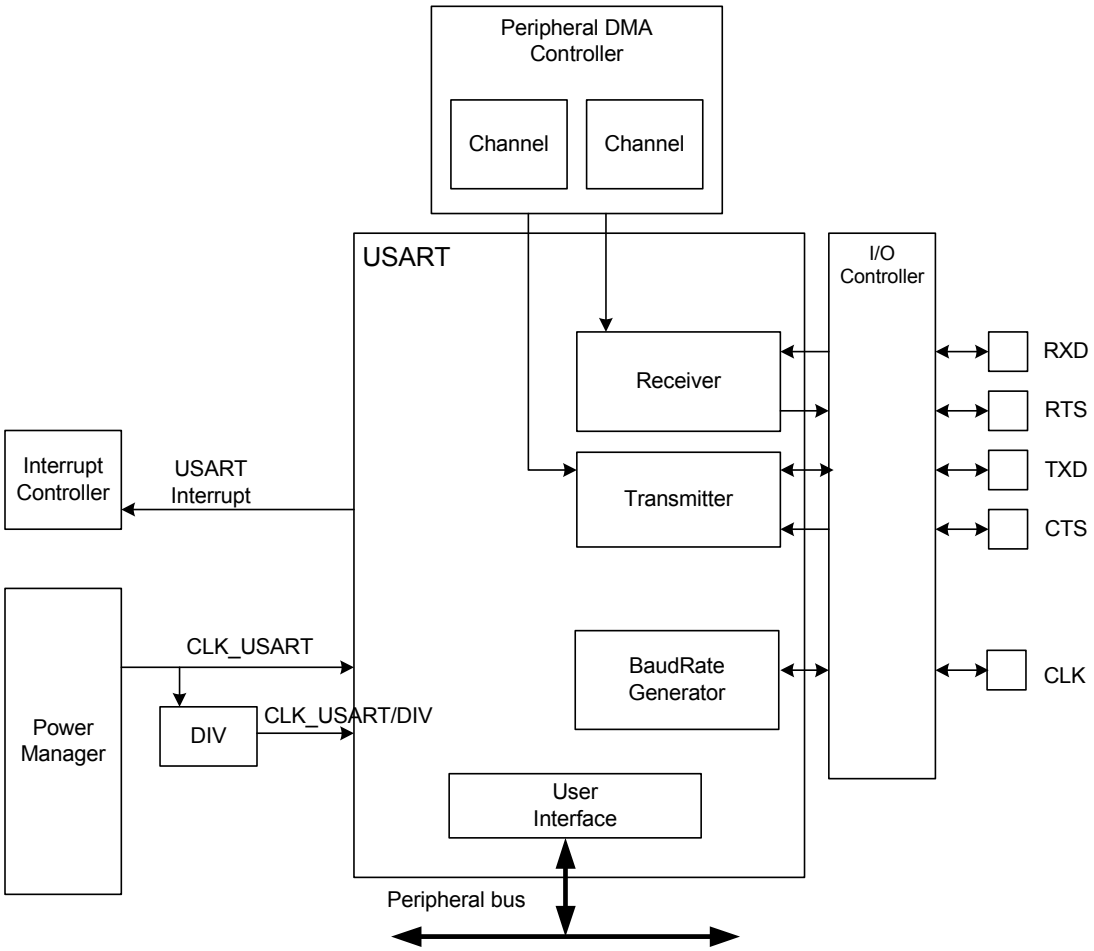


Table 20-1. SPI Operating Mode

PIN	USART	SPI Slave	SPI Master
RXD	RXD	MOSI	MISO
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 USART pins may be multiplexed with the I/O Controller lines. The user must first configure the I/O Controller to assign these pins to their peripheral functions. Unused I/O lines may be used for other purposes.

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is required. 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

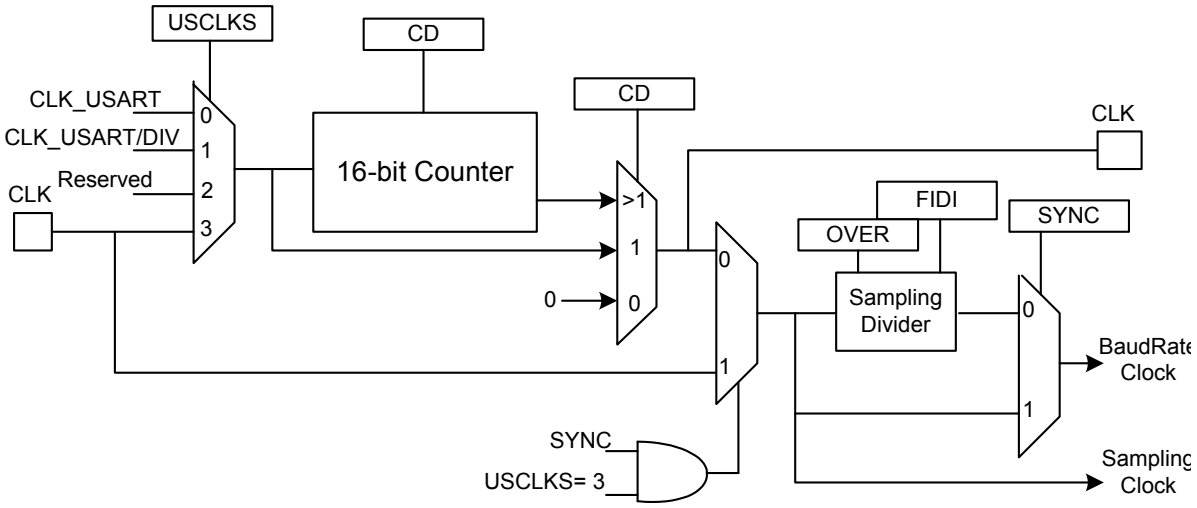
20.6.1 Baud Rate Generator

The baud rate generator provides the bit period clock named the Baud Rate Clock to both receiver and transmitter. It is based on a 16-bit divider, which is specified in the Clock Divider field in the Baud Rate Generator Register (BRGR.CD). A non-zero value enables the generator, and if CD is one, the divider is bypassed and inactive. The Clock Selection field in the Mode Register (MR.USCLKS) selects clock source between:

- CLK_USART (internal clock)
- CLK_USART/DIV (a divided CLK_USART, refer to Module Configuration section)
- CLK (external clock, available on the CLK pin)

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 at least 4.5 times longer than those provided by CLK_USART.

Figure 20-2. Baud Rate Generator



20.6.1.1 *Baud Rate in Asynchronous Mode*

If the USART is configured to operate in an asynchronous mode, the selected clock is divided by the CD value before it is provided to the receiver as a sampling clock. Depending on the Oversampling Mode bit (MR.OVER) value, the clock is then divided by either 8 (OVER=1), or 16 (OVER=0). The baud rate is calculated with the following formula:

$$BaudRate = \frac{SelectedClock}{(8(2-OVER)CD)}$$

This gives a maximum baud rate of CLK_USART divided by 8, assuming that CLK_USART is the fastest clock possible, and that OVER is one.

20.6.1.2 *Baud Rate Calculation Example*

Table 20-3 shows calculations based on the CD field to obtain 38400 baud from different source clock frequencies. This table also shows the actual resulting baud rate and error.

Table 20-3. Baud Rate Example (OVER=0)

Source Clock (Hz)	Expected Baud Rate (bit/s)	Calculation Result	CD	Actual Baud Rate (bit/s)	Error
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 (OVER=0):

$$BaudRate = (CLKUSART)/(CD \times 16)$$

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 Baud Rate in Synchronous and SPI Mode

If the USART is configured to operate in synchronous mode, the selected clock is divided by the BRGR.CD field. This does not apply when CLK is selected.

$$BaudRate = \frac{SelectedClock}{CD}$$

When CLK is selected the external frequency must be at least 4.5 times lower than the system clock, and when either CLK or CLK_USART/DIV are selected, CD must be even to ensure a 50/50 duty cycle. If CLK_USART is selected, the generator ensures this regardless of value.

20.6.2 Receiver and Transmitter Control

After a reset, the transceiver is disabled. The receiver/transmitter is enabled by writing a one to either the Receiver Enable, or Transmitter Enable bit in the Control Register (CR.RXEN, or CR.TXEN). They may be enabled together and can be configured both before and after they have been enabled. The user can reset the USART receiver/transmitter at any time by writing a one to either the Reset Receiver (CR.RSTRX), or Reset Transmitter (CR.RSTTX) bit. This software reset clears status bits and resets internal state machines, immediately halting any communication. The user interface configuration registers will retain their values.

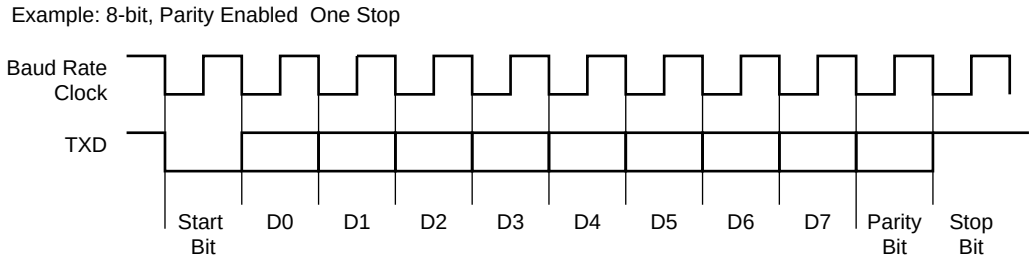
The user can disable the receiver/transmitter by writing a one to either the Receiver Disable, or Transmitter Disable bit (CR.RXDIS, or CR.TXDIS). If the receiver is disabled during a character reception, the USART will wait for the current character to be received before disabling. If the transmitter is disabled during transmission, the USART will wait until both the current character and the character stored in the Transmitter Holding Register (THR) are transmitted before disabling. If a timeguard has been implemented it will remain functional during the transaction.

20.6.3 Synchronous and Asynchronous Modes

20.6.3.1 Transmitter Operations

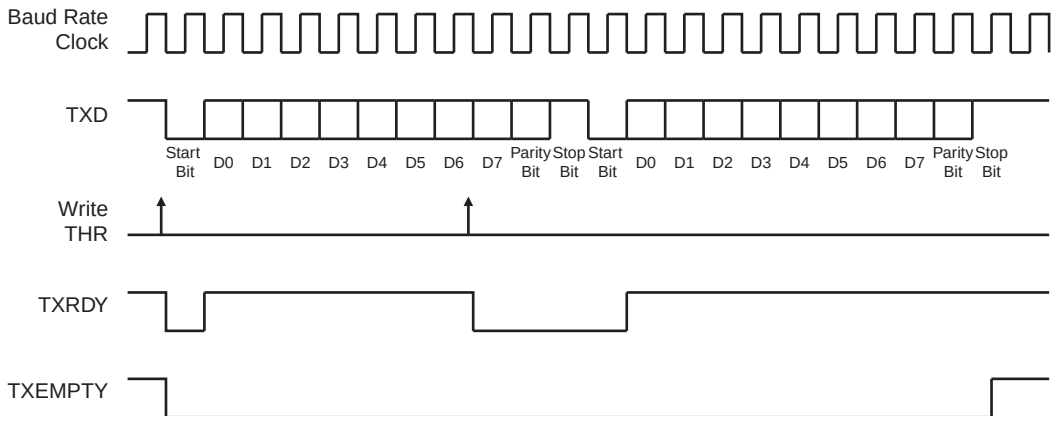
The transmitter performs equally in both synchronous and asynchronous operating modes (MR.SYNC). One start bit, up to 9 data bits, an optional parity bit, and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the serial clock. The number of data bits is selected by the Character Length field (MR.CHRL) and the MR.MODE9 bit. Nine bits are selected by writing a one to MODE9, overriding any value in CHRL. The parity bit configuration is selected in the MR.PAR field. The Most Significant Bit First bit (MR.MSBF) selects which data bit to send first. The number of stop bits is selected by the MR.NBSTOP field. The 1.5 stop bit configuration is only supported in asynchronous mode.

Figure 20-3. Character Transmit



The characters are sent by writing to the Character to be Transmitted field (THR.TXCHR). The transmitter reports status with the Transmitter Ready (TXRDY) and Transmitter Empty (TXEMPTY) bits in the Channel Status Register (CSR). TXRDY is set when THR is empty. TXEMPTY is set when both THR and the transmit shift register are empty (transmission complete). Both TXRDY and TXEMPTY are cleared when the transmitter is disabled. Writing a character to THR while TXRDY is zero has no effect and the written character will be lost.

Figure 20-4. Transmitter Status



20.6.3.2 Asynchronous Receiver

If the USART is configured in an asynchronous operating mode (MR.SYNC = 0), the receiver will oversample the RXD input line by either 8 or 16 times the baud rate clock, as selected by the Oversampling Mode bit (MR.OVER). If the line is zero for half a bit period (four or eight consecutive samples, respectively), a start bit will be assumed, and the following 8th or 16th sample will determine the logical value on the line, in effect resulting in bit values being determined at the middle of the bit period.

The number of data bits, endianess, parity mode, and stop bits are selected by the same bits and fields as for the transmitter (MR.CHRL, MODE9, MSBF, PAR, and NBSTOP). The synchronization mechanism will only consider one stop bit, regardless of the used protocol, and when the first stop bit has been sampled, the receiver will automatically begin looking for a new start bit, enabling resynchronization even if there is a protocol miss-match. Figure 20-5 and Figure 20-6 illustrate start bit detection and character reception in asynchronous mode.

Figure 20-5. Asynchronous Start Bit Detection

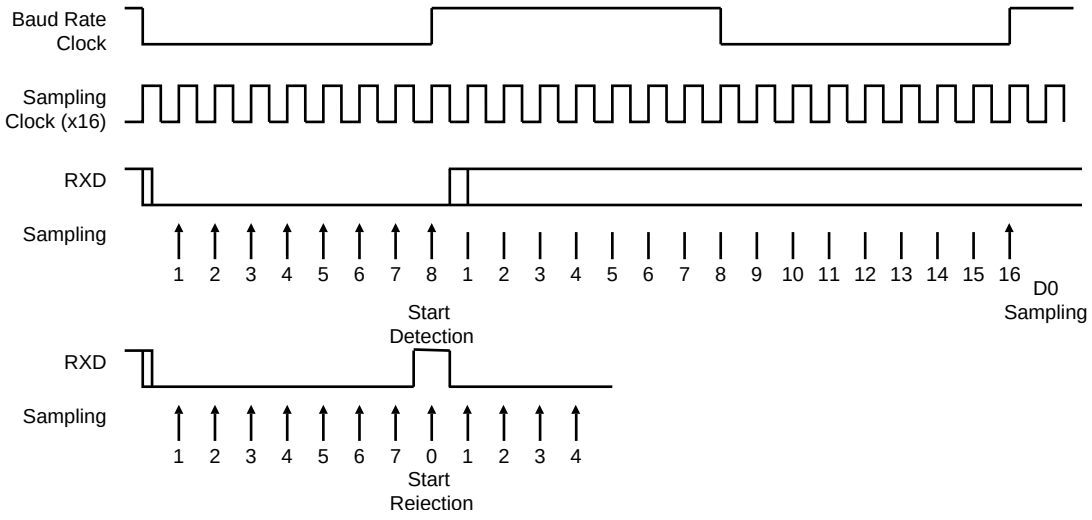
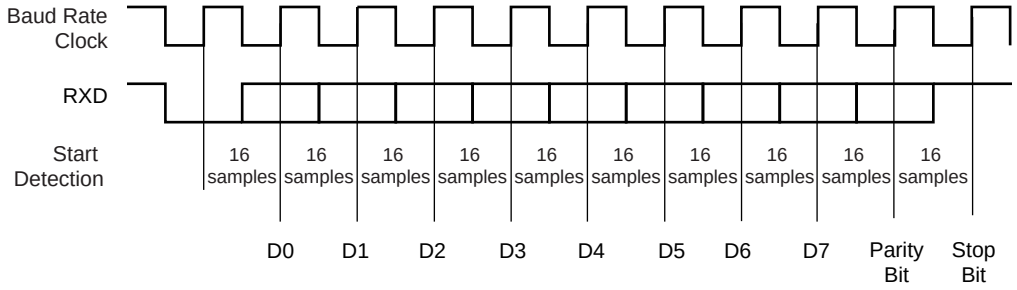


Figure 20-6. Asynchronous Character Reception

Example: 8-bit, Parity Enabled

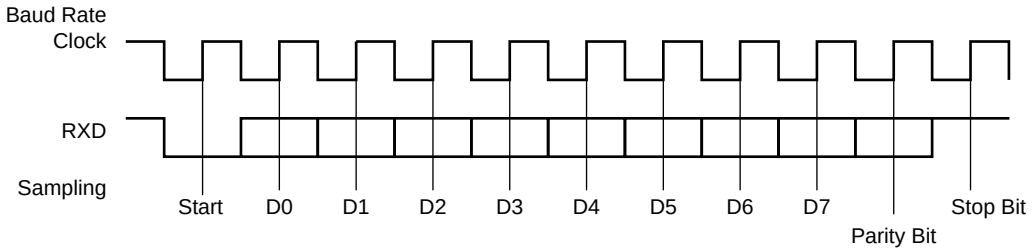


20.6.3.3 Synchronous Receiver

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 bit. Configuration bits and fields are the same as in asynchronous mode.

Figure 20-7. Synchronous Mode Character Reception

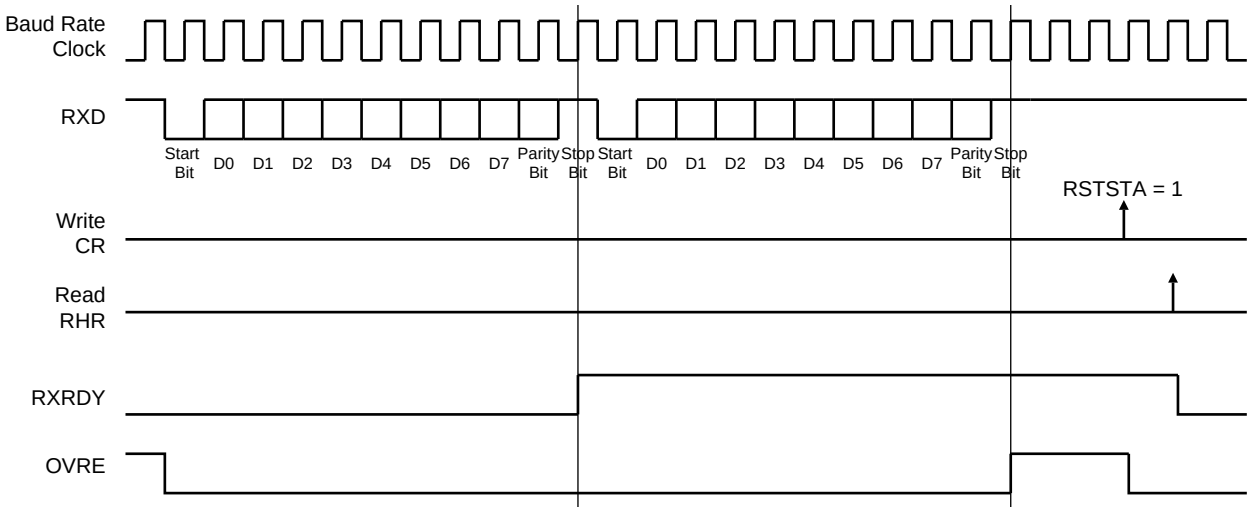
Example: 8-bit, Parity Enabled 1 Stop



20.6.3.4 Receiver Operations

When a character reception is completed, it is transferred to the Received Character field in the Receive Holding Register (RHR.RXCHR), and the Receiver Ready bit in the Channel Status Register (CSR.RXRDY) is set. If RXRDY is already set, RHR will be overwritten and the Overrun Error bit (CSR.OVRE) is set. Reading RHR will clear RXRDY, and writing a one to the Reset Status bit in the Control Register (CR.RSTSTA) will clear OVRE.

Figure 20-8. Receiver Status



20.6.3.5 Parity

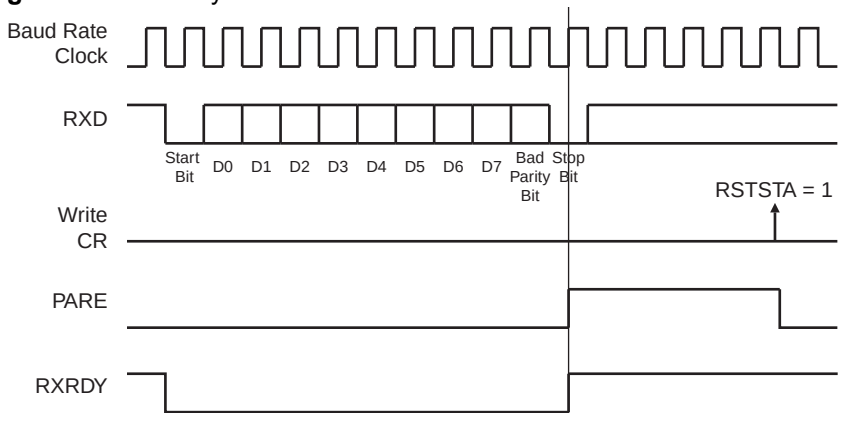
The USART supports five parity modes selected by MR.PAR. The PAR field also enables the Multidrop mode, see ”Multidrop Mode” on page 346. If even parity is selected, the parity bit will be a zero if there is an even number of ones in the data character, and if there is an odd number it will be a one. For odd parity the reverse applies. If space or mark parity is chosen, the parity bit will always be a zero or one, respectively. See Table 20-4.

Table 20-4. Parity Bit Examples

Alphanum Character	Hex	Bin	Parity Mode				
			Odd	Even	Mark	Space	None
A	0x41	0100 0001	1	0	1	0	-
V	0x56	0101 0110	1	0	1	0	-
R	0x52	0101 0010	0	1	1	0	-

The receiver will report parity errors in CSR.PARE, unless parity is disabled. Writing a one to CR.RSTSTA will clear PARE. See Figure 20-9

Figure 20-9. Parity Error



20.6.3.6 *Multidrop Mode*

If PAR is either 0x6 or 0x7, the USART runs in Multidrop mode. This mode differentiates data and address characters. Data has the parity bit zero and addresses have a one. By writing a one to the Send Address bit (CR.SENDA) the user will cause the next character written to THR to be transmitted as an address. Receiving a character with a one as parity bit will set PARE.

20.6.3.7 *Transmitter Timeguard*

The timeguard feature enables the USART to interface slow devices by inserting an idle state on the TXD line in between two characters. This idle state corresponds to a long stop bit, whose duration is selected by the Timeguard Value field in the Transmitter Timeguard Register (TTGR.TG). The transmitter will hold the TXD line high for TG bit periods, in addition to the number of stop bits. As illustrated in Figure 20-10, the behavior of TXRDY and TXEMPTY is modified when TG has a non-zero value. If a pending character has been written to THR, the TXRDY bit will not be set until this characters start bit has been sent. TXEMPTY will remain low until the timeguard transmission has completed.

Figure 20-10. Timeguard Operation

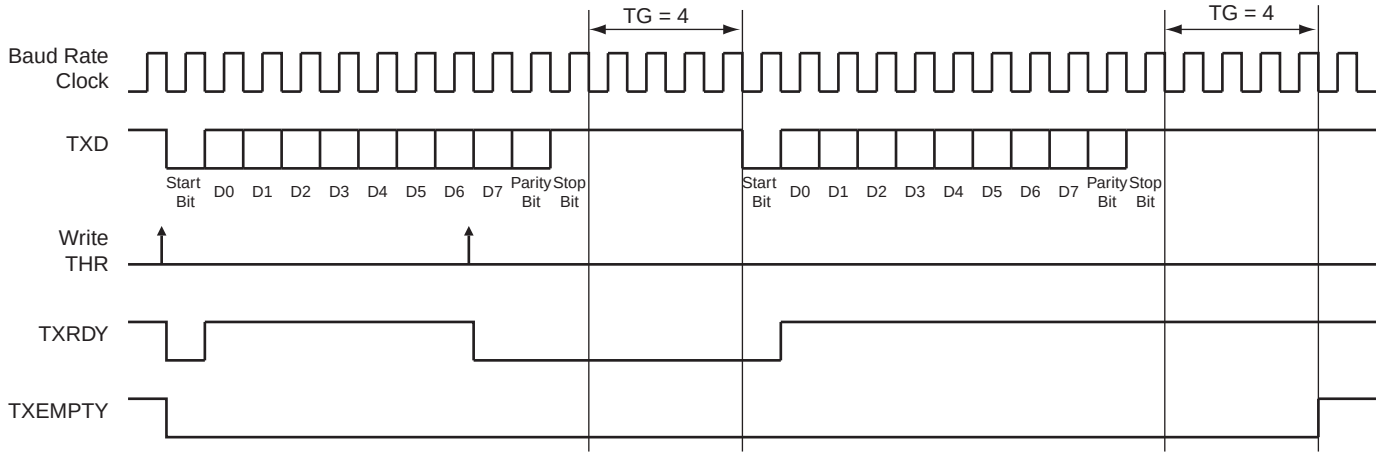


Table 20-5. Maximum Baud Rate Dependent Timeguard Durations

Baud Rate (bit/sec)	Bit time (µs)	Timeguard (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 Time-out Value field in the Receiver Time-out Register (RTOR.TO) enables handling of variable-length frames by detection of selectable idle durations on the RXD line. The value written to

TO is loaded to a decremental counter, and unless it is zero, a time-out will occur when the amount of inactive bit periods match the initial counter value. If a time-out has not occurred, the counter will reload and restart every time a new character arrives. A time-out sets the TIMEOUT bit in CSR. Clearing TIMEOUT can be done in two ways:

- Writing a one to the Start Time-out bit (CR.STTTO). This also aborts count down until the next character has been received.
- Writing a one to the Reload and Start Time-out bit (CR.RETTO). This also reloads the counter and restarts count down immediately.

Figure 20-11. Receiver Time-out Block Diagram

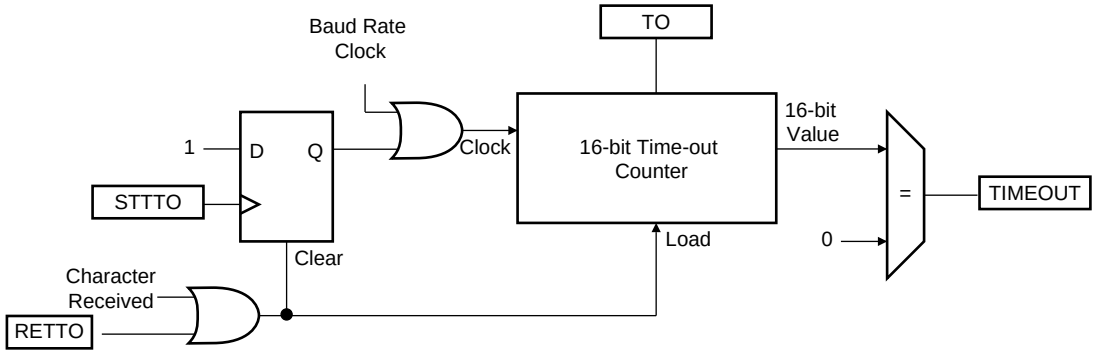


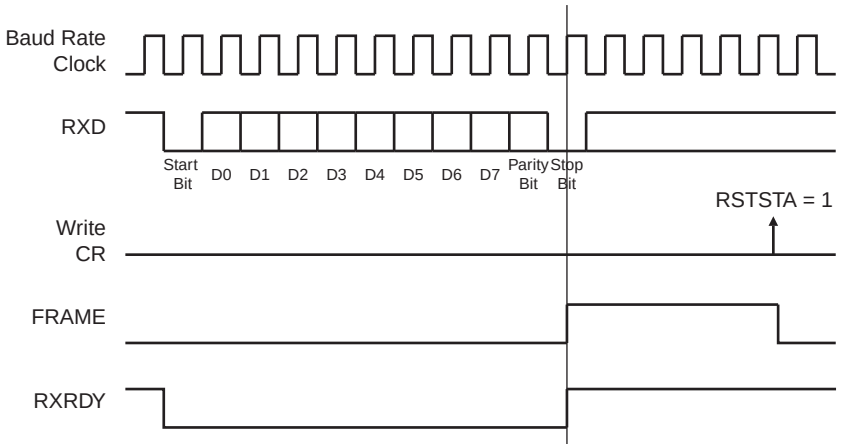
Table 20-6. Maximum Time-out Period

Baud Rate (bit/sec)	Bit Time (µs)	Time-out (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
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 has occurred if a stop bit reads as zero. This can occur if the transmitter and receiver are not synchronized. A framing error is reported by CSR.FRAME as soon as the error is detected, at the middle of the stop bit.

Figure 20-12. Framing Error Status

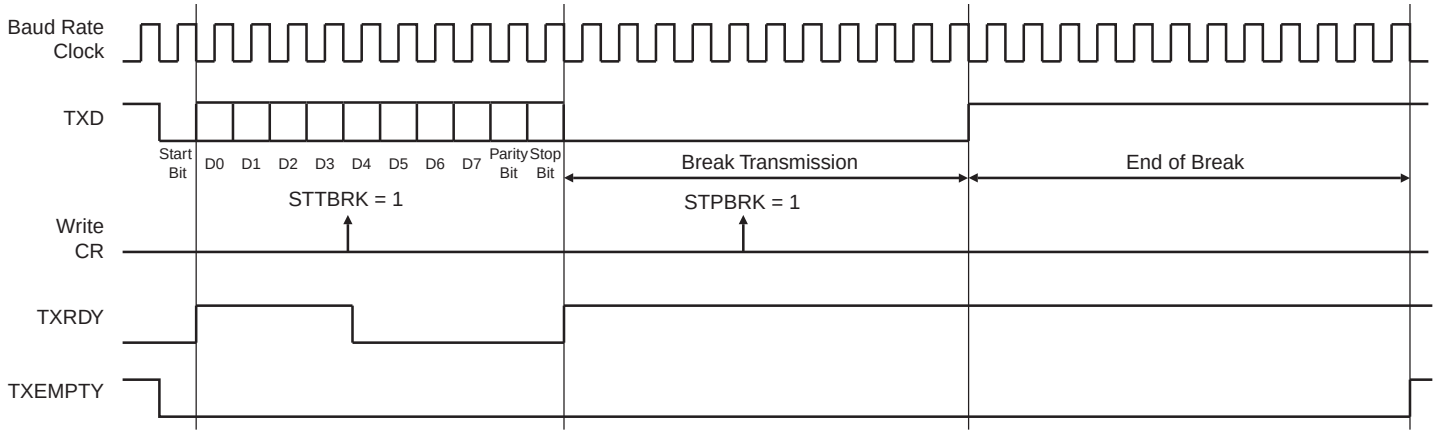


20.6.3.10 Transmit Break

When TXRDY is set, the user can request the transmitter to generate a break condition on the TXD line by writing a one to The Start Break bit (CR.STTBRK). The break is treated as a normal 0x00 character transmission, clearing TXRDY and TXEMPTY, but with zeroes for preambles, start, parity, stop, and time guard bits. Writing a one to the Stop Break bit (CR.STBRK) will stop the generation of new break characters, and send ones for TG duration or at least 12 bit periods, ensuring that the receiver detects end of break, before resuming normal operation. Figure 20-13 illustrates STTBRK and STPBRK effect on the TXD line.

Writing to STTBRK and STPBRK simultaneously can lead to unpredictable results. Writes to THR before a pending break has started will be ignored.

Figure 20-13. Break Transmission



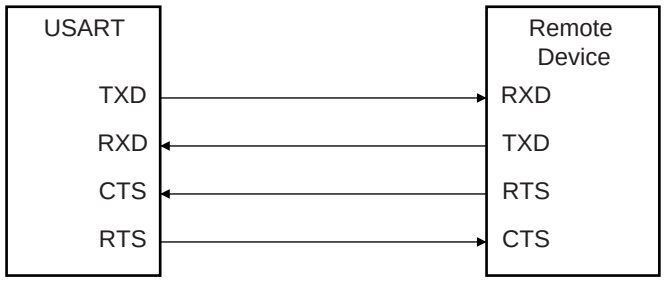
20.6.3.11 Receive Break

A break condition is assumed when incoming data, parity, and stop bits are zero. This corresponds to a framing error, but FRAME will remain zero while the Break Received/End Of Break bit (CSR.RXBRK) is set. Writing a one to CR.RSTSTA will clear RXBRK. An end of break will also set RXBRK, and is assumed when TX is high for at least 2/16 of a bit period in asynchronous mode, or when a high level is sampled in synchronous mode.

20.6.3.12 *Hardware Handshaking*

The USART features an out-of-band hardware handshaking flow control mechanism, implementable by connecting the RTS and CTS pins with the remote device, as shown in Figure 20-14.

Figure 20-14. Connection with a Remote Device for Hardware Handshaking



Writing 0x2 to the MR.MODE field configures the USART to operate in this mode. The receiver will drive its RTS pin high when disabled or when the Reception Buffer Full bit (CSR.RXBUFF) is set by the Buffer Full signal from the Peripheral DMA controller. If the receivers RTS pin is high, the transmitters CTS pin will also be high and only the active character transactions will be completed. Allocating a new buffer to the DMA controller by clearing RXBUFF, will drive the RTS pin low, allowing the transmitter to resume transmission. Detected level changes on the CTS pin can trigger interrupts, and are reported by the CTS Input Change bit in the Channel Status Register (CSR.CTSIC).

Figure 20-15 illustrates receiver functionality, and Figure 20-16 illustrates transmitter functionality.

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

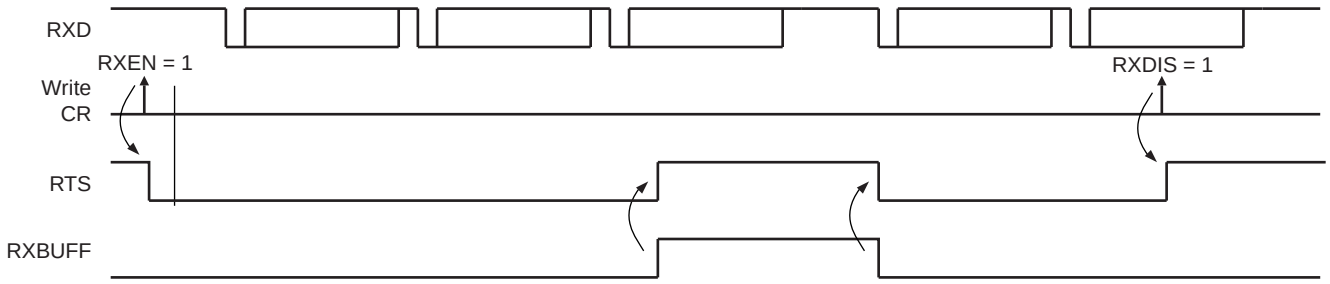


Figure 20-16. Transmitter Behavior when Operating with Hardware Handshaking

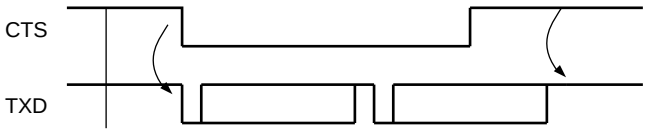


Figure 20-17.

20.6.4 SPI Mode

The USART features a Serial Peripheral Interface (SPI) link compliant mode, supporting synchronous, full-duplex communication, in both master and slave mode. Writing 0xE (master) or 0xF (slave) to MR.MODE will enable this mode. A SPI in master mode controls the data flow to and from the other SPI devices, who are in slave mode. It is possible to let devices take turns being masters (aka multi-master protocol), and one master may shift data simultaneously into several slaves, but only one slave may respond at a time. A slave is selected when its slave select (NSS) signal has been raised by the master. The USART can only generate one NSS signal, and it is possible to use standard I/O lines to address more than one slave.

20.6.4.1 Modes of Operation

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This line supplies the data shifted from master to slave. In master mode this is connected to TXD, and in slave mode to RXD.
- Master In Slave Out (MISO): This line supplies the data shifted from slave to master. In master mode this is connected to RXD, and in slave mode to TXD.
- Serial Clock (CLK): This is controlled by the master. One period per bit transmission. In both modes this is connected to CLK.
- Slave Select (NSS): This control line allows the master to select or deselect a slave. In master mode this is connected to RTS, and in slave mode to CTS.

Changing SPI mode after initial configuration has to be followed by a transceiver software reset in order to avoid unpredictable behavior.

20.6.4.2 Baud Rate

The baud rate generator operates as described in ”Baud Rate in Synchronous and SPI Mode” on page 341, with the following requirements:

In SPI Master Mode:

- The Clock Selection field (MR.USCLKS) must not equal 0x3 (external clock, CLK).
- The Clock Output Select bit (MR.CLKO) must be one.
- The BRGR.CD field must be at least 0x4.
- If USCLKS is one (internal divided clock, CLK_USART/DIV), the value in CD has to be even, ensuring a 50:50 duty cycle. CD can be odd if USCLKS is zero (internal clock, CLK_USART).

In SPI Slave Mode:

- CLK frequency must be at least four times lower than the system clock.

20.6.4.3 Data Transfer

- Up to nine data bits are successively shifted out on the TXD pin at each edge. There are no start, parity, or stop bits, and MSB is always sent first. The SPI Clock Polarity (MR.CPOL), and SPI Clock Phase (MR.CPHA) bits configure CLK by selecting the edges upon which bits are shifted and sampled, resulting in four non-interoperable protocol modes see Table 20-7. A master/slave pair must use the same configuration, and the master must be reconfigured if it is to communicate with slaves using different configurations. See Figures 20-18 and 20-19.

Table 20-7. SPI Bus Protocol Modes

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)

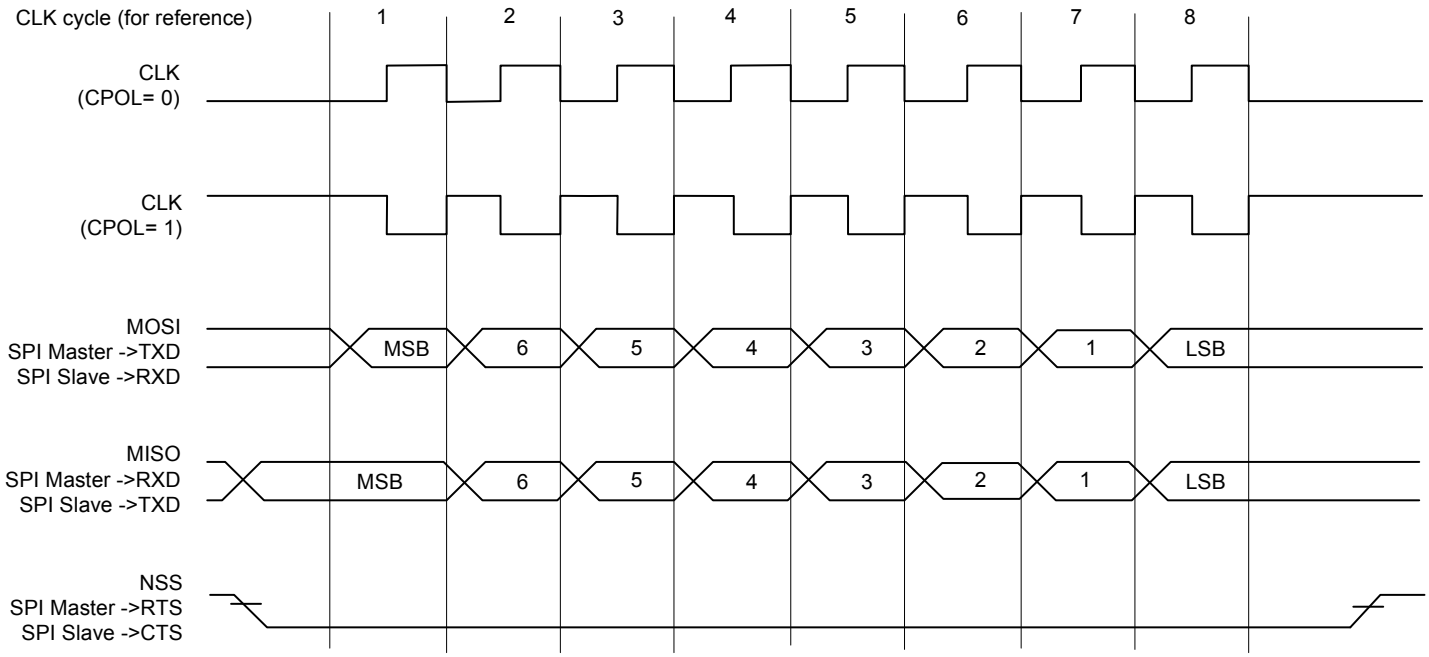
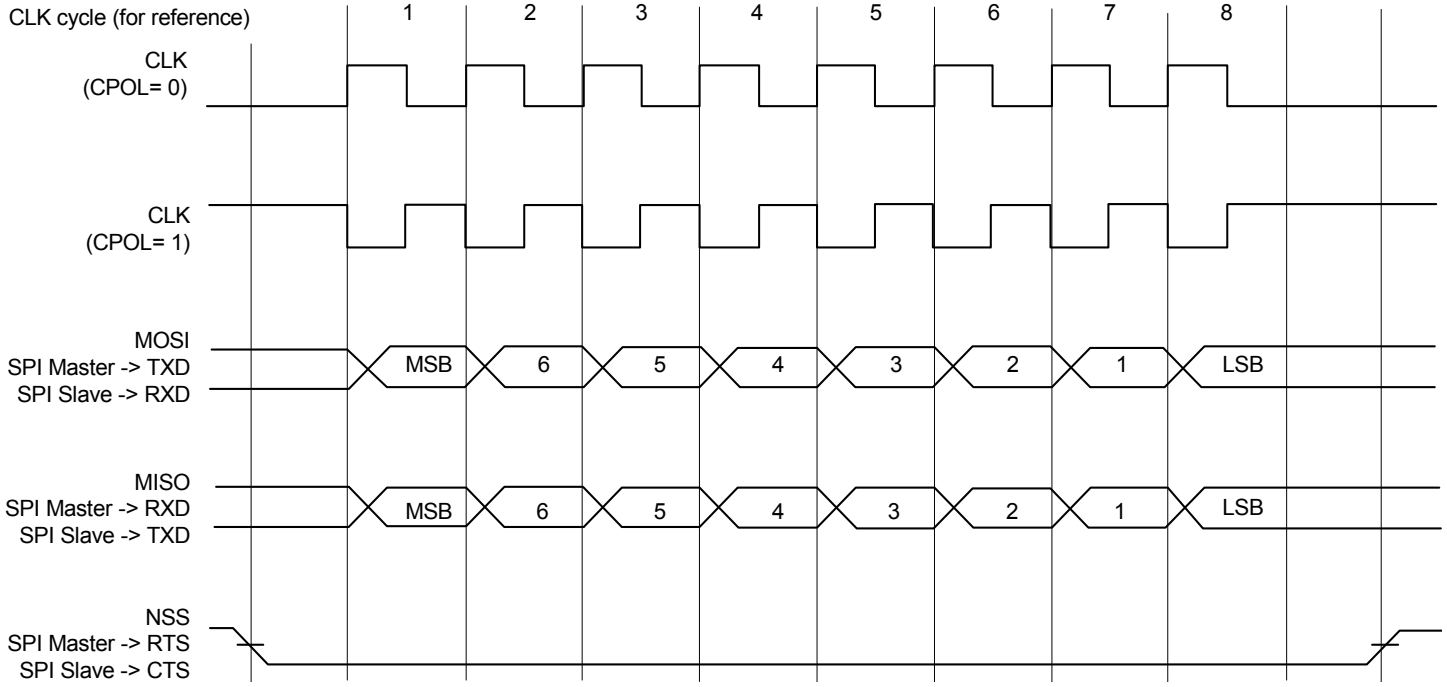


Figure 20-19. SPI Transfer Format (CPHA=0, 8 bits per transfer)



20.6.4.4 Receiver and Transmitter Control

See ”Transmitter Operations” on page 342, and ”Receiver Operations” on page 344.

20.6.4.5 Character Transmission and Reception

In SPI master mode, the slave select line (NSS) is asserted low one bit period before the start of transmission, and released high one bit period after every character transmission. A delay for at least three bit periods is always inserted in between characters. In order to address slave devices supporting the Chip Select Active After Transfer (CSAAT) mode, NSS can be forced low by writing a one to the Force SPI Chip Select bit (CR.RTSEN/FCS). Releasing NSS when FCS is one, is only possible by writing a one to the Release SPI Chip Select bit (CR.RTSDIS/RCS).

In SPI slave mode, a low level on NSS for at least one bit period will allow the slave to initiate a transmission or reception. The Underrun Error bit (CSR.UNRE) is set if a character must be sent while THR is empty, and TXD will be high during character transmission, as if 0xFF was being sent. If a new character is written to THR it will be sent correctly during the next transmission slot. Writing a one to CR.RSTSTA will clear UNRE. To ensure correct behavior of the receiver in SPI slave mode, the master device sending the frame must ensure a minimum delay of one bit period in between each character transmission.

20.6.4.6 Receiver Time-out

Receiver Time-out’s are not possible in SPI mode as the baud rate clock is only active during data transfers.

20.6.5

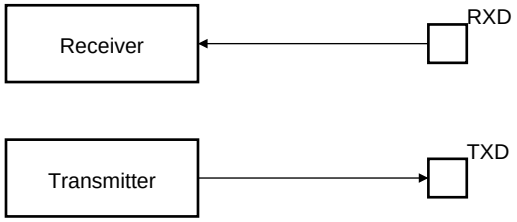
20.6.6 Test Modes

The internal loopback feature enables on-board diagnostics, and allows the USART to operate in three different test modes, with reconfigured pin functionality, as shown below.

20.6.6.1 Normal Mode

During normal operation, a receivers RXD pin is connected to a transmitters TXD pin.

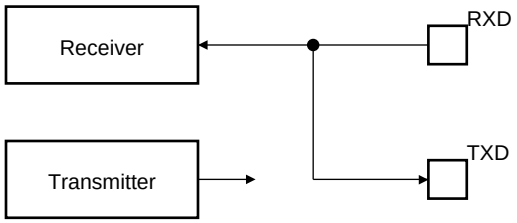
Figure 20-20. Normal Mode Configuration



20.6.6.2 Automatic Echo Mode

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is also sent to the TXD pin, as shown in Figure 20-21. Transmitter configuration has no effect.

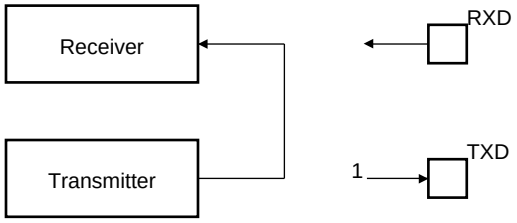
Figure 20-21. Automatic Echo Mode Configuration



20.6.6.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 20-22. 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.

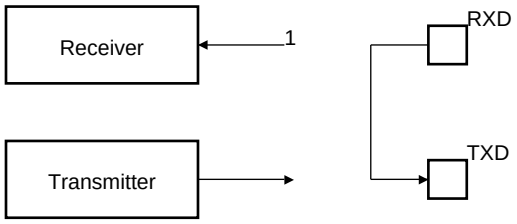
Figure 20-22. Local Loopback Mode Configuration



20.6.6.4 Remote Loopback Mode

Remote loopback mode connects the RXD pin to the TXD pin, as shown in Figure 20-23. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 20-23. Remote Loopback Mode Configuration



20.6.7 Write Protection Registers

To prevent single software errors from corrupting USART behavior, certain address spaces can be write-protected by writing the correct Write Protect KEY and a one to the Write Protect Enable bit in the Write Protect Mode Register (WPMR.WPKEY, and WPMR.WPEN). Disabling the write protection is done by writing the correct key, and a zero to WPEN.

Write attempts to a write protected register are detected and the Write Protect Violation Status bit in the Write Protect Status Register (WPSR.WPVS) is set, while the Write Protect Violation Source field (WPSR.WPVSRC) indicates the targeted register. Writing the correct key to the Write Protect KEY bit (WPMR.WPKEY) clears WPVSRC and WPVS.

The protected registers are:

- ”Mode Register” on page 358
- ”Baud Rate Generator Register” on page 368
- ”Receiver Time-out Register” on page 369
- ”Transmitter Timeguard Register” on page 370

20.7 User Interface

Table 20-8. USART Register Memory Map

Offset	Register	Name	Access	Reset
0x0000	Control Register	CR	Write-only	0x00000000
0x0004	Mode Register	MR	Read-write	0x00000000
0x0008	Interrupt Enable Register	IER	Write-only	0x00000000
0x000C	Interrupt Disable Register	IDR	Write-only	0x00000000
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	0x00000000
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
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: 0x00000000

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

23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS/RCS	RTSEN/FCS	–	–

15	14	13	12	11	10	9	8
RETTO	RSTNACK	–	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RTSDIS/RCS: Request to Send Disable/Release SPI Chip Select**
  Writing a zero to this bit has no effect.
  Writing a one to this bit when USART is not in SPI master mode drives RTS pin high.
  Writing a one to this bit when USART is in SPI master mode releases NSS (RTS pin).
- **RTSEN/FCS: Request to Send Enable/Force SPI Chip Select**
  Writing a zero to this bit has no effect.
  Writing a one to this bit when USART is not in SPI master mode drives RTS low.
  Writing a one to this bit when USART is in SPI master mode when;
  FCS=0: has no effect.
  FCS=1: forces NSS (RTS pin) low, even if USART is not transmitting, in order to address SPI slave devices supporting the CSAAT Mode (Chip Select Active After Transfer).
- **RETTO: Rearm Time-out**
  Writing a zero to this bit has no effect.
  Writing a one to this bit reloads the time-out counter and clears CSR.TIMEOUT.
- **RSTNACK: Reset Non Acknowledge**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears CSR.NACK.
- **SENDA: Send Address**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will in multidrop mode send the next character written to THR as an address.
- **STTTO: Start Time-out**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will abort any current time-out count down, and trigger a new count down when the next character has been received. CSR.TIMEOUT is also cleared.
- **STPBRK: Stop Break**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will stop the generation of break signal characters, and then send ones for TTGR.TG duration, or at least 12 bit periods. No effect if no break is being transmitted.

- **STTBRK: Start Break**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will start transmission of break characters when current characters present in THR and the transmit shift register have been sent. No effect if a break signal is already being generated.
- **RSTSTA: Reset Status Bits**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the following bits in CSR: PARE, FRAME, OVRE, and RXBRK.
- **TXDIS: Transmitter Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the transmitter.
- **TXEN: Transmitter Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the transmitter if TXDIS is zero.
- **RXDIS: Receiver Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the receiver.
- **RXEN: Receiver Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the receiver if RXDIS is zero.
- **RSTTX: Reset Transmitter**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will reset the transmitter.
- **RSTRX: Reset Receiver**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will reset the receiver.

20.7.2 Mode Register

Name: MR

Access Type: Read-write

Offset: 0x4

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–		

23	22	21	20	19	18	17	16
–	–	–	INACK	OVER	CLKO	MODE9	MSBF/CPOL

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC/CPHA

7	6	5	4	3	2	1	0
CHRL		USCLKS		MODE			

This register can only be written if the WPEN bit is cleared in the Write Protect Mode Register.

- **INACK: Inhibit Non Acknowledge**
  0: The NACK is generated.
  1: The NACK is not generated.
- **OVER: Oversampling Mode**
  0: Oversampling at 16 times the baud rate.
  1: Oversampling at 8 times the baud rate.
- **CLKO: Clock Output Select**
  0: The USART does not drive the CLK pin.
  1: The USART drives the CLK pin unless USCLKS selects the external clock.
- **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:
  MSBF=0: Least Significant Bit is sent/received first.
  MSBF=1: Most Significant Bit is sent/received first.
  If USART operates in SPI Mode, CPOL is used with CPHA to produce the required clock/data relationship between devices.
  CPOL=0: The inactive state value of CLK is logic level zero.
  CPOL=1: The inactive state value of CLK is logic level one.

- **CHMODE: Channel Mode**

Table 20-9.

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 20-10.

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 20-11.

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, CPHA determines which edge of CLK 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.
  CPHA = 0: Data is changed on the leading edge of CLK and captured on the following edge of CLK.
  CPHA = 1: Data is captured on the leading edge of CLK and changed on the following edge of CLK.

- **CHRL: Character Length.**

Table 20-12.

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

- **USCLKS: Clock Selection**

Table 20-13.

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 20-14.

MODE				Mode of the USART
0	0	0	0	Normal
0	0	1	0	Hardware Handshaking
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: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
15	14	13	12	11	10	9	8
–	–	NACK	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: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
15	14	13	12	11	10	9	8
–	–	NACK	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: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–
15	14	13	12	11	10	9	8
–	–	NACK	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: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CTS	–	–	–	CTSIC	–	–	–
15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	–	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

- **CTS: Image of CTS Input**
  0: CTS is low.
  1: CTS is high.
- **CTSIC: Clear to Send Input Change Flag**
  0: No change has been detected on the CTS pin since the last CSR read.
  1: At least one change has been detected on the CTS pin since the last CSR read.
- **NACK: Non Acknowledge**
  0: No Non Acknowledge has 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 Buffer Full signal from the Peripheral DMA Controller channel is inactive.
  1: The Buffer Full signal from the 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:
  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:
  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: The transmitter is either disabled or there are characters in THR, or in the transmit shift register.
  1: There are no characters in neither 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 (CR.STTTO), or RTOR.TO is zero.
  1: There has been a time-out since the last Start Time-out command.
- **PARE: Parity Error**
  0: Either no parity error has been detected, or the parity bit is a zero in multidrop mode, since the last RSTSTA.
  1: Either at least one parity error has been detected, or the parity bit is a one in multidrop mode, since the last RSTSTA.
- **FRAME: Framing Error**
  0: No stop bit has been found as low since the last RSTSTA.
  1: At least one stop bit has been found as 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: The transmitter is either disabled, or a character in THR is waiting to be transferred to the transmit shift register, or an STTBRK command has been requested. As soon as the transmitter is enabled, TXRDY becomes one.
  1: There is no character in the THR.
- **RXRDY: Receiver Ready**
  0: The receiver is either disabled, or no complete character has been received since the last read of RHR. 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 Receiver 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
–	–	–	–	–	–	–	RXCHR[8]

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

- **RXCHR: Received Character**
  Last received character.

20.7.8 Transmitter Holding Register

Name: THR

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
–	–	–	–	–	–	–	TXCHR[8]

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

- **TXCHR: Character to be Transmitted**
  If TXRDY is zero this field contains the next character to be transmitted.

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
–	–	–	–	–	–		

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

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

This register can only be written to if write protection is disabled, see ”Write Protect Mode Register” on page 371.

- **CD: Clock Divider**

Table 20-15.

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

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

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

This register can only be written to if write protection is disabled, see ”Write Protect Mode Register” on page 371.

- **TO: Time-out Value**
  0: The receiver Time-out is disabled.
  1 - 65535: The receiver Time-out is enabled and the time-out delay is TO x bit period.
  Note that the size of the TO counter is device dependent, 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 to if write protection is disabled, see ”Write Protect Mode Register” on page 371.

- **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 Write Protect Mode Register

Register Name: WPMR

Access Type: Read-write

Offset: 0xE4

Reset Value: See Table 20-8

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

- **WPKEY: Write Protect KEY**
  Has to be written to 0x555341 (“USA” in ASCII) in order to successfully write WPEN. Always reads as zero.
- **WPEN: Write Protect Enable**
  0 = Write protection disabled.
  1 = Write protection enabled.

Protects the registers:

- ”Mode Register” on page 358
- ”Baud Rate Generator Register” on page 368
- ”Receiver Time-out Register” on page 369
- ”Transmitter Timeguard Register” on page 370

20.7.13 Write Protect Status Register

Register Name: WPSR

Access Type: Read-only

Offset: 0xE8

Reset Value: See Table 20-8

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

23	22	21	20	19	18	17	16
WPVSRC[15:8]							

15	14	13	12	11	10	9	8
WPVSRC[7:0]							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPVS

- **WPVSRC: Write Protect Violation Source**
  If WPVS=1 this field indicates which write-protected register was unsuccessfully written to, either by address offset or code.
- **WPVS: Write Protect Violation Status**
  0= No write protect violation has occurred since the last WPSR read.
  1= A write protect violation has occurred since the last WPSR read.

Note: Reading WPSR automatically clears all fields.

20.7.14 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: -

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

23	22	21	20	19	18	17	16
–	–	–	–	MFN			

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

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

- **MFN**
  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 20-16. Module Configuration

Feature	USART0	USART1	USART2
SPI Logic	Implemented	Implemented	Implemented
LIN Logic	Not Implemented	Not Implemented	Not Implemented
RS485 Logic	Not Implemented	Not Implemented	Not Implemented
Manchester Logic	Not Implemented	Not Implemented	Not Implemented
Modem Logic	Not Implemented	Not Implemented	Not Implemented
IRDA Logic	Not Implemented	Not Implemented	Not Implemented
Fractional Baudrate	Not Implemented	Not Implemented	Not Implemented
ISO7816	Not Implemented	Not Implemented	Not Implemented
DIV	8	8	8
Receiver Time-out Counter Size (Size of the RTOR.TO field)	8-bits	8-bits	8-bits

Table 20-17. Module Clock Name

Module Name	Clock Name
USART0	**CLK_USART0**
USART1	**CLK_USART1**
USART2	**CLK_USART2**

20.8.1 Clock Connections

Each USART can be connected to an internally divided clock:

Table 20-18. USART Clock Connections

USART	Source	Name	Connection
0	Internal	CLK_DIV	PBA Clock / 8
1			
2			

20.8.2 Register Reset Values

Table 20-19.

Register	Reset Value
VERSION	0x00000440

21. Serial Peripheral Interface (SPI)

Rev: 2.1.1.3

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.

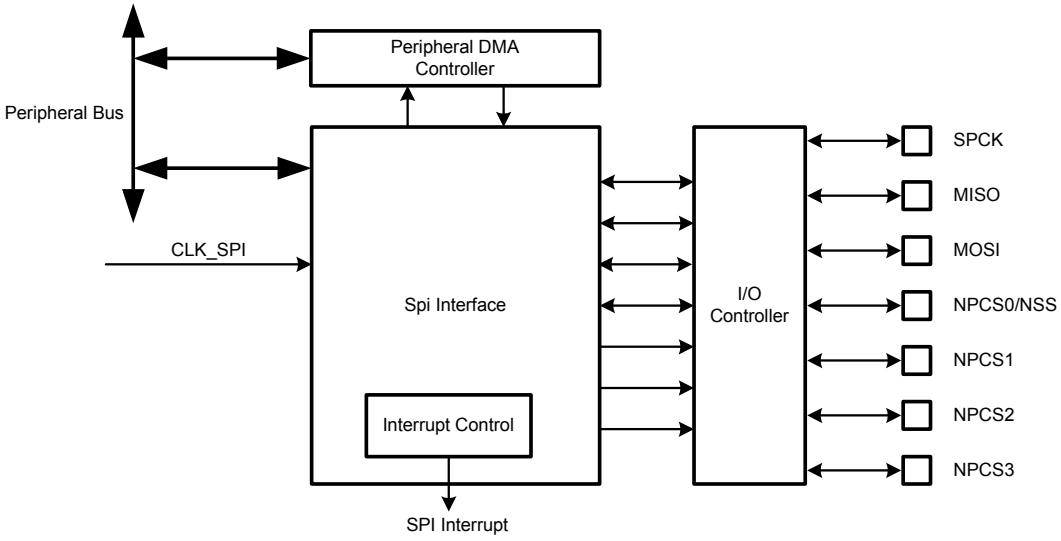
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

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.

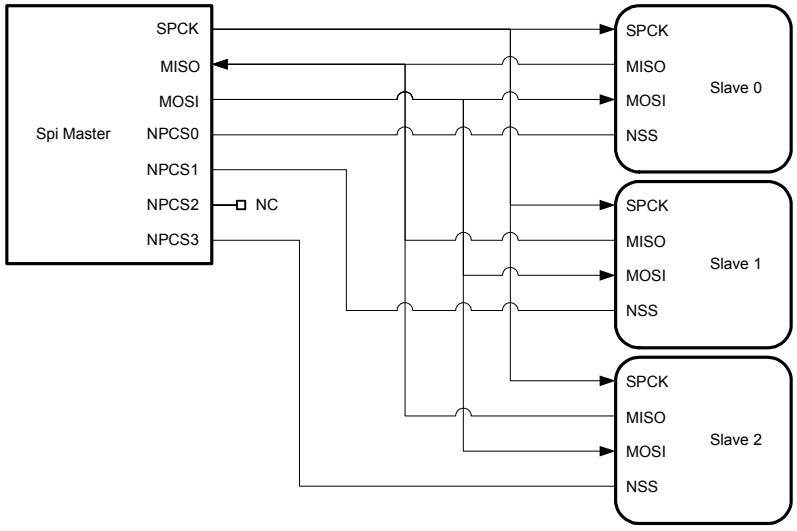
21.3 Block Diagram

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 378 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 378 and Figure 21-4 on page 379 show examples of data transfers.

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

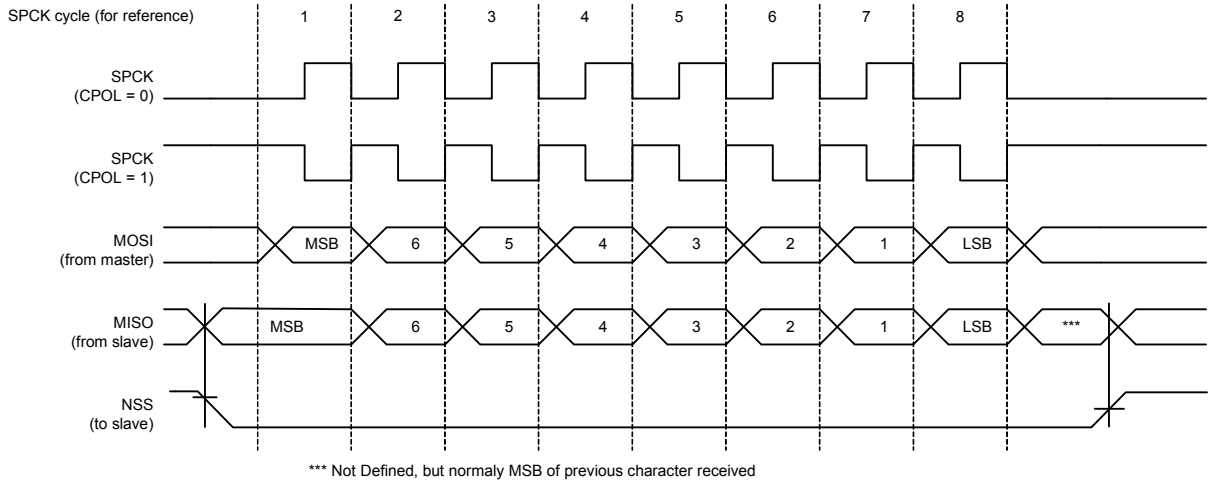
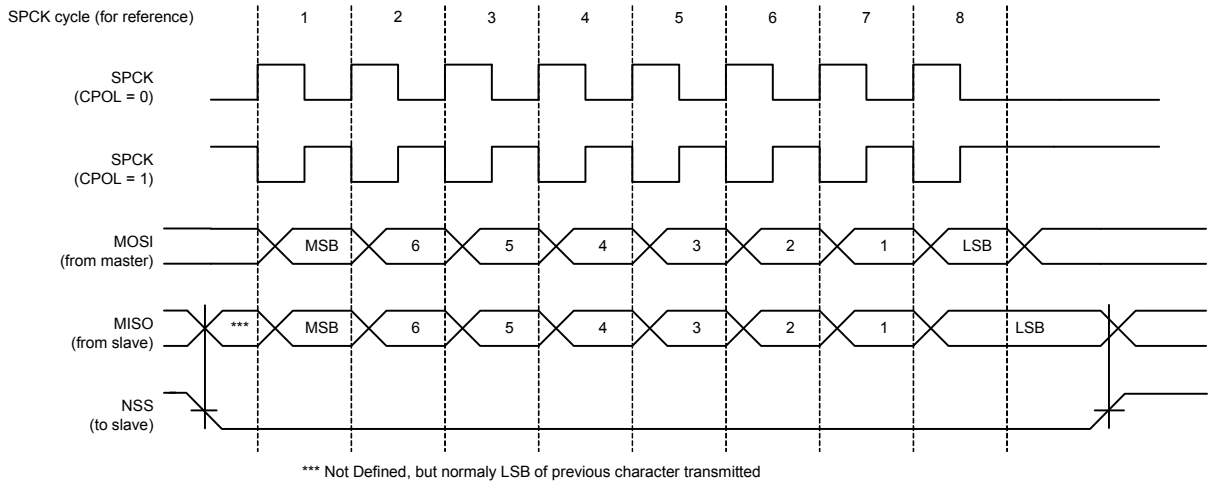


Figure 21-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



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.

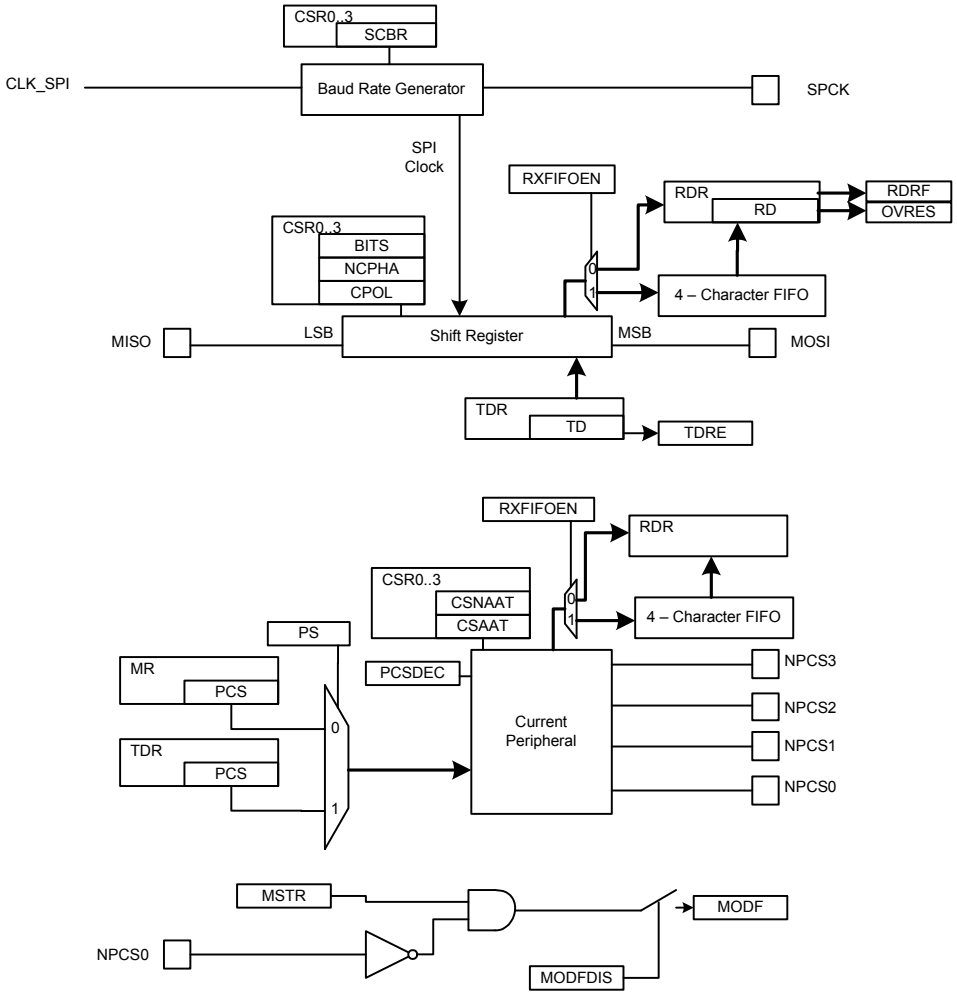
The end of transfer is indicated by the Transmission Registers Empty bit in the SR register (SR.TXEMPTY). If a transfer delay (CSRn.DLYBCT) is greater than zero for the last transfer, SR.TXEMPTY is set after the completion of said delay. The CLK_SPI can be switched off at this time.

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.

Figure 21-5 on page 380shows a block diagram of the SPI when operating in master mode. Figure 21-6 on page 381 shows a flow chart describing how transfers are handled.

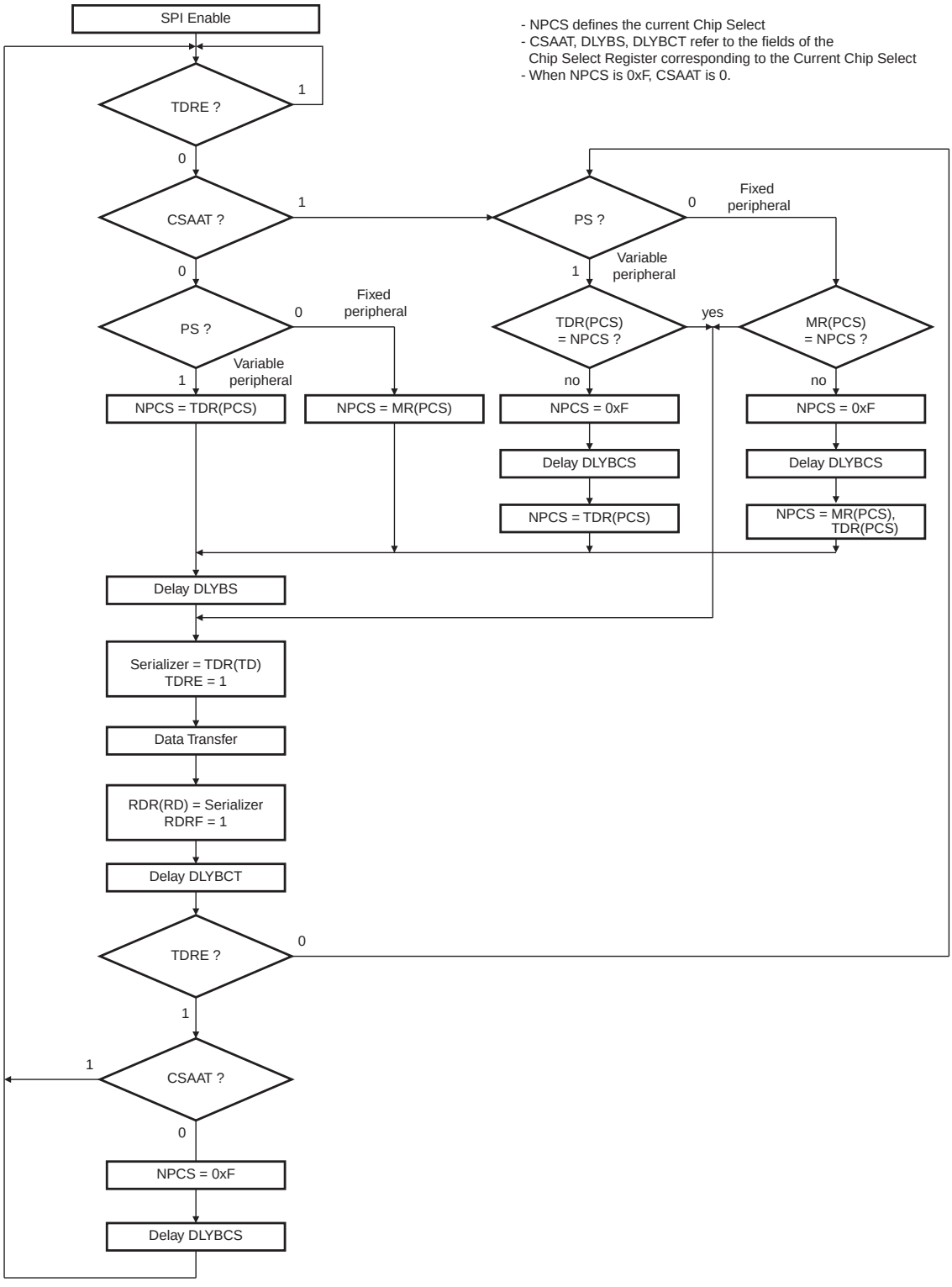
21.7.3.1 Master mode block diagram

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.

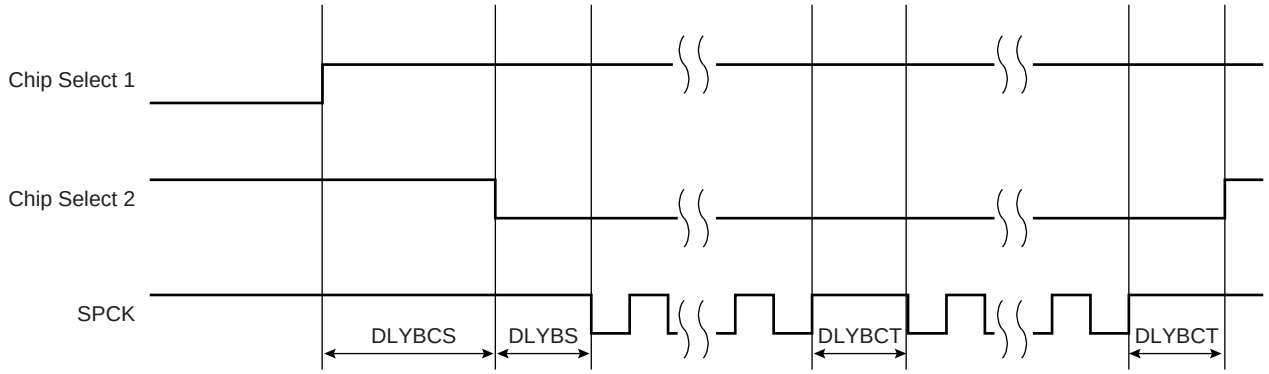
21.7.3.4 Transfer delays

Figure 21-7 on page 382 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 385 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

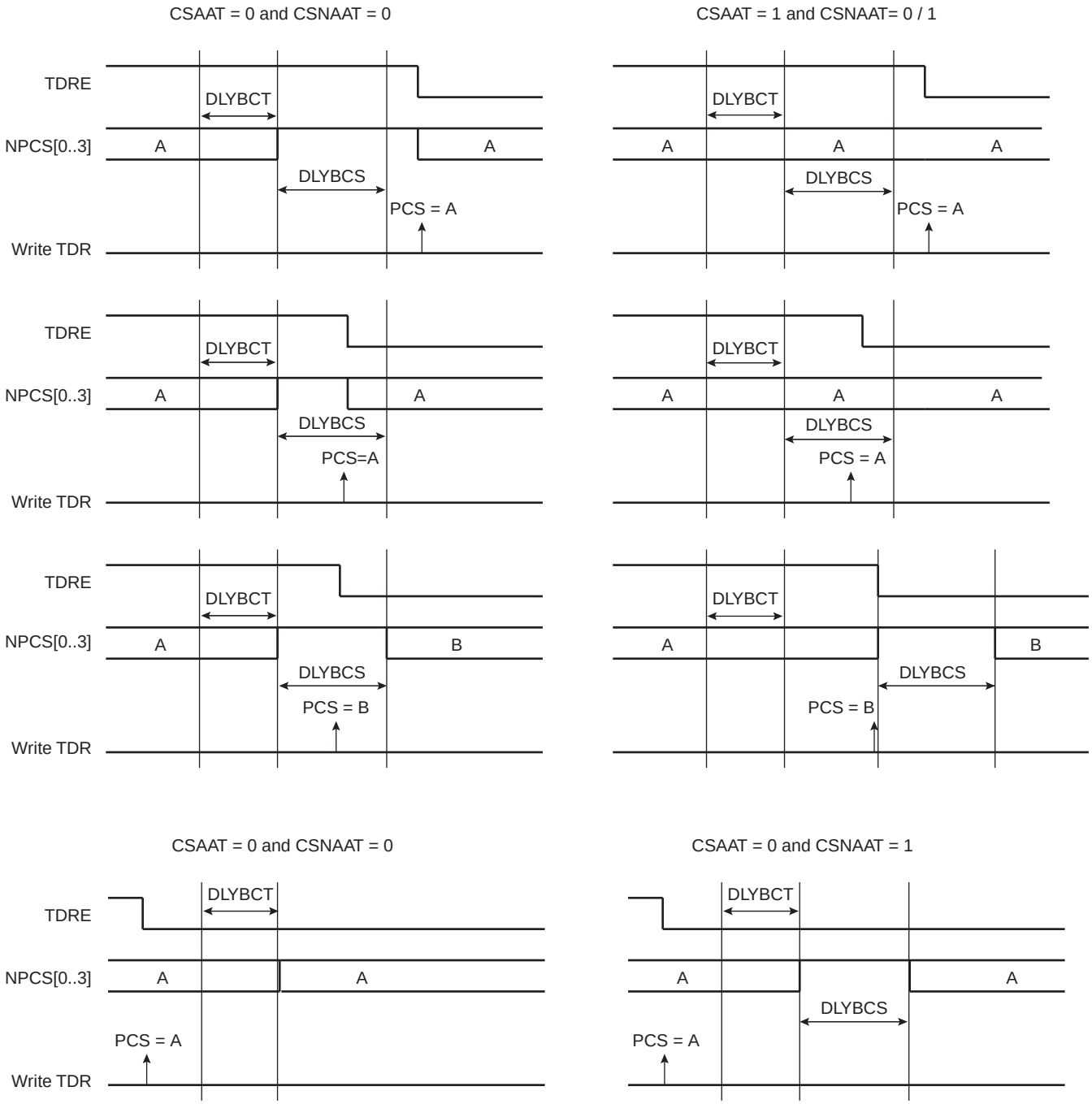


Figure 21-8 on page 385 shows different peripheral deselection cases and the effect of the CSRn.CSAAT and CSRn.CSNAAT bits.

21.7.3.9 Mode fault detection

The SPI is capable of detecting a mode fault when it is configured in master mode and NPCS0, MOSI, MISO, and SPCK are configured as open drain through the I/O Controller with either internal or external pullup resistors. If the I/O Controller does not have open-drain capability, mode fault detection **must** be disabled by writing a one to the Mode Fault Detection bit in the MR

register (MR.MODFDIS). In systems with open-drain I/O lines, a mode fault is detected when a low level is driven by an external master on the NPCS0/NSS signal.

When a mode fault is detected, the Mode Fault Error bit in the SR (SR.MODF) is set until the SR is read and the SPI is automatically disabled until re-enabled by writing a one to the SPI Enable bit in the CR register (CR.SPIEN).

By default, the mode fault detection circuitry is enabled. The user can disable mode fault detection by writing a one to the Mode Fault Detection bit in the MR register (MR.MODFDIS).

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. Data is loaded in RDR even if this flag is set. The user has to read the SR register to clear the SR.OVRES bit.

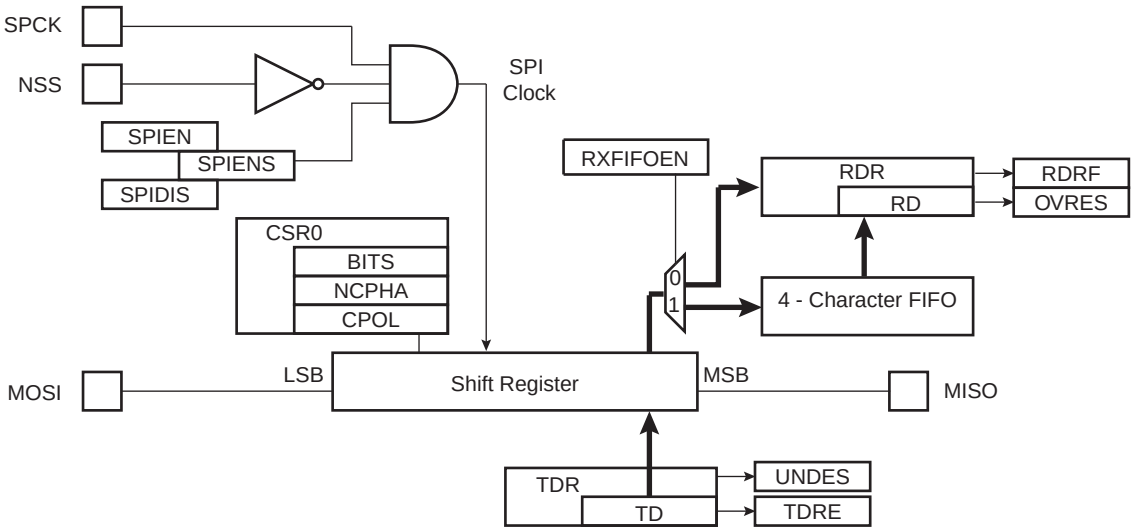
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the TDR register, the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets to zero.

When a first data is written in TDR, it is transferred immediately in the Shift Register and the SR.TDRE bit rises. If new data is written, it remains in TDR until a transfer occurs, i.e. NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in TDR is transferred in the Shift Register and the SR.TDRE bit rises. This enables frequent updates of critical variables with single transfers.

Then, a new data is loaded in the Shift Register from the TDR. In case no character is ready to be transmitted, i.e. no character has been written in TDR since the last load from TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted. In this case the Underrun Error Status bit is set in SR (SR.UNDES).

Figure 21-9 on page 387 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	0x00000000
0x14	Interrupt Enable Register	IER	Write-only	0x00000000
0x18	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x30	Chip Select Register 0	CSR0	Read/Write	0x00000000
0x34	Chip Select Register 1	CSR1	Read/Write	0x00000000
0x38	Chip Select Register 2	CSR2	Read/Write	0x00000000
0x3C	Chip Select Register 3	CSR3	Read/Write	0x00000000
0x E4	Write Protection Control Register	WPCR	Read/Write	0X00000000
0xE8	Write Protection Status Register	WPSR	Read-only	0x00000000
0xF8	Features Register	FEATURES	Read-only	- (1)
0xFC	Version Register	VERSION	Read-only	- (1)

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

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	-	MODFDIS	-	PCSDEC	PS	MSTR

- **DLYBCS: Delay Between Chip Selects**
  This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.
  If DLYBCS is less than or equal to six, six CLK_SPI periods will be inserted by default.
  Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{CLKSPI}$$

- **PCS: Peripheral Chip Select**
  This field is only used if Fixed Peripheral Select is active (PS = 0).
  If PCSDEC = 0:
  PCS = xxx0NPCS[3:0] = 1110
  PCS = xx01NPCS[3:0] = 1101
  PCS = x011NPCS[3:0] = 1011
  PCS = 0111NPCS[3:0] = 0111
  PCS = 1111forbidden (no peripheral is selected)
  (x = don't care)
  If PCSDEC = 1:
  NPCS[3:0] output signals = PCS.
- **LLB: Local Loopback Enable**
  1: Local loopback path enabled. LLB controls the local loopback on the data serializer for testing in master mode only (MISO is internally connected on MOSI).
  0: Local loopback path disabled.
- **RXFIFOEN: FIFO in Reception Enable**
  1: The FIFO is used in reception (four characters can be stored in the SPI).

0: The FIFO is not used in reception (only one character can be stored in the SPI).

- **MODFDIS: Mode Fault Detection**
  1: Mode fault detection is disabled. If the I/O controller does not have open-drain capability, mode fault detection **must** be disabled for proper operation of the SPI.
  0: Mode fault detection is enabled.
- **PCSDEC: Chip Select Decode**
  0: The chip selects are directly connected to a peripheral device.
  1: The four chip select lines are connected to a 4- to 16-bit decoder.
  When PCSDEC equals one, up to 15 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder. The CSRn registers define the characteristics of the 15 chip selects according to the following rules:
  CSR0 defines peripheral chip select signals 0 to 3.
  CSR1 defines peripheral chip select signals 4 to 7.
  CSR2 defines peripheral chip select signals 8 to 11.
  CSR3 defines peripheral chip select signals 12 to 14.
- **PS: Peripheral Select**
  1: Variable Peripheral Select.
  0: Fixed Peripheral Select.
- **MSTR: Master/Slave Mode**
  1: SPI is in master mode.
  0: SPI is in slave mode.

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**
  - 0: Spurious write protection is not implemented.
  - 1: Spurious write protection is implemented.
- **FIFORIMPL: FIFO in Reception Implemented**
  - 0: FIFO in reception is not implemented.
  - 1: FIFO in reception is implemented.
- **BRPBHSB: Bridge Type is PB to HSB**
  - 0: Bridge type is not PB to HSB.
  - 1: Bridge type is PB to HSB.
- **CSNAATIMPL: CSNAAT Features Implemented**
  - 0: CSNAAT (Chip select not active after transfer) features are not implemented.
  - 1: CSNAAT features are implemented.
- **EXTDEC: External Decoder True**
  - 0: External decoder capability is not implemented.
  - 1: External decoder capability is implemented.
- **LENNCONF: Character Length if not Configurable**
  - If the character length is not configurable, this field specifies the fixed character length.
- **LENCONF: Character Length Configurable**
  - 0: The character length is not configurable.
  - 1: The character length is configurable.
- **PHZNCONF: Phase is Zero if Phase not Configurable**
  - 0: If phase is not configurable, phase is non-zero.
  - 1: If phase is not configurable, phase is zero.
- **PHCONF: Phase Configurable**
  - 0: Phase is not configurable.
  - 1: Phase is configurable.

- **PPNCONF: Polarity Positive if Polarity not Configurable**
  0: If polarity is not configurable, polarity is negative.
  1: If polarity is not configurable, polarity is positive.
- **PCONF: Polarity Configurable**
  0: Polarity is not configurable.
  1: Polarity is configurable.
- **NCS: Number of Chip Selects**
  This field indicates the number of chip selects implemented.

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
-	-	-	-	MFN			

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

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

- **MFN**

  Reserved. No functionality associated.

- **VERSION**

  Version number of the module. No functionality associated.

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. Inter-IC Sound Controller (IISC)

Rev: 1.0.0.0

22.1 Features

- **Compliant with Inter-IC Sound (I^2S) bus specification**
- **Master, slave, and controller modes:**
  - **Slave: data received/transmitted**
  - **Master: data received/transmitted and clocks generated**
  - **Controller: clocks generated**
- **Individual enable and disable of receiver, transmitter, and clocks**
- **Configurable clock generator common to receiver and transmitter:**
  - **Suitable for a wide range of sample frequencies (fs), including 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz, 96 kHz, and 192 kHz**
  - **16 fs to 1024 fs Master Clock generated for external oversampling ADCs**
- **Several data formats supported:**
  - **32-, 24-, 20-, 18-, 16-, and 8-bit mono or stereo format**
  - **16- and 8-bit compact stereo format, with left and right samples packed in the same word to reduce data transfers**
- **DMA interfaces for receiver and transmitter to reduce processor overhead:**
  - **Either one DMA channel for both audio channels, or**
  - **One DMA channel per audio channel**
- **Smart holding registers management to avoid audio channels mix after overrun or underrun**

22.2 Overview

The Inter-IC Sound Controller (IISC) provides a 5-wire, bidirectional, synchronous, digital audio link with external audio devices: ISDI, ISDO, IWS, ISCK, and IMCK pins.

This controller is compliant with the Inter-IC Sound (I^2S) bus specification.

The IISC consists of a Receiver, a Transmitter, and a common Clock Generator, that can be enabled separately, to provide Master, Slave, or Controller modes with Receiver, Transmitter, or both active.

Peripheral DMA channels, separate for the Receiver and for the Transmitter, allow a continuous high bitrate data transfer without processor intervention to the following:

- Audio CODECs in Master, Slave, or Controller mode
- Stereo DAC or ADC through dedicated I^2S serial interface

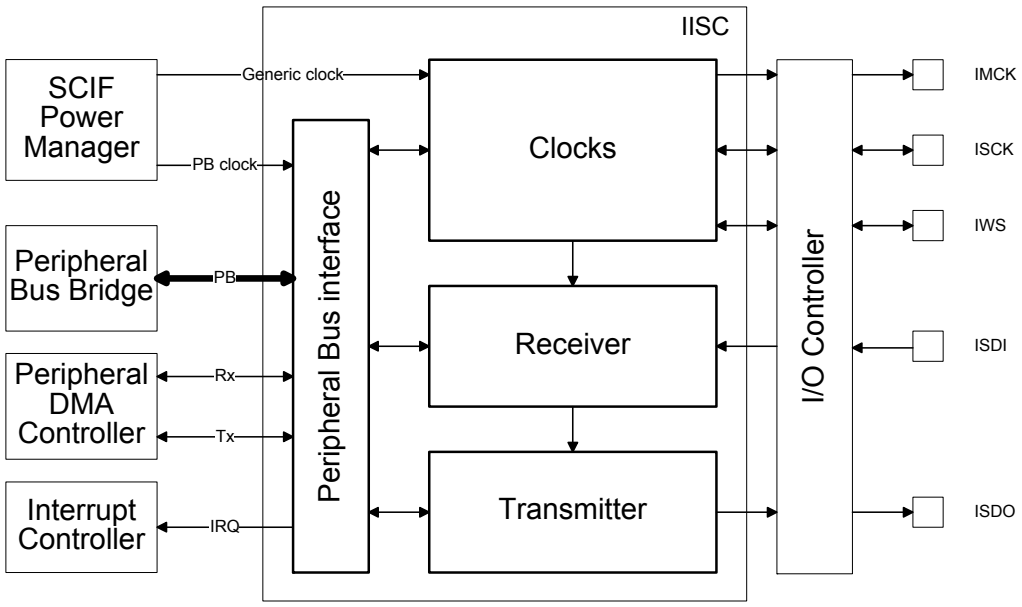
The IISC can use either a single DMA channel for both audio channels or one DMA channel per audio channel.

The 8- and 16-bit compact stereo format allows reducing the required DMA bandwidth by transferring the left and right samples within the same data word.

In Master Mode, the IISC allows outputting a 16 fs to 1024 fs Master Clock, in order to provide an oversampling clock to an external audio codec or digital signal processor (DSP).

22.3 Block Diagram

Figure 22-1. IISC Block Diagram



22.4 I/O Lines Description

Table 22-1. I/O Lines Description

Pin Name	Pin Description	Type
IMCK	Master Clock	Output
ISCK	Serial Clock	Input/Output
IWS	I^2S Word Select	Input/Output
ISDI	Serial Data Input	Input
ISDO	Serial Data Output	Output

22.5 Product Dependencies

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

22.5.1 I/O lines

The IISC pins may be multiplexed with I/O Controller lines. The user must first program the I/O Controller to assign the desired IISC pins to their peripheral function. If the IISC I/O lines are not used by the application, they can be used for other purposes by the I/O Controller. It is required to enable only the IISC inputs and outputs actually in use.

22.5.2 Power Management

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

22.5.3 Clocks

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

One of the generic clocks is connected to the IISC. The generic clock (GCLK_IISC) can be set to a wide range of frequencies and clock sources. The GCLK_IISC must be enabled and configured before use. Refer to the module configuration section for details on the GCLK_IISC used for the IISC. The frequency for this clock has to be set as described in Table.

22.5.4 DMA

The IISC DMA handshake interfaces are connected to the Peripheral DMA Controller. Using the IISC DMA functionality requires the Peripheral DMA Controller to be programmed first.

22.5.5 Interrupts

The IISC interrupt line is connected to the Interrupt Controller. Using the IISC interrupt requires the Interrupt Controller to be programmed first.

22.5.6 Debug Operation

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

22.6 Functional Description

22.6.1 Initialization

The IISC features a Receiver, a Transmitter, and, for Master and Controller modes, a Clock Generator. Receiver and Transmitter share the same Serial Clock and Word Select.

Before enabling the IISC, the chosen configuration must be written to the Mode Register (MR). The IMCKMODE, MODE, and DATALENGTH fields in the MR register must be written. If the IMCKMODE field is written as one, then the IMCKFS field should be written with the chosen ratio, as described in Section 22.6.5 ”Serial Clock and Word Select Generation” on page 420.

Once the Mode Register has been written, the IISC Clock Generator, Receiver, and Transmitter can be enabled by writing a one to the CKEN, RXEN, and TXEN bits in the Control Register (CR). The Clock Generator can be enabled alone, in Controller Mode, to output clocks to the IMCK, ISCK, and IWS pins. The Clock Generator must also be enabled if the Receiver or the Transmitter is enabled.

The Clock Generator, Receiver, and Transmitter can be disabled independently by writing a one to CR.CXDIS, CR.RXDIS and/or CR.TXDIS respectively. Once requested to stop, they will only stop when the transmission of the pending frame transmission will be completed.

22.6.2 Basic Operation

The Receiver can be operated by reading the Receiver Holding Register (RHR), whenever the Receive Ready (RXRDY) bit in the Status Register (SR) is set. Successive values read from RHR will correspond to the samples from the left and right audio channels for the successive frames.

The Transmitter can be operated by writing to the Transmitter Holding Register (RHR), whenever the Transmit Ready (TXRDY) bit in the Status Register (SR) is set. Successive values written to THR should correspond to the samples from the left and right audio channels for the successive frames.

The Receive Ready and Transmit Ready bits can be polled by reading the Status Register.

The IISC processor load can be reduced by enabling interrupt-driven operation. The RXRDY and/or TXRDY interrupt requests can be enabled by writing a one to the corresponding bit in the Interrupt Enable Register (IER). The interrupt service routine associated to the IISC interrupt request will then be executed whenever the Receive Ready or the Transmit Ready status bit is set.

22.6.3 Master, Controller, and Slave Modes

In Master and Controller modes, the IISC provides the Master Clock, the Serial Clock and the Word Select. IMCK, ISCK, and IWS pins are outputs.

In Controller mode, the IISC Receiver and Transmitter are disabled. Only the clocks are enabled and used by an external receiver and/or transmitter.

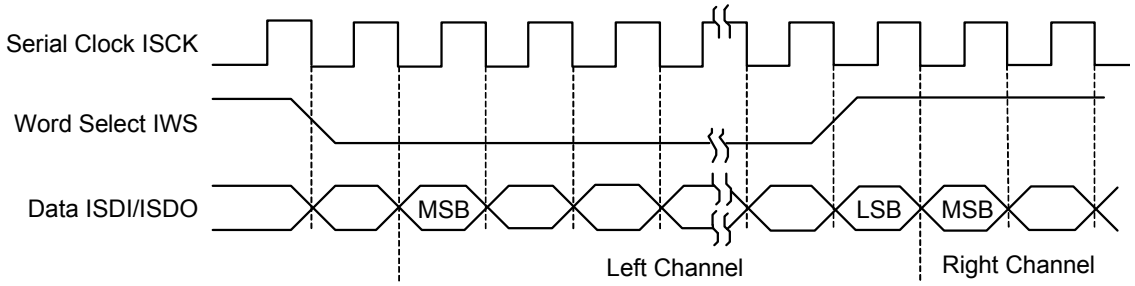
In Slave mode, the IISC receives the Serial Clock and the Word Select from an external master. ISCK and IWS pins are inputs.

The mode is selected by writing the MODE field of the Mode Register (MR). Since the MODE field changes the direction of the IWS and ISCK pins, the Mode Register should only be written when the IISC is stopped, in order to avoid unwanted glitches on the IWS and ISCK pins.

22.6.4 I^2S Reception and Transmission Sequence

As specified in the I^2S protocol, data bits are left-adjusted in the Word Select time slot, with the MSB transmitted first, starting one clock period after the transition on the Word Select line.

Figure 22-2. I^2S Reception and Transmission Sequence



Data bits are sent on the falling edge of the Serial Clock and sampled on the rising edge of the Serial Clock. The Word Select line indicates the channel in transmission, a low level for the left channel and a high level for the right channel.

The length of transmitted words can be chosen among 8, 16, 18, 20, 24, and 32 bits by writing the MR.DATALENGTH field.

If the time slot allows for more data bits than written in the MR.DATALENGTH field, zeroes are appended to the transmitted data word or extra received bits are discarded. If the time slot allows for less data bits than written, the extra bits to be transmitted are not sent or the missing bits are set to zero in the received data word.

22.6.5 Serial Clock and Word Select Generation

The generation of clocks in the IISC is described in Figure 22-3 on page 421.

In Slave mode, the Serial Clock and Word Select Clock are driven by an external master. ISCK and IWS pins are inputs and no generic clock is required by the IISC.

In Master mode, the user can configure the Master Clock, Serial Clock, and Word Select Clock through the Mode Register (MR). IMCK, ISCK, and IWS pins are outputs and a generic clock is used to derive the IISC clocks.

Audio codecs connected to the IISC pins may require a Master Clock signal with a frequency multiple of the audio sample frequency (fs), such as 256fs. When the IISC is in Master mode, writing a one to MR.IMCKMODE will output GCLK_IISC as Master Clock to the IMCK pin, and will divide GCLK_IISC to create the internal bit clock, output on the ISCK pin. The clock division factor is defined by writing to MR.IMCKFS and MR.DATALENGTH, as described ”IMCKFS: Master Clock to fs Ratio” on page 427.

The Master Clock (IMCK) frequency is 16*(IMCKFS+1) times the sample frequency (fs), i.e. IWS frequency. The Serial Clock (ISCK) frequency is 2*Slot Length times the sample frequency (fs), where Slot Length is defined in Table 22-2 on page 420.

Table 22-2. Slot Length

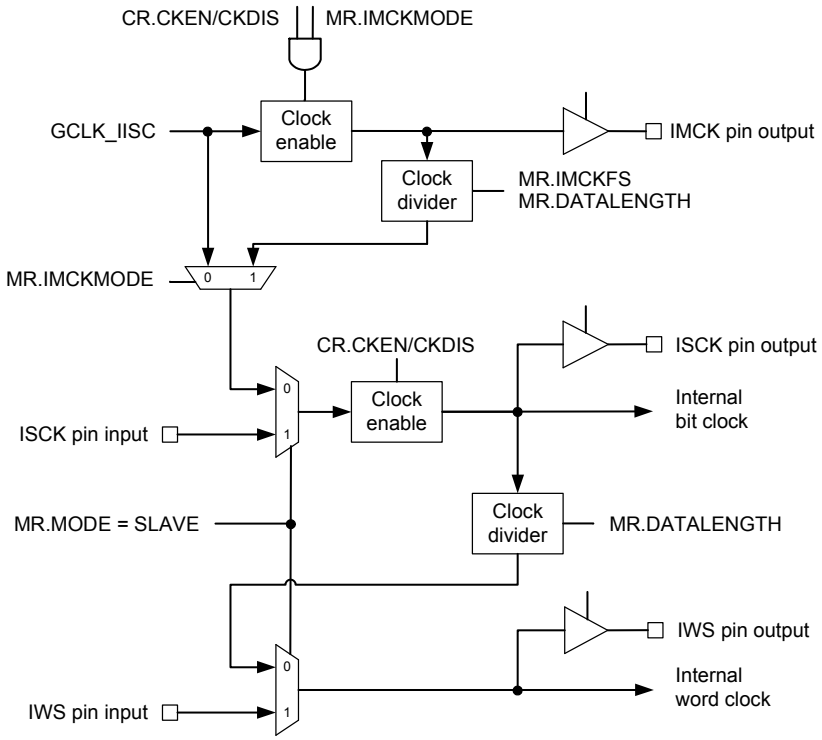
MR.DATALENGTH	Word Length	Slot Length
0	32 bits	32
1	24 bits	32 if MR.IWS24 is zero 24 if MR.IWS24 is one
2	20 bits	
3	18 bits	
4	16 bits	16
5	16 bits compact stereo	
6	8 bits	8
7	8 bits compact stereo	

Warning: MR.IMCKMODE should only be written as one if the Master Clock frequency is strictly higher than the Serial Clock.

If a Master Clock output is not required, the GCLK_IISC generic clock is used as ISCK, by writing a zero to MR.IMCKMODE. Alternatively, if the frequency of the generic clock used is a multiple of the required ISCK frequency, the IMCK to ISCK divider can be used with the ratio defined by writing the MR.IMCKFS field.

The IWS pin is used as Word Select as described in Section 22.6.4.

Figure 22-3. IISC Clocks Generation



22.6.6 Mono

When the Transmit Mono (TXMONO) in the Mode Register is set, data written to the left channel is duplicated to the right output channel.

When the Receive Mono (RXMONO) in the Mode Register is set, data received from the left channel is duplicated to the right channel.

22.6.7 Holding Registers

The IISC user interface includes a Receive Holding Register (RHR) and a Transmit Holding Register (THR). RHR and THR are used to access audio samples for both audio channels.

When a new data word is available in the RHR register, the Receive Ready bit (RXRDY) in the Status Register (SR) is set. Reading the RHR register will clear this bit.

A receive overrun condition occurs if a new data word becomes available before the previous data word has been read from the RHR register. Then, the Receive Overrun bit in the Status Register will be set and bit i of the RXORCH field in the Status Register is set, where i is the current receive channel number.

When the THR register is empty, the Transmit Ready bit (TXRDY) in the Status Register (SR) is set. Writing into the THR register will clear this bit.

A transmit underrun condition occurs if a new data word needs to be transmitted before it has been written to the THR register. Then, the Transmit Underrun bit in the Status Register will be set and bit i of the TXORCH field in the Status Register is set, where i is the current transmit channel number. If the TXSAME bit in the Mode Register is zero, then a zero data word is transmitted in case of underrun. If MR.TXSAME is one, then the previous data word for the current transmit channel number is transmitted.

Data words are right-justified in the RHR and THR registers. For 16-bit compact stereo, the left sample uses bits 15 through 0 and the right sample uses bits 31 through 16 of the same data word. For 8-bit compact stereo, the left sample uses bits 7 through 0 and the right sample uses bits 15 through 8 of the same data word.

22.6.8 DMA Operation

The Receiver and the Transmitter can each be connected either to one single Peripheral DMA channel or to one Peripheral DMA channel per data channel. This is selected by writing to the MR.RXDMA and MR.TXDMA bits. If a single Peripheral DMA channel is selected, all data samples use IISC Receiver or Transmitter DMA channel 0.

The Peripheral DMA reads from the RHR register and writes to the RHR register for both audio channels, successively.

The Peripheral DMA transfers may use 32-bit word, 16-bit halfword, or 8-bit byte according to the value of the MR.DATALENGTH field.

22.6.9 Loop-back Mode

For debugging purposes, the IISC can be configured to loop back the Transmitter to the Receiver. Writing a one to the MR.LOOP bit will internally connect ISDO to ISDI, so that the transmitted data is also received. Writing a zero to MR.LOOP will restore the normal behavior with independent Receiver and Transmitter. As for other changes to the Receiver or Transmitter configuration, the IISC Receiver and Transmitter must be disabled before writing to the MR register to update MR.LOOP.

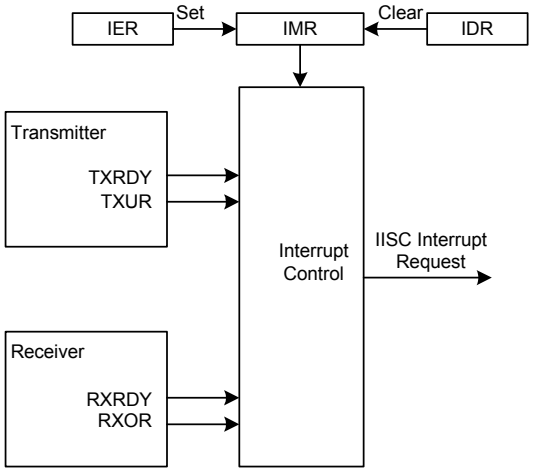
22.6.10 Interrupts

An IISC interrupt request can be triggered whenever one or several of the following bits are set in the Status Register (SR): Receive Ready (RXRDY), Receive Overrun (RXOR), Transmit Ready (TXRDY), or Transmit Underrun (TXOR).

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

For debugging purposes, interrupt requests can be simulated by writing a one to the corresponding bit in the Status Set Register (SSR).

Figure 22-4. Interrupt Block Diagram



22.7 IISC Application Examples

The IISC can support several serial communication modes used in audio or high-speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the IISC are not listed here.

Figure 22-5. Audio Application Block Diagram

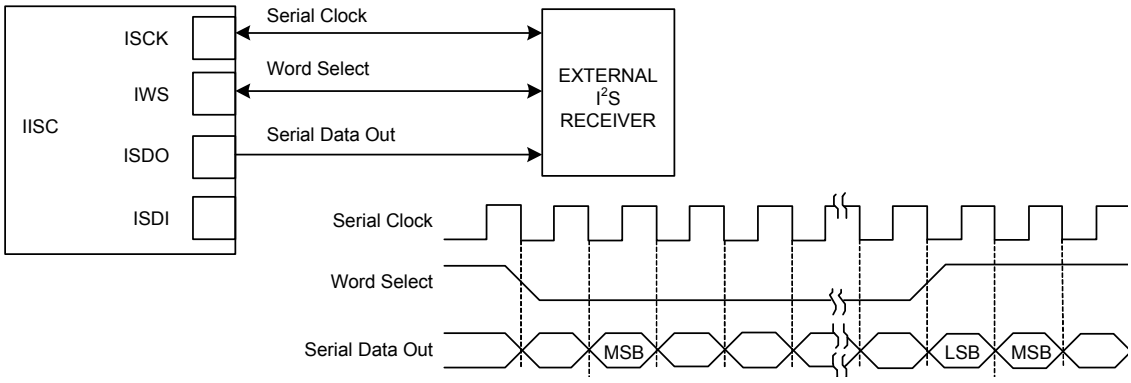


Figure 22-6. Codec Application Block Diagram

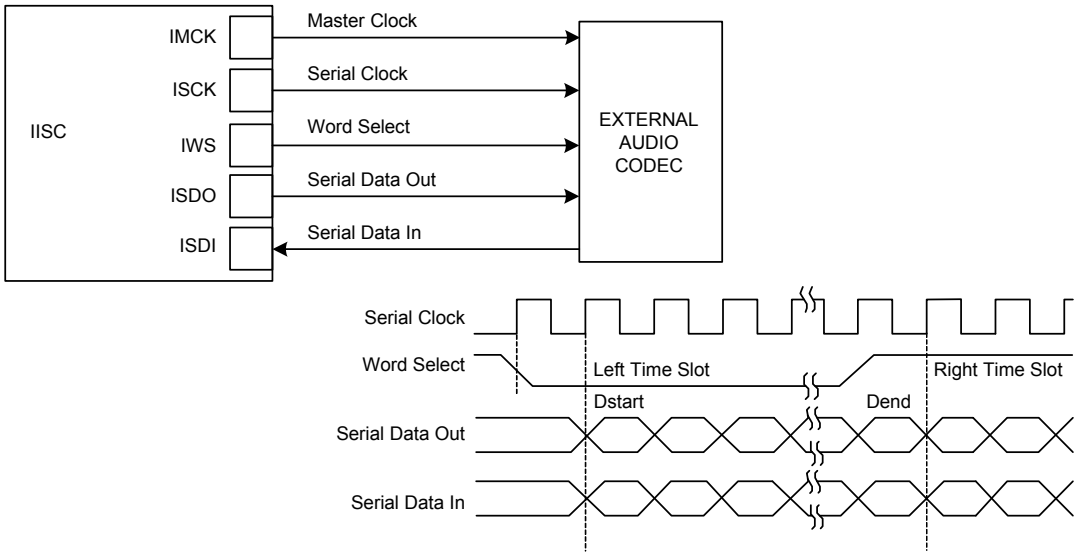
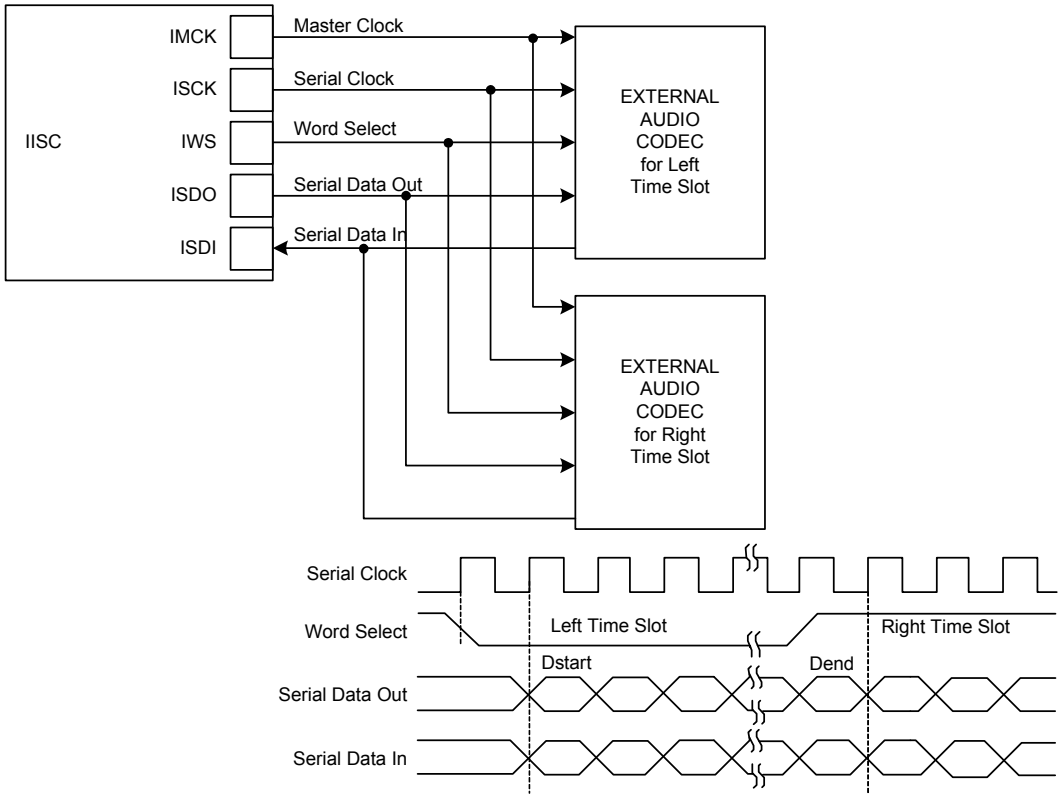


Figure 22-7. Time Slot Application Block Diagram



22.8 User Interface

Table 22-3. IISC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	0x00000000
0x04	Mode Register	MR	Read/Write	0x00000000
0x08	Status Register	SR	Read-only	0x00000000
0x0C	Status Clear Register	SCR	Write-only	0x00000000
0x10	Status Set Register	SSR	Write-only	0x00000000
0x14	Interrupt Enable Register	IER	Write-only	0x00000000
0x18	Interrupt Disable Register	IDR	Write-only	0x00000000
0x1C	Interrupt Mask Register	IMR	Read-only	0x00000000
0x20	Receiver Holding Register	RHR	Read-only	0x00000000
0x24	Transmitter Holding Register	THR	Write-only	0x00000000
0x28	Version Register	VERSION	Read-only	-(1)
0x2C	Parameter Register	PARAMETER	Read-only	-(1)

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

22.8.1 Control Register

Name: CR

Access Type: Write-only

Offset: 0x00

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
SWRST	-	TXDIS	TXEN	CKDIS	CKEN	RXDIS	RXEN

The Control Register should only be written to enable the IISC after the chosen configuration has been written to the Mode Register, in order to avoid unwanted glitches on the IWS, ISCK, and ISDO outputs. The proper sequence is to write the MR register, then write the CR register to enable the IISC, or to disable the IISC before writing a new value into MR.

- **SWRST: Software Reset**
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all the registers in the module. The module will be disabled after the reset.
  This bit always reads as zero.
- **TXDIS: Transmitter Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the IISC Transmitter. SR.TXEN will be cleared when the Transmitter is effectively stopped.
- **TXEN: Transmitter Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the IISC Transmitter, if TXDIS is not one. SR.TXEN will be set when the Transmitter is effectively started.
- **CKDIS: Clocks Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the IISC clocks generation.
- **CKEN: Clocks Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the IISC clocks generation, if CKDIS is not one.
- **RXDIS: Receiver Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the IISC Receiver. SR.TXEN will be cleared when the Transmitter is effectively stopped.
- **RXEN: Receiver Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the IISC Receiver, if RXDIS is not one. SR.RXEN will be set when the Receiver is effectively started.

22.8.2 Mode Register

Name: MR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
IWS24	IMCKMODE	IMCKFS					

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

15	14	13	12	11	10	9	8
-	TXSAME	TXDMA	TXMONO		RXLOOP	RXDMA	RXMONO

7	6	5	4	3	2	1	0
-	-	-	DATALENGTH			-	MODE

The Mode Register should only be written when the IISC is stopped, in order to avoid unwanted glitches on the IWS, ISCK, and ISDO outputs. The proper sequence is to write the MR register, then write the CR register to enable the IISC, or to disable the IISC before writing a new value into MR.

- **IWS24: IWS TDM Slot Width**
  0: IWS slot is 32-bit wide for DATALENGTH=18/20/24-bit
  1: IWS slot is 24-bit wide for DATALENGTH=18/20/24-bit
  Refer to Table 22-2, “Slot Length,” on page 420.
- **IMCKMODE: Master Clock Mode**
  0: No Master Clock generated (generic clock is used as ISCK output)
  1: Master Clock generated (generic clock is used as IMCK output)
  Warning: if IMCK frequency is the same as ISCK, IMCKMODE should not be written as one. Refer to Section 22.6.5 ”Serial Clock and Word Select Generation” on page 420 and Table 22-2, “Slot Length,” on page 420.
- **IMCKFS: Master Clock to fs Ratio**
  Master Clock frequency is 16*(IMCKFS+1) times the sample rate, i.e. IWS frequency:

Table 22-4. Master Clock to Sample Frequency (fs) Ratio

fs Ratio	IMCKFS
16 fs	0
32 fs	1
48fs	2
64 fs	3
96fs	5
128 fs	7
192fs	11
256 fs	15

Table 22-4. Master Clock to Sample Frequency (fs) Ratio

fs Ratio	IMCKFS
384 fs	23
512 fs	31
768 fs	47
1024 fs	63

- **TXSAME: Transmit Data when Underrun**
  0: Zero sample transmitted when underrun
  1: Previous sample transmitted when underrun
- **TXDMA: Single or multiple DMA Channels for Transmitter**
  0: Transmitter uses a single DMA channel for both audio channels
  1: Transmitter uses one DMA channel per audio channel
- **TXMONO: Transmit Mono**
  0: Stereo
  1: Mono, with left audio samples duplicated to right audio channel by the IISC
- **RXLOOP: Loop-back Test Mode**
  0: Normal mode
  1: ISDO output of IISC is internally connected to ISDI input
- **RXMONO: Receive Mono**
  0: Stereo
  1: Mono, with left audio samples duplicated to right audio channel by the IISC
- **RXDMA: Single or multiple DMA Channels for Receiver**
  0: Receiver uses a single DMA channel for both audio channels
  1: Receiver uses one DMA channel per audio channel-
- **DATALENGTH: Data Word Length**

Table 22-5. Data Word Length

DATALENGTH	Word Length	Comments
0	32 bits	
1	24 bits	
2	20 bits	
3	18 bits	
4	16 bits	
5	16 bits compact stereo	Left sample in bits 15 through 0 and right sample in bits 31 through 16 of the same word
6	8 bits	
7	8 bits compact stereo	Left sample in bits 7 through 0 and right sample in bits 15 through 8 of the same word

- **MODE: Mode**

Table 22-6. Mode

MODE		Comments
0	SLAVE	ISCK and IWS pin inputs used as Bit Clock and Word Select/Frame Sync.
1	MASTER	Bit Clock and Word Select/Frame Sync generated by IISC from GCLK_IISC and output to ISCK and IWS pins. GCLK_IISC is output as Master Clock on IMCK if MR.IMCKMODE is one.

22.8.3 Status Register

Name: SR

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
-	-	TXURCH		-	-	-	-

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

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	TXEN	-	RXOR	RXRDY	RXEN

- **TXURCH: Transmit Underrun Channel**
  This field is cleared when SCR.TXUR is written to one
  Bit i of this field is set when a transmit underrun error occurred in channel i (i=0 for first channel of the frame)
- **RXORCH: Receive Overrun Channel**
  This field is cleared when SCR.RXOR is written to one
  Bit i of this field is set when a receive overrun error occurred in channel i (i=0 for first channel of the frame)
- **TXUR: Transmit Underrun**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when an underrun error occurs on the THR register or when the corresponding bit in SSR is written to one
- **TXRDY: Transmit Ready**
  This bit is cleared when data is written to THR
  This bit is set when the THR register is empty and can be written with new data to be transmitted
- **TXEN: Transmitter Enabled**
  This bit is cleared when the Transmitter is effectively disabled, following a CR.TXDIS or CR.SWRST request
  This bit is set when the Transmitter is effectively enabled, following a CR.TXEN request
- **RXOR: Receive Overrun**
  This bit is cleared when the corresponding bit in SCR is written to one
  This bit is set when an overrun error occurs on the RHR register or when the corresponding bit in SSR is written to one
- **RXRDY: Receive Ready**
  This bit is cleared when the RHR register is read
  This bit is set when received data is present in the RHR register
- **RXEN: Receiver Enabled**
  This bit is cleared when the Receiver is effectively disabled, following a CR.RXDIS or CR.SWRST request
  This bit is set when the Receiver is effectively enabled, following a CR.RXEN request

22.8.4 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x0C

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	TXUR	-	-	-	RXOR	-	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

22.8.5 Status Set Register

Name: SSR

Access Type: Write-only

Offset: 0x10

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	TXUR	-	-	-	RXOR	-	-

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

22.8.6 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x14

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	-	-	RXOR	RXRDY	-

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

22.8.7 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x18

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	-	-	RXOR	RXRDY	-

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

22.8.8 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x1C

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	-	-	RXOR	RXRDY	-

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

22.8.9 Receive Holding Register

Name: RHR

Access Type: Read-only

Offset: 0x20

Reset Value: 0x00000000

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

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

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

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

- **RHR: Received Word**
  This field is set by hardware to the last received data word. If MR.DATALENGTH specifies less than 32 bits, data shall be right-justified into the RHR field.

22.8.10 Transmit Holding Register

Name: THR

Access Type: Write-only

Offset: 0x24

Reset Value: 0x00000000

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

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

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

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

- **THR: Data Word to Be Transmitted**
  Next data word to be transmitted after the current word if TXRDY is not set. If MR.DATALENGTH specifies less than 32 bits, data shall be right-justified into the THR field.

22.8.11 Module Version

Name: VERSION

Access Type: Read-only

Offset: 0x28

Reset Value: -

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

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

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

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

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

22.8.12 Module Parameters

Name: PARAMETER

Access Type: Read-only

Offset: 0x2C

Reset Value: -

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

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

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

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

Reserved. No functionality associated.

22.9 Module configuration

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

Table 22-7. IISC Clocks

Clock Name	Description
CLK_IISC	Clock for the IISC bus interface
GCLK_IISC	IISC output clock source. The generic clock used for the IISC is GCLK5

Table 22-8. Register Reset Values

Register	Reset Value
VERSION	**0x00000100**
PARAMETER	**0x00010000**

23. Two-wire Master Interface (TWIM)

Rev.: 1.1.0.1

23.1 Features

- **Compatible with I²C standard**
  - **Multi-master support**
  - **Transfer speeds of 100 and 400 kbit/s**
  - **7- and 10-bit and General Call addressing**
- **Compatible with SMBus standard**
  - **Hardware Packet Error Checking (CRC) generation and verification with ACK control**
  - **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**

23.2 Overview

The Atmel Two-wire Master Interface (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 time clock (RTC), dot matrix/graphic LCD controller, and temperature sensor, to name a few. The 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 23-1 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I²C compatible device.

Table 23-1. Atmel TWIM Compatibility with I²C Standard

I²C Standard	Atmel TWIM
Standard-mode (100 kbit/s)	Supported
Fast-mode (400 kbit/s)	Supported
Fast-mode Plus (1 Mbit/s)	Supported
7- or 10-bits Slave Addressing	Supported
START BYTE(1)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Slope Control and Input Filtering (Fast mode)	Supported
Clock Stretching	Supported

Note: 1. START + b000000001 + Ack + Sr

Table 23-2 lists the compatibility level of the Atmel Two-wire Master Interface and a full SMBus compatible master.

Table 23-2. Atmel TWIM Compatibility with SMBus Standard

SMBus Standard	Atmel TWIM
Bus Timeouts	Supported
Address Resolution Protocol	Supported
Host Functionality	Supported
Packet Error Checking	Supported

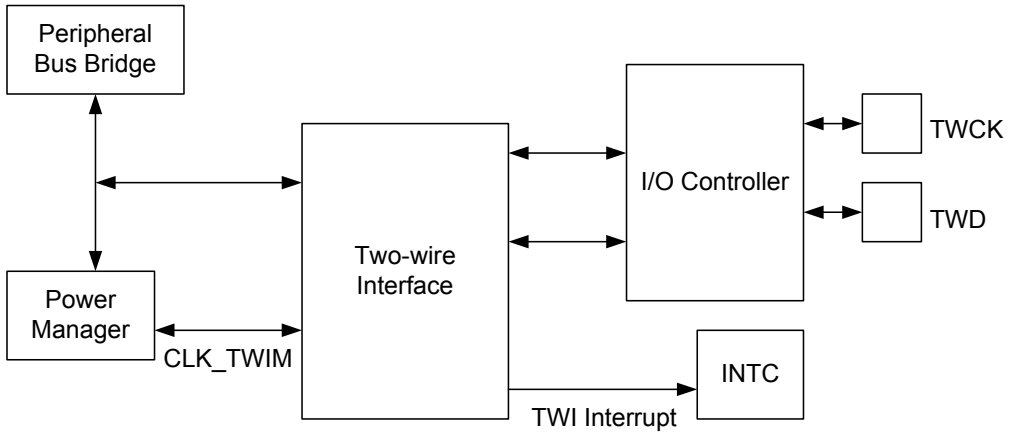
23.3 List of Abbreviations

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

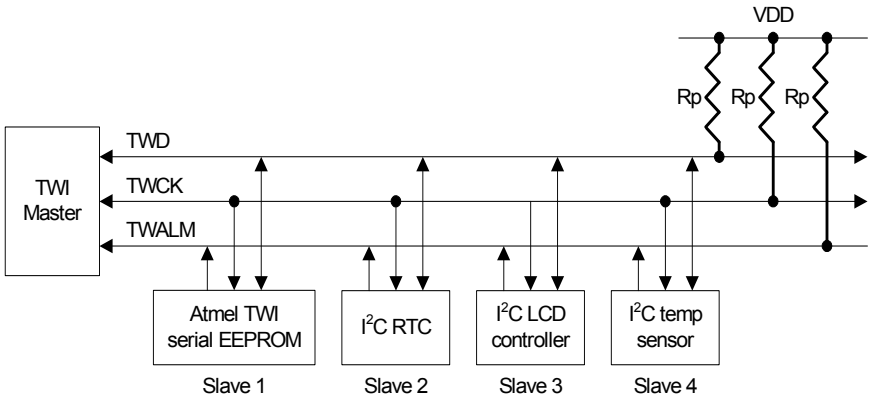
23.4 Block Diagram

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

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

TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 23-4 on page 444). 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.

The TWD and TWCK pins may be multiplexed with I/O Controller lines. To enable the TWIM, the user must perform the following steps:

- Program the I/O Controller to:
  - Dedicate TWD, TWCK as peripheral lines.
  - Define TWD, TWCK as open-drain.

23.7.2 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.

23.7.3 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.

23.7.4 DMA

The TWIM DMA handshake interface is connected to the Peripheral DMA Controller. Using the TWIM DMA functionality requires the Peripheral DMA Controller to be programmed after setting up the TWIM.

23.7.5 Interrupts

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

23.7.6 Debug Operation

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

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).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 23-4).

- 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

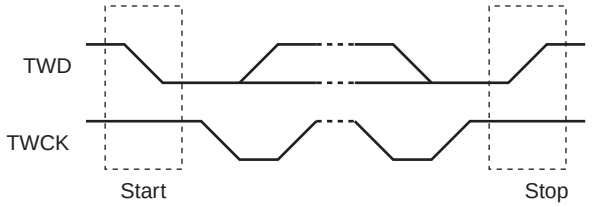
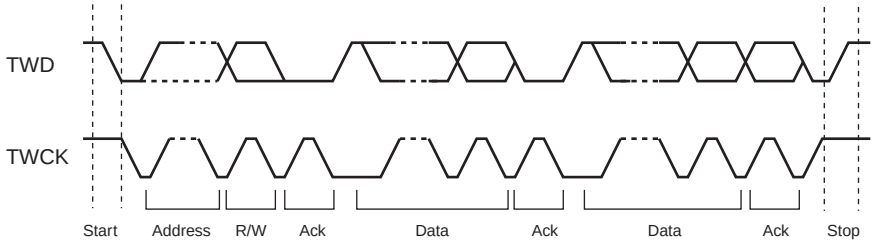


Figure 23-4. Transfer Format



23.8.2 Operation

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.

23.8.2.1 Clock Generation

The Clock Waveform Generator Register (CWGR) is used to control the waveform of the TWCK clock. CWGR must be written 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 Clock Prescaler field in CWGR (CWGR.EXP).

$$f_{PRESCALER} = \frac{f_{CLK\_TWIM}}{2^{(EXP+1)}}$$

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}.

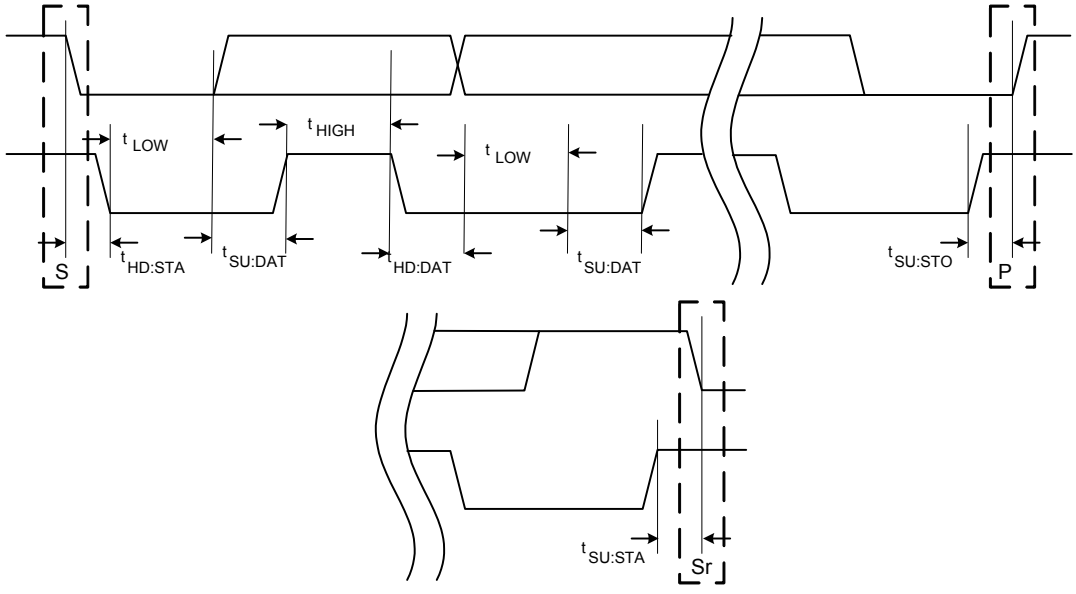
EXP: Specifies the clock prescaler setting.

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 23-5. Bus Timing Diagram



23.8.2.2 *Setting up and Performing a Transfer*

Operation of the TWIM is mainly controlled by the Control Register (CR) and the Command Register (CMDR). TWIM status is provided in the Status Register (SR). The following list presents the main steps in a typical communication:

1. Before any transfers can be performed, bus timings must be configured by writing to the Clock Waveform Generator Register (CWGR). If operating in SMBus mode, the SMBus Timing Register (SMBTR) register must also be configured.
2. If the Peripheral DMA Controller is to be used for the transfers, it must be set up.
3. CMDR or NCMDR must be written with a value describing the transfer to be performed.

The interrupt system can be set up to give interrupt requests on specific events or error conditions in the SR, for example when the transfer is complete or if arbitration is lost. The Interrupt Enable Register (IER) and Interrupt Disable Register (IDR) can be written to specify which bits in the SR will generate interrupt requests.

The SR.BUSFREE bit is set when activity is completed on the two-wire bus. The SR.CRDY bit is set when CMDR and/or NCMDR is ready to receive one or more commands.

The controller will refuse to start a new transfer while ANAK, DNAK, or ARBLST in the Status Register (SR) is one. 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 occurs, a STOP condition is sent automatically. The user will have to restart the transmission by clearing the error bits 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.

23.8.3 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 the Peripheral 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

Writing 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 (IER).

TXRDY is used as Transmit Ready for the Peripheral DMA Controller transmit channel.

The end of a command is marked when the TWIM sets the SR.CCOMP bit. See Figure 23-6 and Figure 23-7.

Figure 23-6. Master Write with One Data Byte

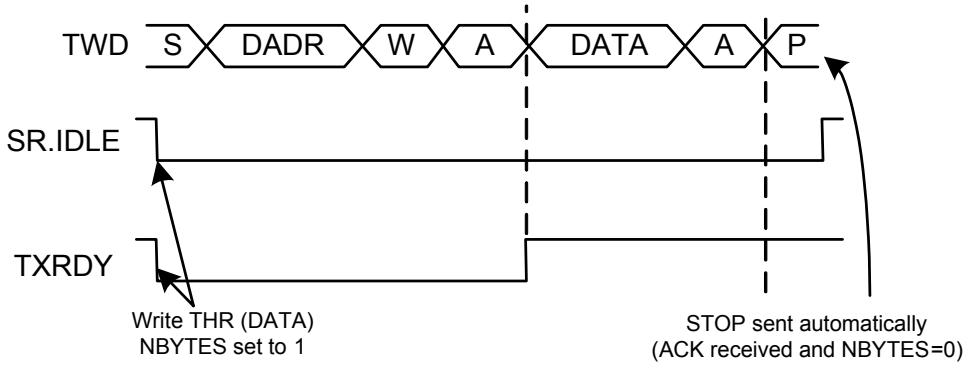
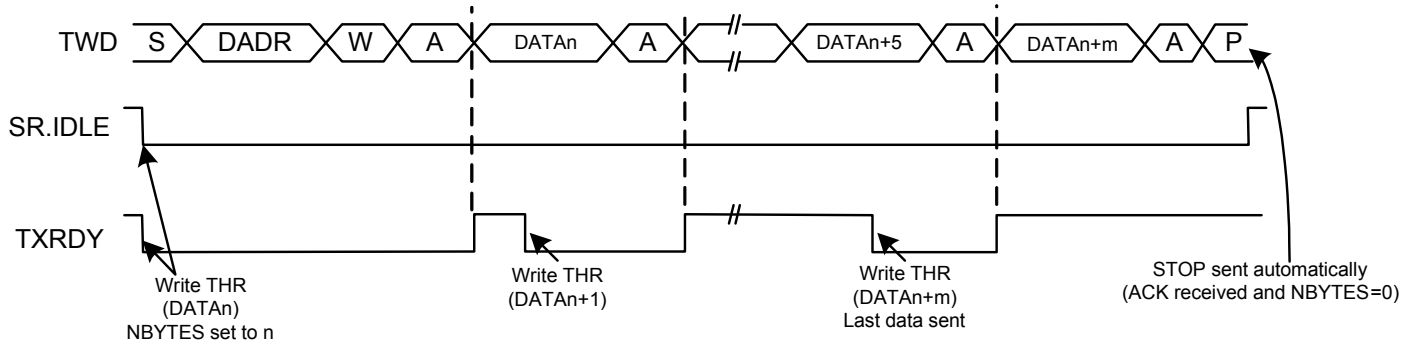


Figure 23-7. Master Write with Multiple Data Bytes



23.8.4 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 the Peripheral 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.

Writing 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 23-8. Master Read with One Data Byte

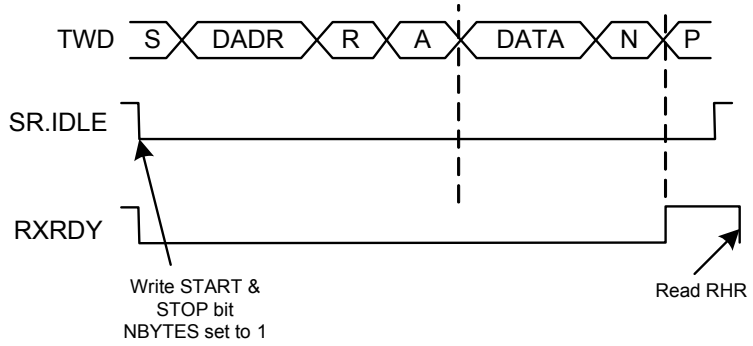
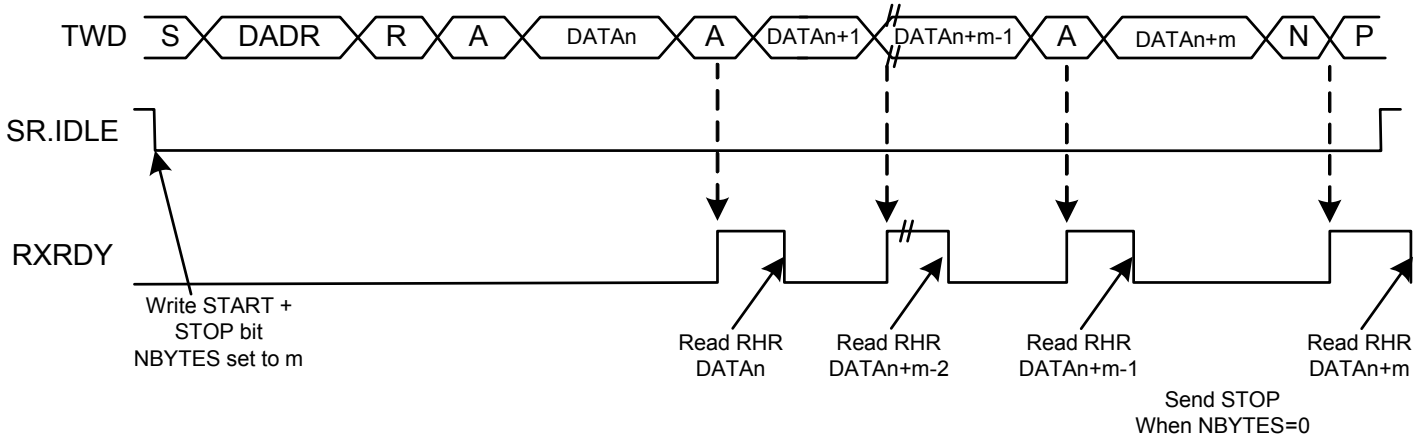


Figure 23-9. Master Read with Multiple Data Bytes



23.8.5 Using the Peripheral DMA Controller

The use of the Peripheral DMA Controller significantly reduces the CPU load. The user can set up ring buffers for the Peripheral DMA Controller, containing data to transmit or free buffer space to place received data.

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 TWIM (ADR, NBYTES, etc.).
3. Start the transfer by enabling the Peripheral DMA Controller to transmit.
4. Wait for the Peripheral DMA Controller end-of-transmit flag.
5. Disable the Peripheral DMA Controller.

23.8.5.2 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 enabling the Peripheral DMA Controller to receive.
4. Wait for the Peripheral DMA Controller end-of-receive flag.
5. Disable the Peripheral DMA Controller.

23.8.6 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 23-11.

If the user starts a transfer and if the bus is busy, the TWIM automatically waits for a STOP condition on the bus before initiating the transfer (see Figure 23-10).

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

Figure 23-10. User Sends Data While the Bus is Busy

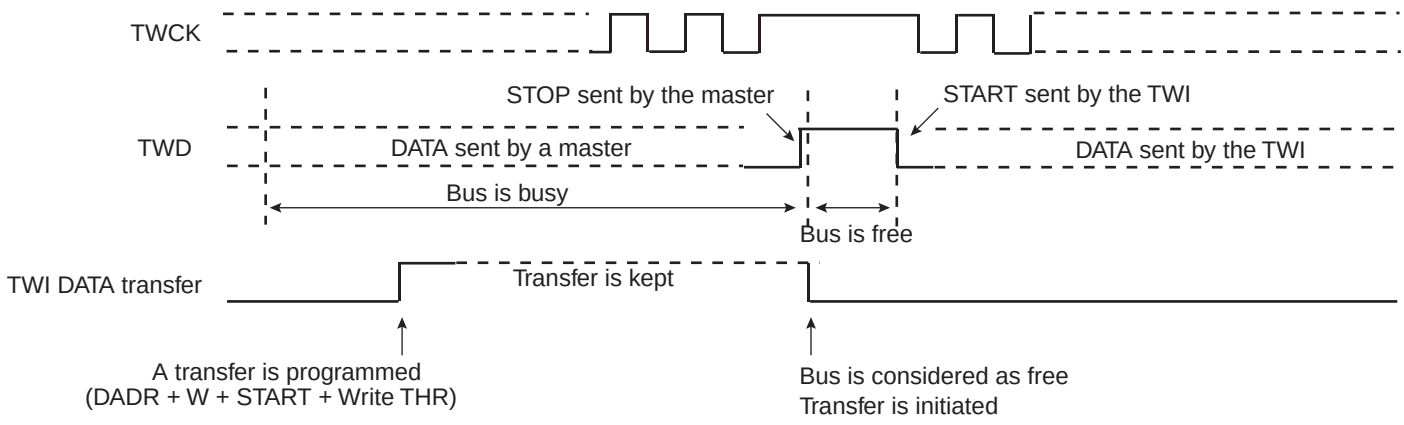
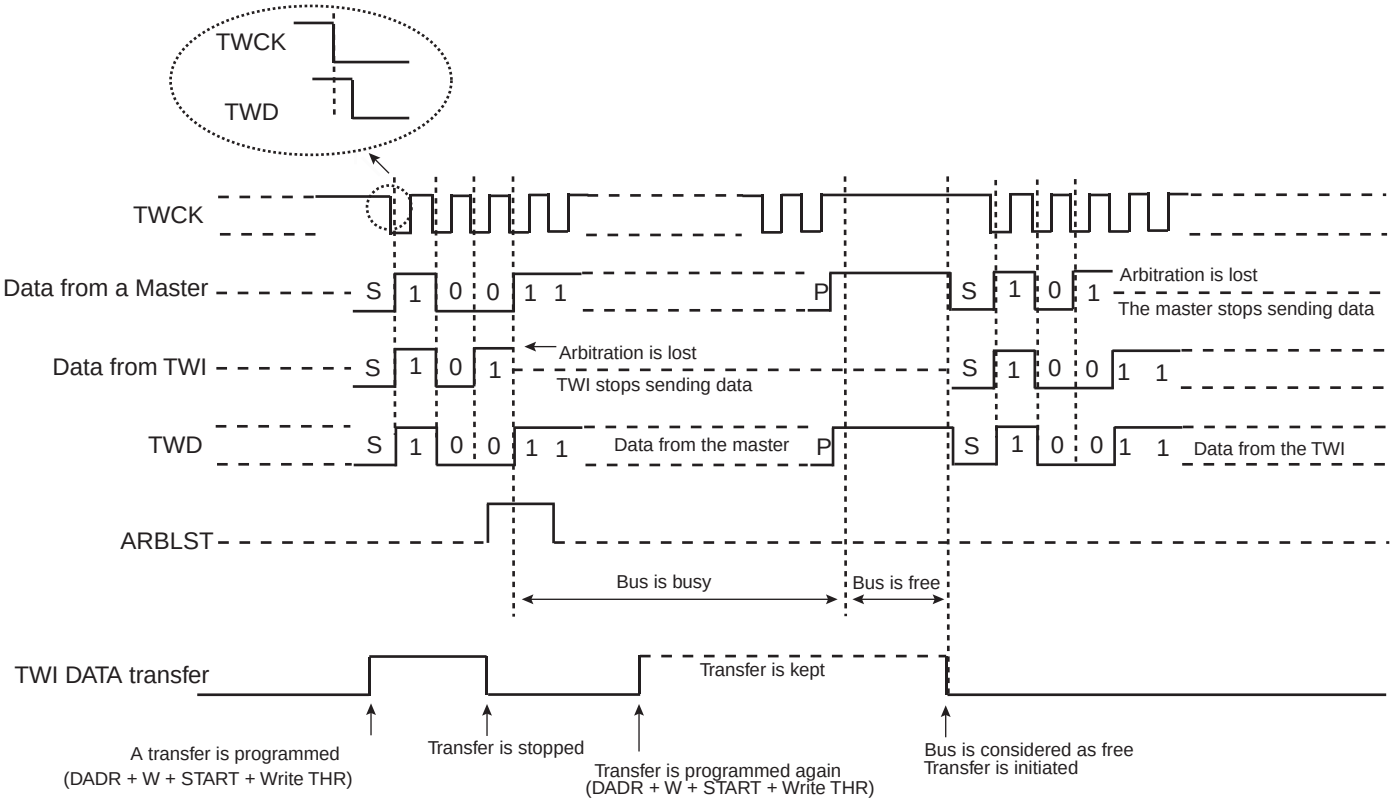


Figure 23-11. Arbitration Cases



23.8.7 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.

Writing NCMDR with START=1 when the previous transfer was written 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 write 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 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 Section 23.8.5

23.8.7.1 *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. Write CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=0.
2. Write 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.

23.8.7.2 *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. Write CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=1.
2. Write 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.

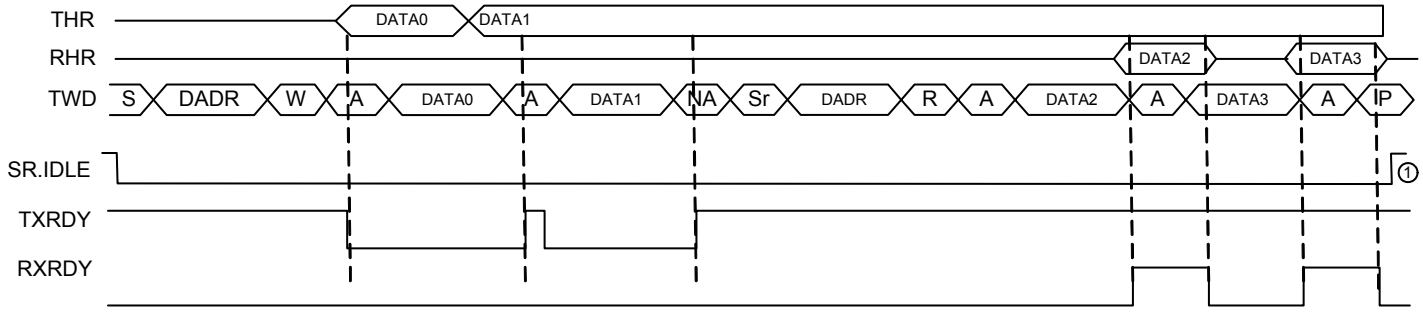
If combining several transfers, without any STOP or REPEATED START between them, remember to write a one to the ACKLAST bit in CMDR to keep from ending each of the partial transfers with a NACK.

23.8.7.3 *Write Followed by Read*

Consider the following transfer:

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

Figure 23-12. Combining a Write and Read Transfer



To generate this transfer:

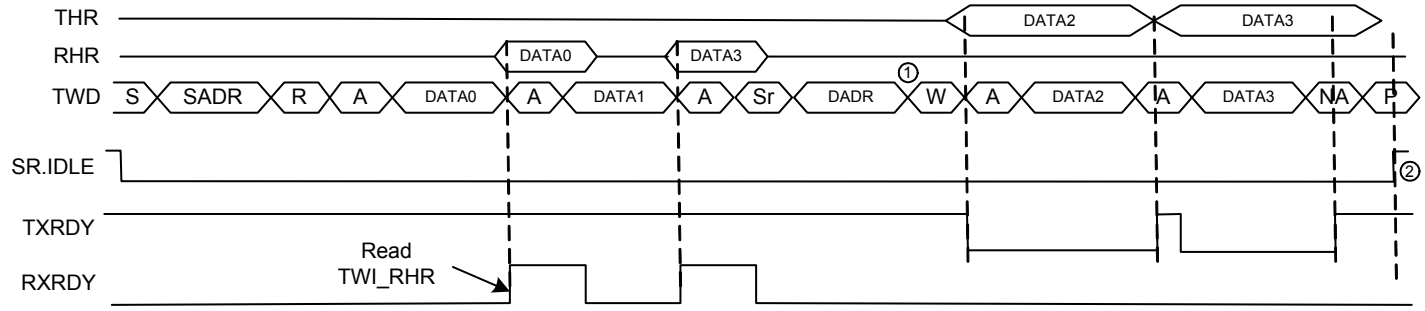
1. Write CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=0.
2. Write NCMDR with START=1, STOP=1, DADR, NBYTES=2 and READ=1.
3. Wait until SR.TXRDY==1, then write first data byte to transfer to THR.
4. Wait until SR.TXRDY==1, then write second data byte to transfer to THR.
5. Wait until SR.RXRDY==1,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

23.8.7.4 Read Followed by Write

Consider the following transfer:

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

Figure 23-13. Combining a Read and Write Transfer



To generate this transfer:

1. Write CMDR with START=1, STOP=0, DADR, NBYTES=2 and READ=1.
2. Write 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.

23.8.8 Ten Bit Addressing

Writing a one to 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 9:7 of CMDR.SADR must be written appropriately.

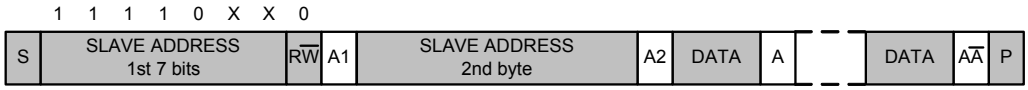
In Figure 23-14 and Figure 23-15, the grey boxes represent signals driven by the master, the white boxes are driven by the slave.

23.8.8.1 Master Transmitter

To perform a master transmitter transfer:

1. Write CMDR with TENBIT=1, REPSAME=0, READ=0, START=1, STOP=1 and the desired address and NBYTES value.

Figure 23-14. A Write Transfer with 10-bit Addressing



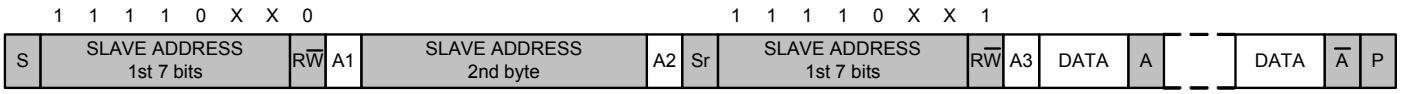
23.8.8.2 Master Receiver

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. Write CMDR with TENBIT=1, REPSAME=0, READ=0, START=1, STOP=0, NBYTES=0 and the desired address.
2. Write NCMDR with TENBIT=1, REPSAME=1, READ=1, START=1, STOP=1 and the desired address and NBYTES value.

Figure 23-15. A Read Transfer with 10-bit Addressing



23.8.9 SMBus Mode

SMBus mode is enabled and disabled by writing to 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 written into SMBTR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A set of addresses have been reserved for protocol handling, such as Alert Response Address (ARA) and Host Header (HH) Address.

23.8.9.1 Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to CMDR.PECEN 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 SR.PECERR 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 written with the total number of data bytes in the transmission, including the PEC byte.

In combined transfers, the PECEN bit should only be written to one 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.

23.8.9.2 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 set.

23.8.10 Identifying Bus Events

This chapter lists the different bus events, and how they affect bits in the TWIM registers. This is intended to help writing drivers for the TWIM.

Table 23-5. 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 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.

23.9 User Interface

Table 23-6. TWIM Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	0x00000000
0x04	Clock Waveform Generator Register	CWGR	Read/Write	0x00000000
0x08	SMBus Timing Register	SMBTR	Read/Write	0x00000000
0x0C	Command Register	CMDR	Read/Write	0x00000000
0x10	Next Command Register	NCMDR	Read/Write	0x00000000
0x14	Receive Holding Register	RHR	Read-only	0x00000000
0x18	Transmit Holding Register	THR	Write-only	0x00000000
0x1C	Status Register	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.

23.9.1 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
-	-	-	-	-	-	-	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.

23.9.2 Clock Waveform Generator Register

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
  $f_{PRESCALER} = \frac{f_{CLK\_TWIM}}{2^{(EXP+1)}}$

- **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}.

23.9.3 SMBus Timing Register

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

$$f_{prescaled,SMBus} = \frac{f_{CLKTWIM}}{2^{(EXP+1)}}$$

- **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 $T_{LOW:SEXT}$.

23.9.4 Command Register

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**
  0: 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.
  1: 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**
  0: 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.
  1: 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 one. In SMBus mode, if PEC is used, NBYTES includes the PEC byte, i.e. there are NBYTES-1 data bytes and a PEC byte.
- **VALID: CMDR Valid**
  0: Indicates that CMDR does not contain a valid command.
  1: Indicates that CMDR contains a valid command. This bit is cleared when the command is finished.
- **STOP: Send STOP Condition**
  0: Do not transmit a STOP condition after the data bytes have been transmitted.
  1: Transmit a STOP condition after the data bytes have been transmitted.
- **START: Send START Condition**
  0: The transfer in CMDR should not commence with a START or REPEATED START condition.
  1: 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**

  0: Use 7-bit addressing mode.

  1: Use 10-bit addressing mode. Must not be used when the 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**

  0: Allow the master to transmit data.

  1: Allow the master to receive data.

23.9.5 Next Command Register

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 content 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 content is copied immediately.

23.9.6 Receive Holding Register

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 one, this field contains a byte received from the TWI bus.

23.9.7 Transmit Holding Register

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.

23.9.8 Status Register

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	-	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 one when a STOP request caused by writing a one to 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 one 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 one when a SMBus timeout occurred.
  This bit is cleared by writing 1 to the corresponding bit in the Status Clear Register (SCR).
- **ARBLST: Arbitration Lost**
  This bit is one 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 one 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 one 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 one when activity has completed on the two-wire bus.
  Otherwise, this bit is cleared.
- **IDLE: Master Interface is Idle**
  This bit is one when no command is in progress, and no command waiting to be issued.
  Otherwise, this bit is cleared.

- **CCOMP: Command Complete**
  This bit is one when the current command has completed successfully.
  This bit is zero 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 one 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 one 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 one when RX data are ready to be read from RHR.
  This bit is cleared when this is no longer true.

23.9.9 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
-	STOP	PECERR	TOUT	-	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

23.9.10 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
-	STOP	PECERR	TOUT	-	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

23.9.11 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
-	STOP	PECERR	TOUT	-	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.

23.9.12 Status Clear Register

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	-	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.

23.9.13 Parameter Register (PR)

Name: PR

Access Type: Read-only

Offset: 0x30

Reset Value: -

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

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

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

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

23.9.14 Version Register (VR)

Name: VR

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.

23.10 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 23-7. Module Clock Name

Module Name	Clock Name
TWIM	CLK_TWIM

Table 23-8. Register Reset Values

Register	Reset Value
VERSION	0x0000 0110
PARAMETER	0x0000 0000

24. Two-wire Slave Interface (TWIS)

Rev.: 1.2.0.1

24.1 Features

- **Compatible with I²C standard**
  - **Transfer speeds of 100 and 400 kbit/s**
  - **7 and 10-bit and General Call addressing**
- **Compatible with SMBus standard**
  - **Hardware Packet Error Checking (CRC) generation and verification with ACK response25 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**

24.2 Overview

The Atmel Two-wire Slave Interface (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. The TWIS is always a bus slave and can transfer sequential or single bytes.

Below, Table 24-1 lists the compatibility level of the Atmel Two-wire Slave Interface and a full I²C compatible device.

Table 24-1. Atmel TWIS Compatibility with I²C Standard

I²C Standard	Atmel TWIS
Standard-mode (100 kbit/s)	Supported
Fast-mode (400 kbit/s)	Supported
7 or 10 bits Slave Addressing	Supported
START BYTE(1)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NAK Management	Supported
Slope control and input filtering (Fast mode)	Supported
Clock stretching	Supported

Note: 1. START + b000000001 + Ack + Sr

Below, Table 24-2 lists the compatibility level of the Atmel Two-wire Slave Interface and a full SMBus compatible device.

Table 24-2. Atmel TWIS Compatibility with SMBus Standard

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

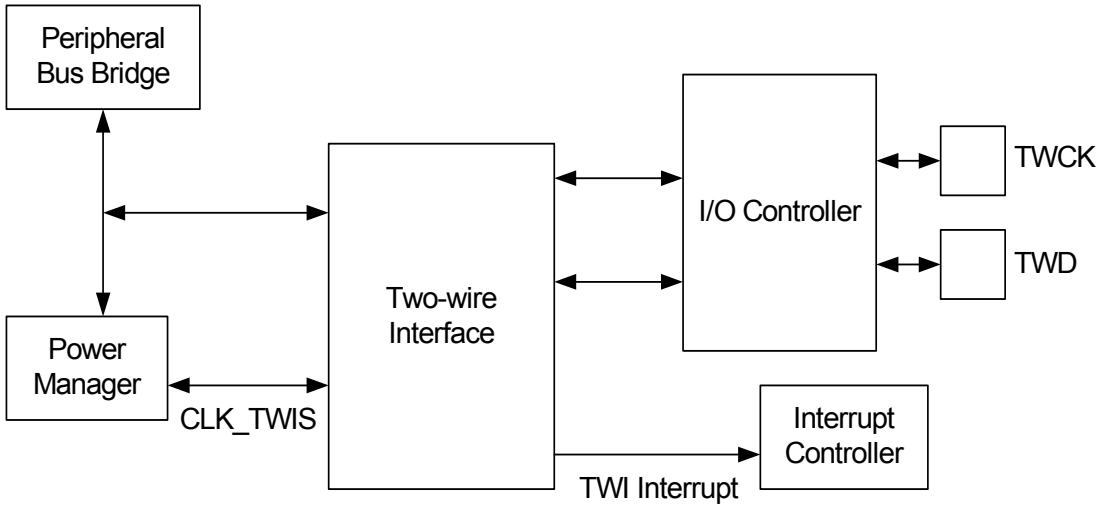
24.3 List of Abbreviations

Table 24-3. 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

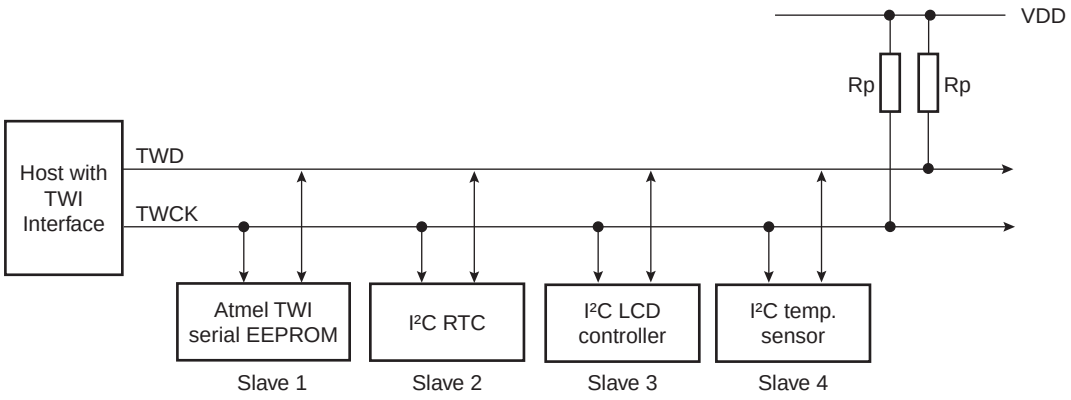
24.4 Block Diagram

Figure 24-1. Block Diagram



24.5 Application Block Diagram

Figure 24-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

24.6 I/O Lines Description

Table 24-4. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

24.7 Product Dependencies

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

24.7.1 I/O Lines

TWDand TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 24-5 on page 478). 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.

TWD and TWCK pins may be multiplexed with I/O Controller lines. To enable the TWIS, the user must perform the following steps:

- Program the I/O Controller to:
  - Dedicate TWD, TWCKas peripheral lines.
  - Define TWD, TWCKas open-drain.

24.7.2 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. The TWIS is able to wake the system from sleep mode upon address match, see Section 24.8.8 on page 485.

24.7.3 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.

24.7.4 DMA

The TWIS DMA handshake interface is connected to the Peripheral DMA Controller. Using the TWIS DMA functionality requires the Peripheral DMA Controller to be programmed after setting up the TWIS.

24.7.5 Interrupts

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

24.7.6 Debug Operation

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

24.8 Functional Description

24.8.1 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 24-4 on page 477).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 24-3).

- 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 24-3. START and STOP Conditions

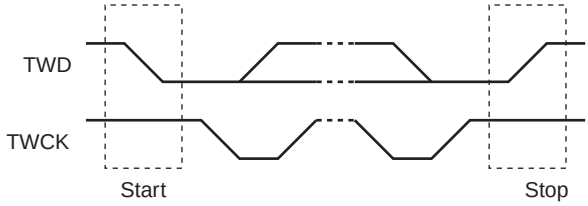
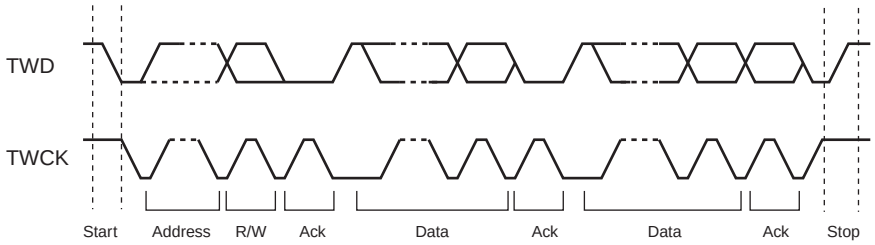


Figure 24-4. Transfer Format



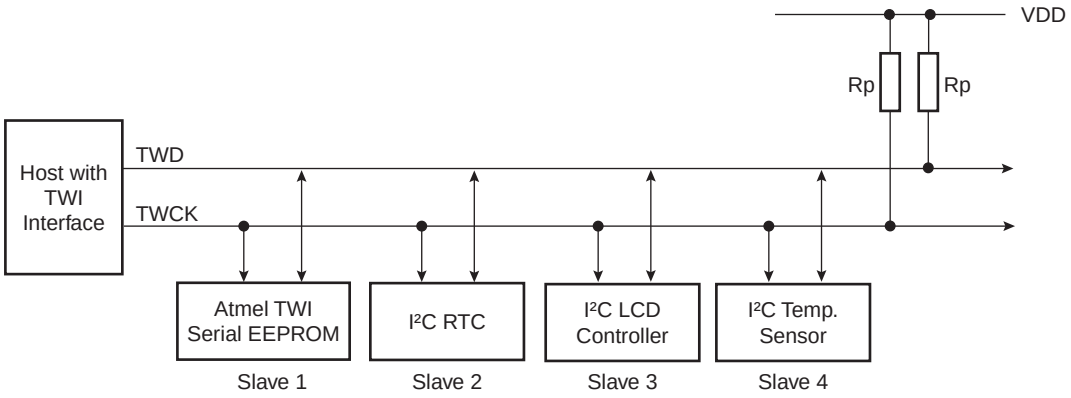
24.8.2 Operation

The 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 24-5. Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

24.8.2.1 Bus Timing

The Timing Register (TR) is used to control the timing of bus signals driven by the 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:

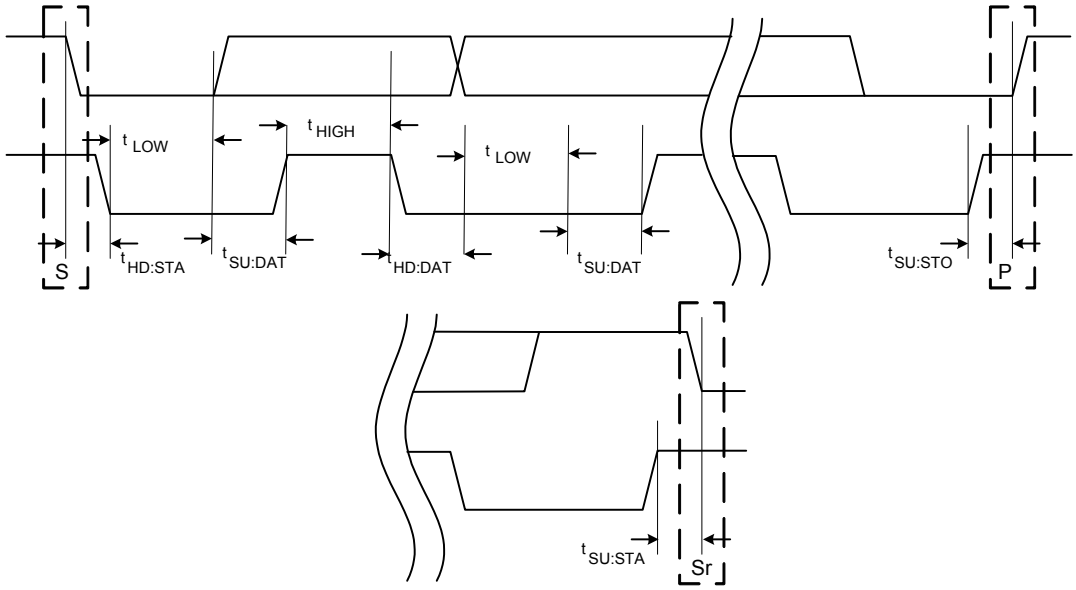
TLOWS: Prescaled clock cycles used to time SMBUS timeout $T_{LOW:SEXT}$.

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 24-6. Bus Timing Diagram



24.8.2.2 Setting Up and Performing a Transfer

Operation of the TWIS is mainly controlled by the Control Register (CR). The following list presents the main steps in a typical communication:

3. Before any transfers can be performed, bus timings must be configured by writing to the Timing Register (TR).If the Peripheral DMA Controller is to be used for the transfers, it must be set up.
4. 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 generate 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 zero. 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.

24.8.2.3 Address Matching

The TWIS can be set up to match several different addresses. More than one address match may be enabled simultaneously, allowing the TWIS to be assigned to several addresses. The address matching phase is initiated after a START or REPEATED START condition. When the 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 one.
- The General Call address is checked for address match if CR.GCMATCH is one.

In SMBus mode:

- The address in CR.ADR is checked for address match if CR.SMATCH is one.
- The Alert Response Address is checked for address match if CR.SMAL is one.
- The Default Address is checked for address match if CR.SMDA is one.
- The Host Header Address is checked for address match if CR.SMHH is one.

24.8.2.4 Clock Stretching

Any slave or bus master taking part in a transfer may extend the TWCK low period at any time. The TWIS may extend the TWCK low period after each byte transfer if CR.STREN is one 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 the Receive Holding Register (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 the Status Register (SR) have been cleared.

If CR.STREN is zero 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.

24.8.2.5 Bus Errors

If a bus error (misplaced START or STOP) condition is detected, the SR.BUSERR bit is set and the TWIS waits for a new START condition.

24.8.3 Slave Transmitter Mode

If the TWIS matches an address in which the R/$\overline{W}$ bit in the TWI address phase transfer is set, it will enter slave transmitter mode and set the SR.TRA bit (note that SR.TRA is set one CLK_TWIS cycle after the relevant address match bit in the same register is set).

After the address phase, the following actions are performed:

1. If SMBus mode and PEC is used, NBYTES must be set up with the number of bytes to transmit. This is necessary in order to know when to transmit the 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 is zero or NBYTES is non-zero: The TWIS waits until THR contains a valid data byte, possibly stretching the low period of TWCK. After THR contains a valid data byte, the data byte is transferred to a shifter, and then SR.TXRDY is changed to one because the THR is empty again.
   – SMBus mode and CR.PEC is one: If NBYTES is zero, the generated PEC byte is automatically transmitted instead of a data byte from THR. TWCK will not be stretched by the TWIS.
3. The data byte in the shifter is transmitted.
4. NBYTES is updated. If CR.CUP is one, NBYTES is incremented, otherwise NBYTES is decremented.
5. After each data byte has been transmitted, the master transmits an ACK (Acknowledge) or NAK (Not Acknowledge) bit. If a NAK bit is received by the TWIS, the SR.NAK bit is set. Note that this is done two CLK_TWIS cycles after TWCK has been sampled by the TWIS to be HIGH (see Figure 24-9). The NAK indicates that the transfer is fin-

ished, and the TWIS will wait for a STOP or REPEATED START. If an ACK bit is received, the SR.NAK bit remains LOW. The ACK indicates that more data should be transmitted, jump to step 2. At the end of the ACK/NAK clock cycle, the Byte Transfer Finished (SR.BTF) bit is set. Note that this is done two CLK_TWIS cycles after TWCK has been sampled by the TWIS to be LOW (see Figure 24-9). Also note that in the event that SR.NAK bit is set, it must not be cleared before the SR.BTF bit is set to ensure correct TWIS behavior.

6. If STOP is received, SR.TCOMP and SR.STO will be set.
7. If REPEATED START is received, SR.REP will be set.

The TWI transfers require the receiver to acknowledge each received data byte. During the acknowledge clock pulse (9th pulse), the slave releases the data line (HIGH), enabling the master to pull it down in order to generate the acknowledge. The slave polls the data line during this clock pulse and sets the NAK bit in SR 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).

SR.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 changing from zero to one. See Figure 24-7 and Figure 24-8.

Figure 24-7. Slave Transmitter with One Data Byte

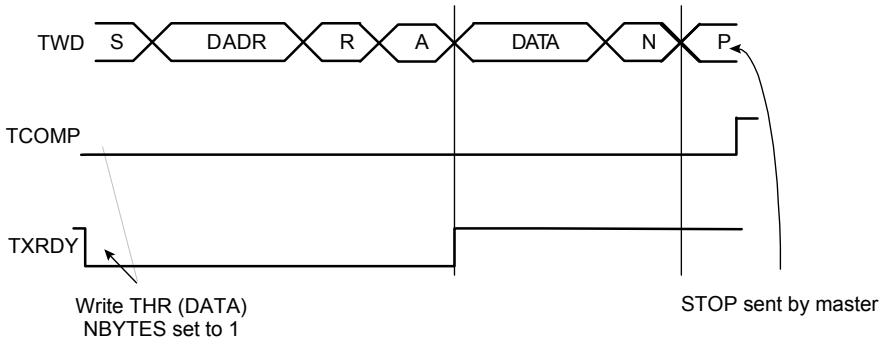


Figure 24-8. Slave Transmitter with Multiple Data Bytes

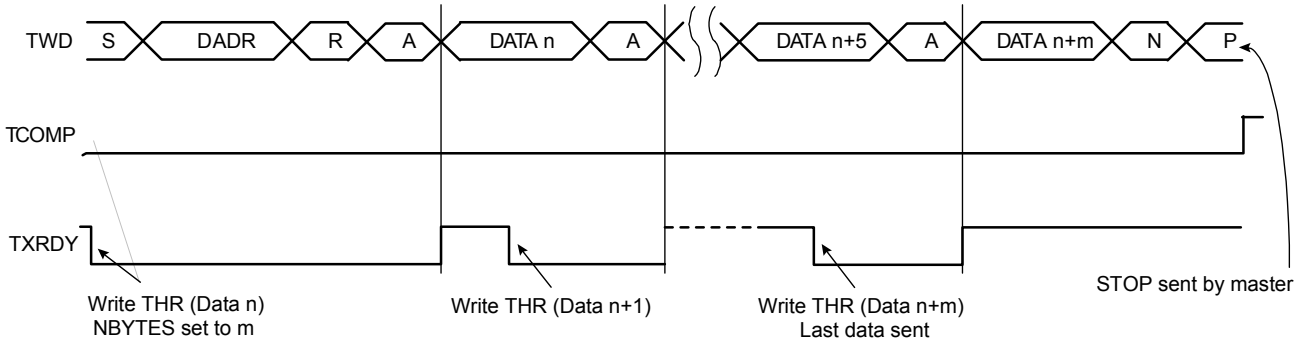
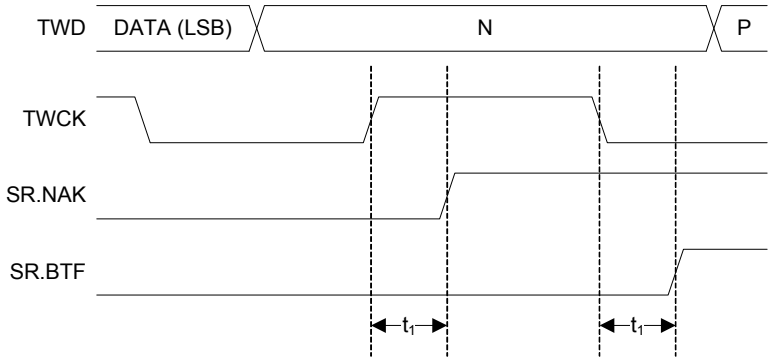


Figure 24-9. Timing Relationship between TWCK, SR.NAK, and SR.BTF



t_1: (CLK_TWIS period) x 2

24.8.4 Slave Receiver Mode

If the TWIS matches an address in which the R/$\overline{W}$ bit in the TWI address phase transfer is cleared, it will enter slave receiver mode and clear SR.TRA (note that SR.TRA is cleared one CLK_TWIS cycle after the relevant address match bit in the same register is set).

After the address phase, the following is repeated:

1. If SMBus mode and PEC is used, NBYTES must be set up with the number of bytes to receive. This is necessary in order to know which of the received bytes is the PEC byte. NBYTES can also be used to count the number of bytes received if using DMA.
2. Receive a byte. Set SR.BTF when done.
3. Update NBYTES. If CR.CUP is written to one, NBYTES is incremented, otherwise NBYTES is decremented. NBYTES is usually configured to count downwards if PEC is used.
4. After a data byte has been received, the slave transmits an ACK or NAK bit. For ordinary data bytes, the CR.ACK field controls if an ACK or NAK should be returned. If PEC is enabled and the last byte received was a PEC byte (indicated by NBYTES equal to zero), The 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.

The SR.RXRDY bit indicates that a data byte is available in the RHR. The RXRDY bit is also used as Receive Ready for the Peripheral DMA Controller receive channel.

Figure 24-10. Slave Receiver with One Data Byte

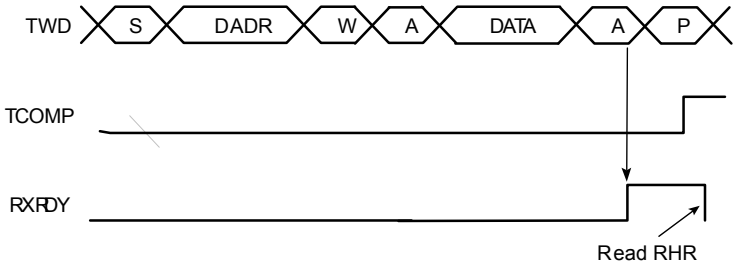
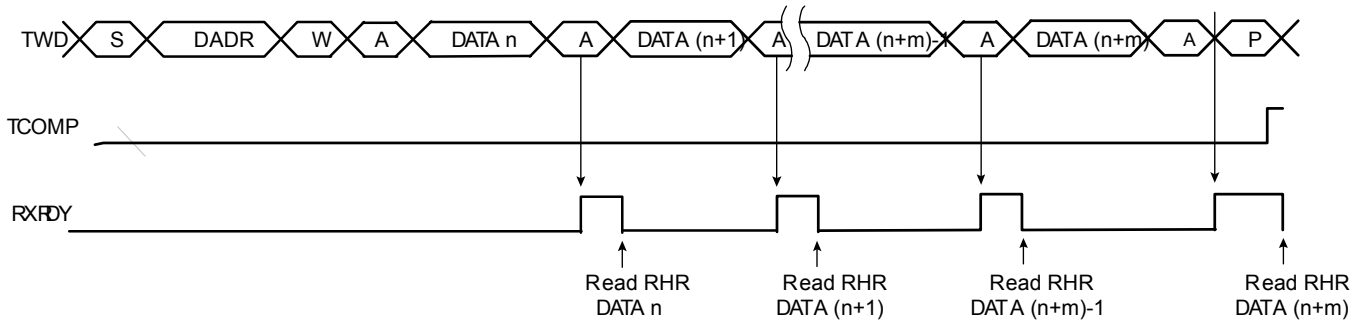


Figure 24-11. Slave Receiver with Multiple Data Bytes



24.8.5 Interactive ACKing Received Data Bytes

When implementing a register interface over TWI, it may sometimes be necessary or just useful to report reads and writes to invalid register addresses by sending a NAK to the host. To be able to do this, one must first receive the register address from the TWI bus, and then tell the TWIS whether to ACK or NAK it. In normal operation of the TWIS, this is not possible because the controller will automatically ACK the byte at about the same time as the RXRDY bit changes from zero to one. Writing a one to the Stretch on Data Byte Received bit (CR.SODR) will stretch the clock allowing the user to update CR.ACK bit before returning the desired value. After the last bit in the data byte is received, the TWI bus clock is stretched, the received data byte is transferred to the RHR register, and SR.BTF is set. At this time, the user can examine the received byte and write the desired ACK or NACK value to CR.ACK. When the user clears SR.BTF, the desired ACK value is transferred on the TWI bus. This makes it possible to look at the byte received, determine if it is valid, and then decide to ACK or NAK it.

24.8.6 Using the Peripheral DMA Controller

The use of the Peripheral DMA Controller significantly reduces the CPU load. The user can set up ring buffers for the Peripheral DMA Controller, containing data to transmit or free buffer space to place received data. By initializing NBYTES to zero before a transfer, and writing a one to CR.CUP, NBYTES is incremented by one each time a data has been transmitted or received. This allows the user to detect how much data was actually transferred by the DMA system.

To assure correct behavior, respect the following programming sequences:

24.8.6.1 Data Transmit with the Peripheral DMA Controller

1. Initialize the transmit Peripheral DMA Controller (memory pointers, size, etc.).
2. Configure the TWIS (ADR, NBYTES, etc.).

3. Start the transfer by enabling the Peripheral DMA Controller to transmit.
4. Wait for the Peripheral DMA Controller end-of-transmit flag.
5. Disable the Peripheral DMA Controller.

24.8.6.2 Data Receive with the Peripheral DMA Controller

1. Initialize the receive Peripheral DMA Controller (memory pointers, size - 1, etc.).
2. Configure the TWIS (ADR, NBYTES, etc.).
3. Start the transfer by enabling the Peripheral DMA Controller to receive.
4. Wait for the Peripheral DMA Controller end-of-receive flag.
5. Disable the Peripheral DMA Controller.

24.8.7 SMBus Mode

SMBus mode is enabled by writing a one to the SMBus Mode Enable (SMEN) bit 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 written to TR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- 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.

24.8.7.1 Packet Error Checking (PEC)

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to the Packet Error Checking Enable (PECEN) bit in CR enables automatic PEC handling in the current transfer. The PEC generator is always updated on every bit transmitted or received, so that PEC handling on following linked transfers will be correct.

In slave receiver mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave will compare it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave will return an ACK to the master. If the PEC values differ, data was corrupted, and the slave will return a NAK value. The SR.SMBPECERR bit is set automatically if a PEC error occurred.

In slave transmitter mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master will compare it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the master must take appropriate action.

The PEC byte is automatically inserted in a slave transmitter transmission if PEC enabled when NBYTES reaches zero. The PEC byte is identified in a slave receiver transmission if PEC enabled when NBYTES reaches zero. NBYTES must therefore be set to the total number of data bytes in the transmission, including the PEC byte.

24.8.7.2 Timeouts

The Timing Register (TR) configures the SMBus timeout values. If a timeout occurs, the slave will leave the bus. The SR.SMBTOUT bit is also set.

24.8.8 Wakeup from Sleep Modes by TWI Address Match

The TWIS is able to wake the device up from a sleep mode upon an address match, including sleep modes where CLK_TWIS is stopped. After detecting the START condition on the bus, The TWIS will stretch TWCK until CLK_TWIS has started. The time required for starting CLK_TWIS depends on which sleep mode the device is in. After CLK_TWIS has started, the TWIS releases its TWCK stretching and receives one byte of data on the bus. At this time, only a limited part of the device, including the TWIS, receives a clock, thus saving power. The TWIS goes on to receive the slave address. If the address phase causes a TWIS address match, the entire device is wakened and normal TWIS address matching actions are performed. Normal TWI transfer then follows. If the TWIS is not addressed, CLK_TWIS is automatically stopped and the device returns to its original sleep mode.

24.8.9 Identifying Bus Events

This chapter lists the different bus events, and how these affects the bits in the TWIS registers. This is intended to help writing drivers for the TWIS.

Table 24-5. Bus Events

Event	Effect
Slave transmitter has sent a data byte	SR.THR is cleared. SR.BTF is set. The value of the ACK bit sent immediately after the data byte is given by CR.ACK.
Slave receiver has received a data byte	SR.RHR is set. SR.BTF is set. SR.NAK updated according to value of ACK bit received from master.
Start+Sadr on bus, but address is to another slave	None.
Start+Sadr on bus, current slave is addressed, but address match enable bit in CR is not set	None.
Start+Sadr on bus, current slave is addressed, corresponding address match enable bit in CR set	Correct address match bit in SR is set. SR.TRA updated according to transfer direction (updating is done one CLK_TWIS cycle after address match bit is set) Slave enters appropriate transfer direction mode and data transfer can commence.
Start+Sadr on bus, current slave is addressed, corresponding address match enable bit in CR set, SR.STREN and SR.SOAM are set.	Correct address match bit in SR is set. SR.TRA updated according to transfer direction (updating is done one CLK_TWIS cycle after address match bit is set). Slave stretches TWCK immediately after transmitting the address ACK bit. TWCK remains stretched until all address match bits in SR have been cleared. Slave 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.

Table 24-5. Bus Events

Event	Effect
Start, Repeated Start, or Stop received in illegal position on bus	SR.BUSERR set. SR.STO and SR.TCOMP may or may not be set depending on the exact position of an illegal stop.
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.

24.9 User Interface

Table 24-6. 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.

24.9.1 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
SODR	SOAM	CUP	ACK	PECEN	SMHH	SMDA	-

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

- **TENBIT: Ten Bit Address Match**
  0: Disables Ten Bit Address Match.
  1: Enables Ten Bit Address Match.
- **ADR: Slave Address**
  Slave address used in slave address match. Bits 9:0 are used if in 10-bit mode, bits 6:0 otherwise.
- **SODR: Stretch Clock on Data Byte Reception**
  0: Does not stretch bus clock immediately before ACKing a received data byte.
  1: Stretches bus clock immediately before ACKing a received data byte.
- **SOAM: Stretch Clock on Address Match**
  0: Does not stretch bus clock after address match.
  1: Stretches bus clock after address match.
- **CUP: NBYTES Count Up**
  0: Causes NBYTES to count down (decrement) per byte transferred.
  1: Causes NBYTES to count up (increment) per byte transferred.
- **ACK: Slave Receiver Data Phase ACK Value**
  0: Causes a low value to be returned in the ACK cycle of the data phase in slave receiver mode.
  1: Causes a high value to be returned in the ACK cycle of the data phase in slave receiver mode.
- **PECEN: Packet Error Checking Enable**
  0: Disables SMBus PEC (CRC) generation and check.
  1: Enables SMBus PEC (CRC) generation and check.
- **SMHH: SMBus Host Header**
  0: Causes the TWIS not to acknowledge the SMBus Host Header.
  1: Causes the TWIS to acknowledge the SMBus Host Header.
- **SMDA: SMBus Default Address**
  0: Causes the TWIS not to acknowledge the SMBus Default Address.
  1: Causes the TWIS to acknowledge the SMBus Default Address.
- **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**
  0: Disables clock stretching if RHR/THR buffer full/empty. May cause over/underrun.
  1: Enables clock stretching if RHR/THR buffer full/empty.
- **GCMATCH: General Call Address Match**
  0: Causes the TWIS not to acknowledge the General Call Address.
  1: Causes the TWIS to acknowledge the General Call Address.
- **SMATCH: Slave Address Match**
  0: Causes the TWIS not to acknowledge the Slave Address.
  1: Causes the TWIS to acknowledge the Slave Address.
- **SMEN: SMBus Mode Enable**
  0: Disables SMBus mode.
  1: Enables SMBus mode.
- **SEN: Slave Enable**
  0: Disables the slave interface.
  1: Enables the slave interface.

24.9.2 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. The field can also be read to learn the progress of the transfer. NBYTES can be incremented or decremented automatically by hardware.

24.9.3 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:

$$f_{PRESCALED} = \frac{f_{CLK\_TWIS}}{2^{(EXP+1)}}$$

- **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 $T_{TIMEOUT}$ Cycles**
  Prescaled clock cycles used to time SMBus $T_{TIMEOUT}$.
- **TLOWS: SMBus $T_{LOW:SEXT}$ Cycles**
  Prescaled clock cycles used to time SMBus $T_{LOW:SEXT}$.

24.9.4 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 one, this field contains a byte received from the TWI bus.

24.9.5 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.

24.9.6 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.

24.9.7 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	-	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 cleared when the corresponding bit in SCR is written to one.
  This bit is set when byte transfer has completed.
- **REP: Repeated Start Received**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when a REPEATED START condition is received.
- **STO: Stop Received**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the STOP condition is received.
- **SMBDAM: SMBus Default Address Match**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the received address matched the SMBus Default Address.
- **SMBHHM: SMBus Host Header Address Match**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the received address matched the SMBus Host Header Address.

- **GCM: General Call Match**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the received address matched the General Call Address.
- **SAM: Slave Address Match**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when the received address matched the Slave Address.
- **BUSERR: Bus Error**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when a misplaced START or STOP condition has occurred.

- **SMBPECERR: SMBus PEC Error**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when a SMBus PEC error has occurred.
- **SMBTOUT: SMBus Timeout**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when a SMBus timeout has occurred.
- **NAK: NAK Received**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when a NAK was received from the master during slave transmitter operation.
- **ORUN: Overrun**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when an overrun has occurred in slave receiver mode. Can only occur if CR.STREN is zero.
- **URUN: Underrun**
  This bit is cleared when the corresponding bit in SCR is written to one.
  This bit is set when an underrun has occurred in slave transmitter mode. Can only occur if CR.STREN is zero.
- **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 cleared when the corresponding bit in SCR is written to one.
  This bit is set when transmission is complete. Set after receiving a STOP after being addressed.
- **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.

24.9.8 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	-	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 write a one to the corresponding bit in IMR.

24.9.9 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	-	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.

24.9.10 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	-	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.

24.9.11 Status Clear Register

Name: SCR

Access Type: Write-only

Offset: 0x28

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
BTF	REP	STO	SMBDAM	SMBHHM	-	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.

24.9.12 Parameter Register

Name: PR

Access Type: Read-only

Offset: 0x2C

Reset Value: -

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

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

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

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

24.9.13 Version Register (VR)

Name: VR

Access Type: Read-only

Offset: 0x30

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.

24.10 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 24-7. Module Clock Name

Module Name	Clock Name
TWIS	CLK_TWIS

Table 24-8. Register Reset Values

Register	Reset Value
VERSION	0x00000120
PARAMETER	0x00000000

25. Pulse Width Modulation Controller (PWMA)

Rev: 2.0.0.0

25.1 Features

- **Left-aligned non-inverted 12-bit PWM**
- **Common 12-bit timebase counter**
  - **Asynchronous clock source supported**
  - **Spread-spectrum counter to allow a constantly varying duty cycle**
- **Separate 12-bit duty cycle register per channel**
- **Synchronized channel updates**
  - **No glitches when changing the duty cycles**
- **Interlinked operation supported**
  - **Up to 32 channels can be updated with the same duty cycle value at a time**
  - **Up to 4 channels can be updated with different duty cycle values at a time**
- **Interrupt on PWM timebase overflow**
- **Output PWM waveforms**
  - **Support normal waveform output for each channel**
  - **Support composite waveform generation (XOR'ed) for each pair channels**

25.2 Overview

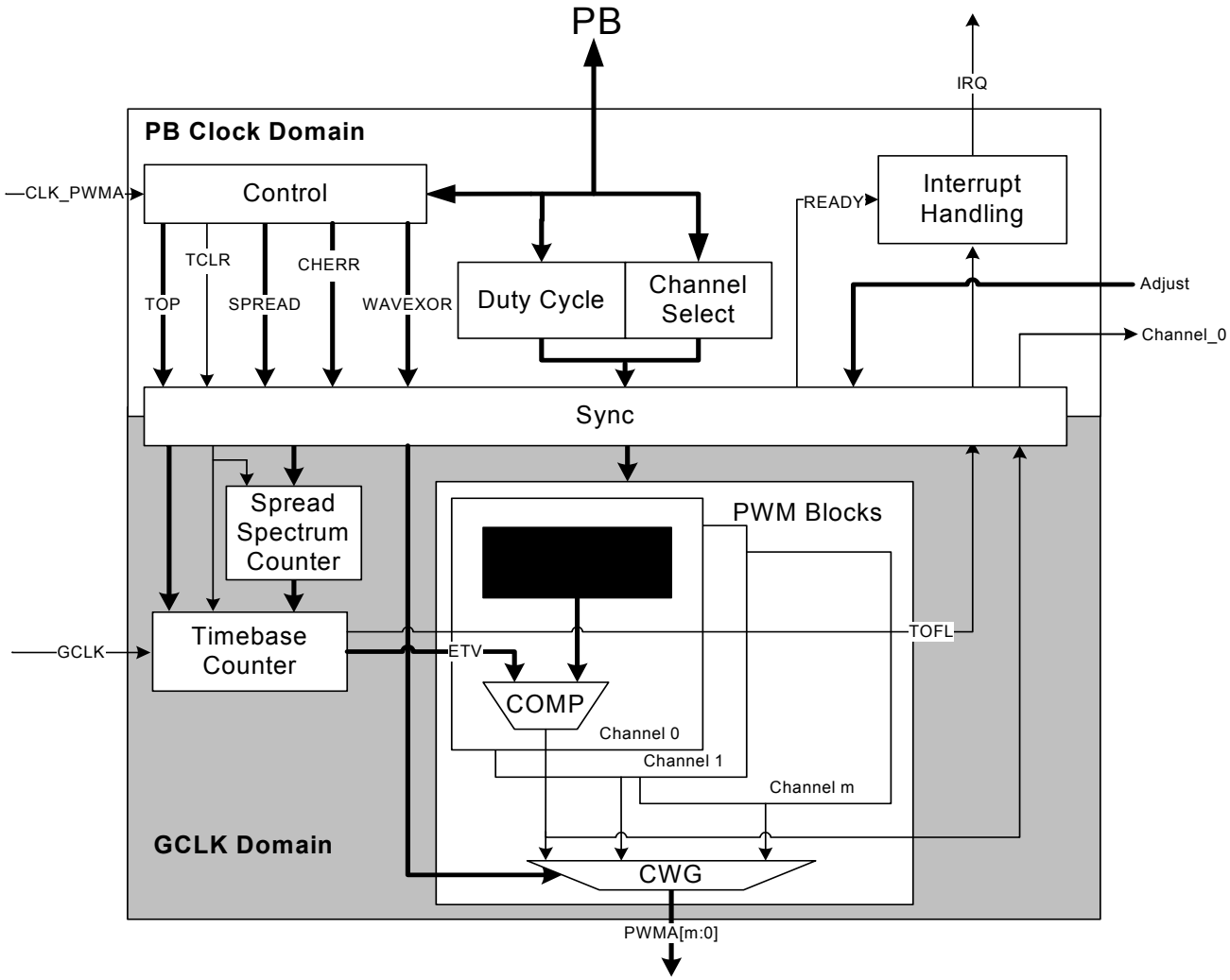
The Pulse Width Modulation Controller (PWMA) controls several pulse width modulation (PWM) channels. The number of channels is specific to the device. Each channel controls one square output PWM waveform. Characteristics of the output PWM waveforms such as period and duty cycle are configured through 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. There is also an interlinked multi-value mode, where the 8 least significant bits of up to 4 channels can be updated with 4 different values while the other channels remain unchanged.

25.3 Block Diagram

Figure 25-1. PWMA Block Diagram



25.4 I/O Lines Description

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

Table 25-1. I/O Line Description

Pin Name	Pin Description	Type
PWMA[n]	Output PWM waveform for one channel n	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 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.

25.5.2 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.

25.5.3 Clocks

The clock for the PWMA bus interface (CLK_PWMA) is controlled 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.

25.5.4 Interrupts

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

25.5.5 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.

25.6 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.

25.6.1 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).

25.6.2 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 Top Value Register (TVR.TOP) in normal operation (the

SPREAD field of CR (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:

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

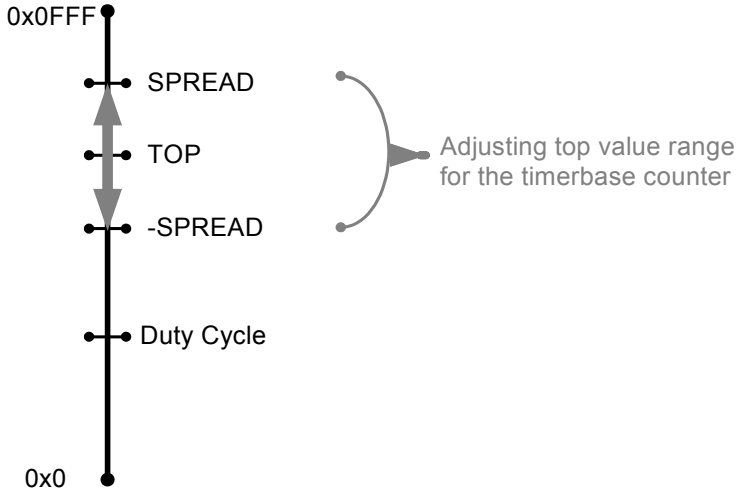
The timebase counter can be reset by writing a one to the Timebase Clear bit in CR (CR.TCLR). Note that this can cause a glitch to the output PWM waveforms in use.

25.6.3 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 to vary from TOP-SPREAD to TOP+SPREAD. Figure 25-2 on page 507 illustrates this. This leads to a constantly varying duty cycle on the PWM output waveforms though the duty cycle values stored are unchanged.

Figure 25-2. PWMA Adjusting Top Value for Timebase Counter



25.6.3.1 Special considerations

The maximum value of the timebase counter is 0x0FFF. 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 spectrum 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.

25.6.4 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 Section25.6.6.

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:

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

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 25-2.

Table 25-2. 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

25.6.5 Waveform Output

PWMA waveforms are output to I/O lines. The output waveform properties are controlled by Composite Waveform Generation (CWG) register(s). If this register is cleared (by default), the channel waveforms are out directly to the I/O lines. To avoid too many I/O toggling simultaneously on the output I/O lines, every other output PWM waveform toggles on the negative edge of the GCLK instead of the positive edge.

In CWG mode, all channels are paired and their outputs are XOR'ed together if the corresponding bit of CWG register is set. The even number of output is the XOR'ed output and the odd number of output is the inverse of its. Each bit of CWG register controls one pair channels and the least significant bit refers to the lowest number of pair channels.

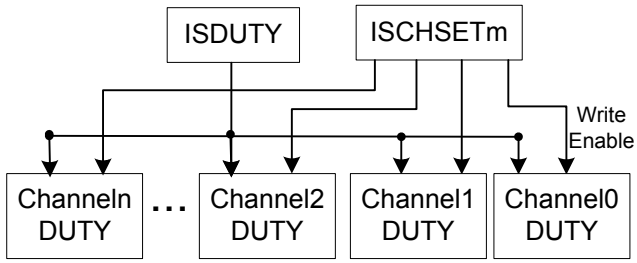
25.6.6 Updating Duty Cycle Values

25.6.6.1 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 (ISCHSET) register(s). When a bit is written to one in the ISCHSET 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 25-3 on page 509 shows the writing procedure. It is thus possible to update the duty cycle values for up to 32 PWM channels within one ISCHSET register at a time.

Figure 25-3. Interlinked Single Value PWM Operation Flow



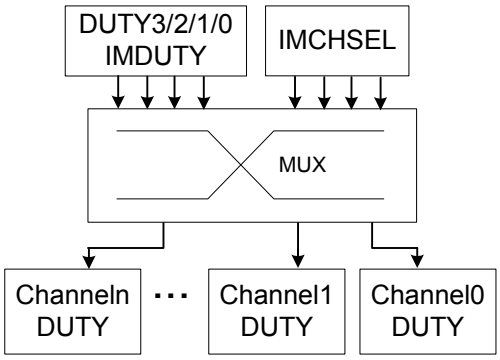
25.6.6.2 *Interlinked Multiple Value PWM Operation*

The interlinked multiple value PWM operation allows up to four channels to be updated simultaneously with different duty cycle values. The four duty cycle values are required to be written to the four registers, DUTY3, DUTY2, DUTY1 and DUTY0 , respectively. 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 DUTY0/1/2/3 registers are synchronized to the duty cycle registers for the channels selected by the SEL fields. Figure 25-4 on page 509 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 DUTYx register will not be written. If the same channel is specified multiple times in the IMCHSEL.SEL fields, the channel will be updated with the value referred by the upper IMCHSEL.SEL field.

When only the least significant 8-bits duty cycle value are considered for updating, the four duty cycle values can be written to the IMDUTY register once. This is equivalent to writing the four duty cycle values to the four DUTY registers one by one.

Figure 25-4. Interlinked Multiple Value PWM Operation Flow



25.6.7 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.

25.6.8 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).

25.7 User Interface

Table 25-3. 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)
0x2C	Top Value Register	TVR	Read/Write	0x00000000
0x30+m*0x10	Interlinked Single Value Channel Set m	ISCHSETm	Write-only	0x00000000
0x3C+k*0x10	CWG Register	CWGk	Read/Write	0x00000000
0x80	Interlinked Multiple Value Duty0 Register	DUTY0	Write-only	0x00000000
0x84	Interlinked Multiple Value Duty1 Register	DUTY1	Write-only	0x00000000
0x88	Interlinked Multiple Value Duty2 Register	DUTY2	Write-only	0x00000000
0x8C	Interlinked Multiple Value Duty3 Register	DUTY3	Write-only	0x00000000

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

25.7.1 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

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

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

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 Section25.6.3 for more information.
- **TOP: Timebase Counter Top Value**

  The top value for the timebase counter. The value written to this field will update the least significant 8 bits of the TVR.TOP field in case only 8-bits resolution is required. The 4 most significant bits of TVR.TOP will be written to 0. When the TVR.TOP field is written, this CR.TOP field will also be updated with only the least significant 8 bits of TVR.TOP field.
- **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

25.7.2 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
-	-	-	-	DUTY[11:8]			

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

- **DUTY: Duty Cycle Value**

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

25.7.3 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 automatically written to the least significant 8 bits of the DUTYn register for a PWMA channel while the most significant 4bits of the DUTYn register are unchanged. Which channel is selected for updating is defined by the corresponding SEL field in the IMCHSEL register.

  To write mulitple channels at a time with more than 8 bits of the duty cycle value, refer to DUTY3/2/1/0 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.

25.7.4 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 stored in the DUTYn 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 DUTY3, DUTY2, DUTY1 and DUTY0 registers when the IMCHSEL register is written. Synchronization takes place immediately 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. When only 8 bits duty cycle value are considered for updating, the four duty cycle values can be written to the IMDUTY register once. This is equivalent to writing the 8 bits four duty cycle values to the four DUTY registers one by one while the upper 4 bits remain unchanged.

25.7.5 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.

25.7.6 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.

25.7.7 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.

25.7.8 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.

25.7.9 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.

25.7.10 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.

25.7.11 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.

25.7.12 Top Value Register

Name: TVR

Access Type: Read/Write

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
-	-	-	-	TOP[11:8]			
7	6	5	4	3	2	1	0
TOP[7:0]							

- **TOP: Timebase Counter Top Value**

  The top value for the timebase counter. The value written to the CR.TOP field will automatically be written to the 8 least significant bits of this field while the 4 most significant bits will be 0. When this register is written, it will also automatically update the CR.TOP field with the 8 least significant bits.

  The effective top value of the timebase counter is defined by both TVR.TOP and the CR.SPREAD. Refer to Section25.6.2 for more information.

25.7.13 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 immediately 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.

25.7.14 Composite Waveform Generation

Name: CWG

Access Type: Read/Write

Offset: 0x3C+k*0x10

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
XOR							

23	22	21	20	19	18	17	16
XOR							

15	14	13	12	11	10	9	8
XOR							

7	6	5	4	3	2	1	0
XOR							

- **XOR: Pair Waveform XOR'ed**
  If the bit *n* in XOR field is one, the pair of PWMA output waveforms will be XORed before output. The even number output will be the XOR'ed output and the odd number output will be reverse of it. For example, if bit 0 in XOR is one, the pair of PWMA output waveforms for channel 0 and 1 will be XORed together.
  If bit *n* in XOR is zero, normal waveforms are output for that pair. Note that
  If more than one CWG register is present, CWG0 controls the first 32 pairs, corresponding to channels 63 downto 0, and CWG1 controls the second 32 pairs, corresponding to channels 127 downto 64.

25.7.15 Interlinked Multiple Value Duty0/1/2/3 Register

Name: DUTY0/1/2/3

Access Type: Write-only

Offset: 0x80-0x8C

Reset Value: 0x00000000

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

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

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

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

These registers allows up to 4 channels to be updated with a common 12-bits duty cycle value at a time. They are the extension of the IMDUTY register which only supports updating the least significant 8 bits of the duty registers for up to 4 channels.

- **DUTY: Duty Cycle Value**
  The duty cycle value written to this field will be updated to the channel specified by IMCHSEL.
  DUTY0 is specified by IMCHSEL.SEL0, DUTY1 is specified by IMCHSEL.SEL1, and so on.

25.8 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 25-4. PWMA Configuration

Feature	PWMA
Number of PWM channels	7

Table 25-5. PWMA Clocks

Clock Name	Description
CLK_PWMA	Clock for the PWMA bus interface
GCLK_PWMA	PWMA output clock source. The generic clock used for the PWMA is GCLK4

Table 25-6. Register Reset Values

Register	Reset Value
VERSION	0x00000200
PARAMETER	0x0000007

26. Timer/Counter (TC)

Rev: 2.2.3.3

26.1 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**

26.2 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.

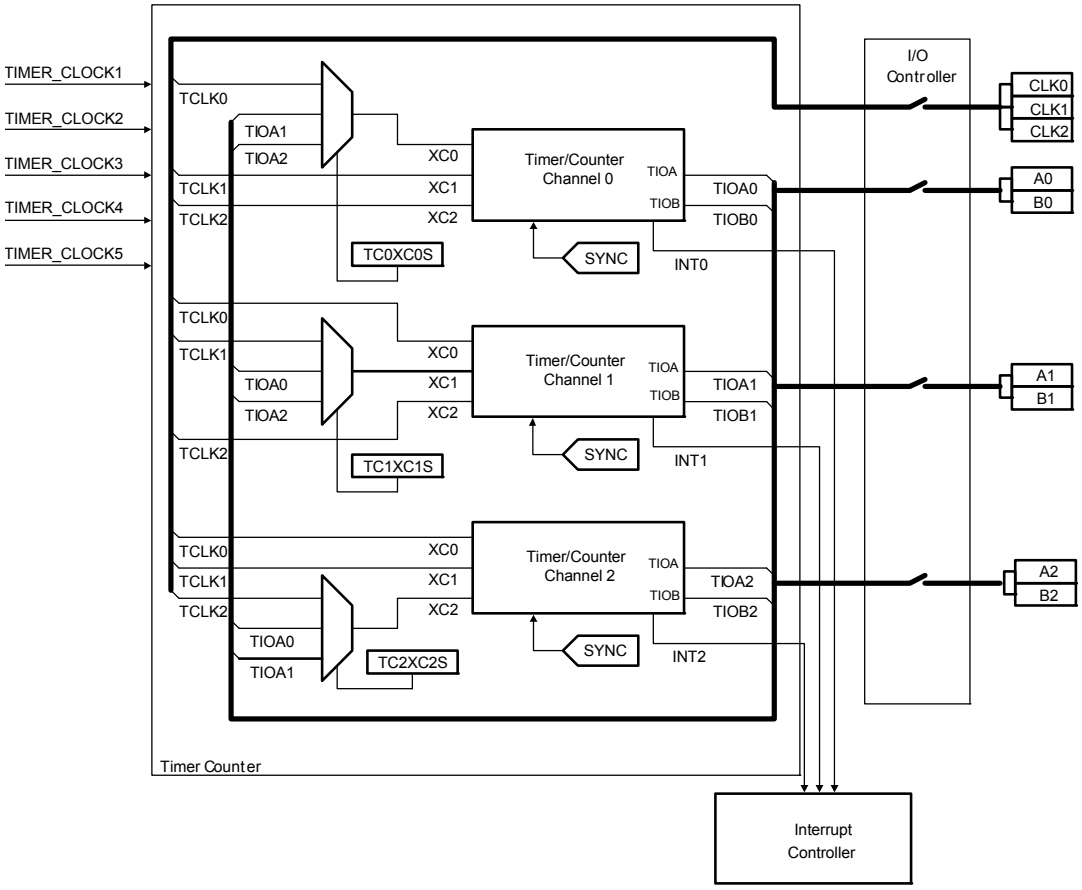
The TC block has two global registers which act upon all three TC channels.

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.

26.3 Block Diagram

Figure 26-1. TC Block Diagram



26.4 I/O Lines Description

Table 26-1. 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

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 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.

26.5.2 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.

26.5.3 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.

26.5.4 Interrupts

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

26.5.5 Debug Operation

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

26.6 Functional Description

26.6.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Figure 26-3 on page 545.

26.6.1.1 Channel I/O Signals

As described in Figure 26-1 on page 529, each Channel has the following I/O signals.

Table 26-2. 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

26.6.1.2 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.

26.6.1.3 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 26-2 on page 531.

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.

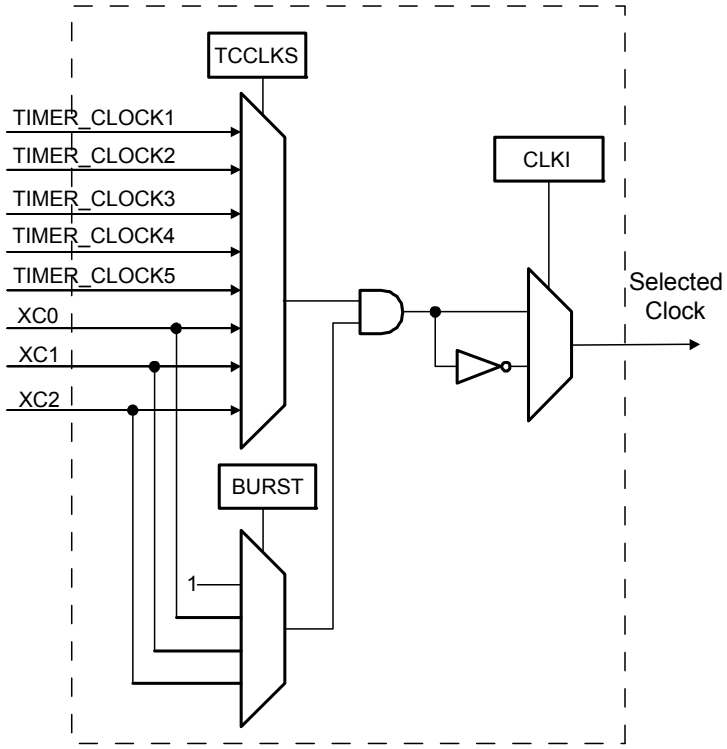
This selection is made by the Clock Selection field in the Channel n Mode Register (CMRn.TCCLKS).

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 26-2. Clock Selection

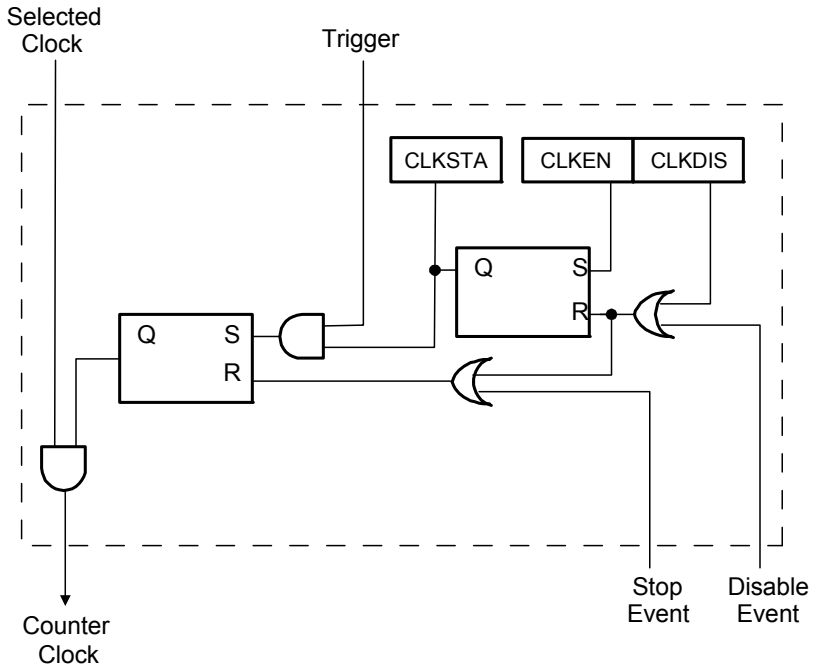


26.6.1.4 Clock control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 26-3 on page 532.

- 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 26-3. Clock Control



26.6.1.5 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.

26.6.1.6 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.

26.6.2 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 26-4 on page 535 shows the configuration of the TC channel when programmed in Capture mode.

26.6.2.1 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.

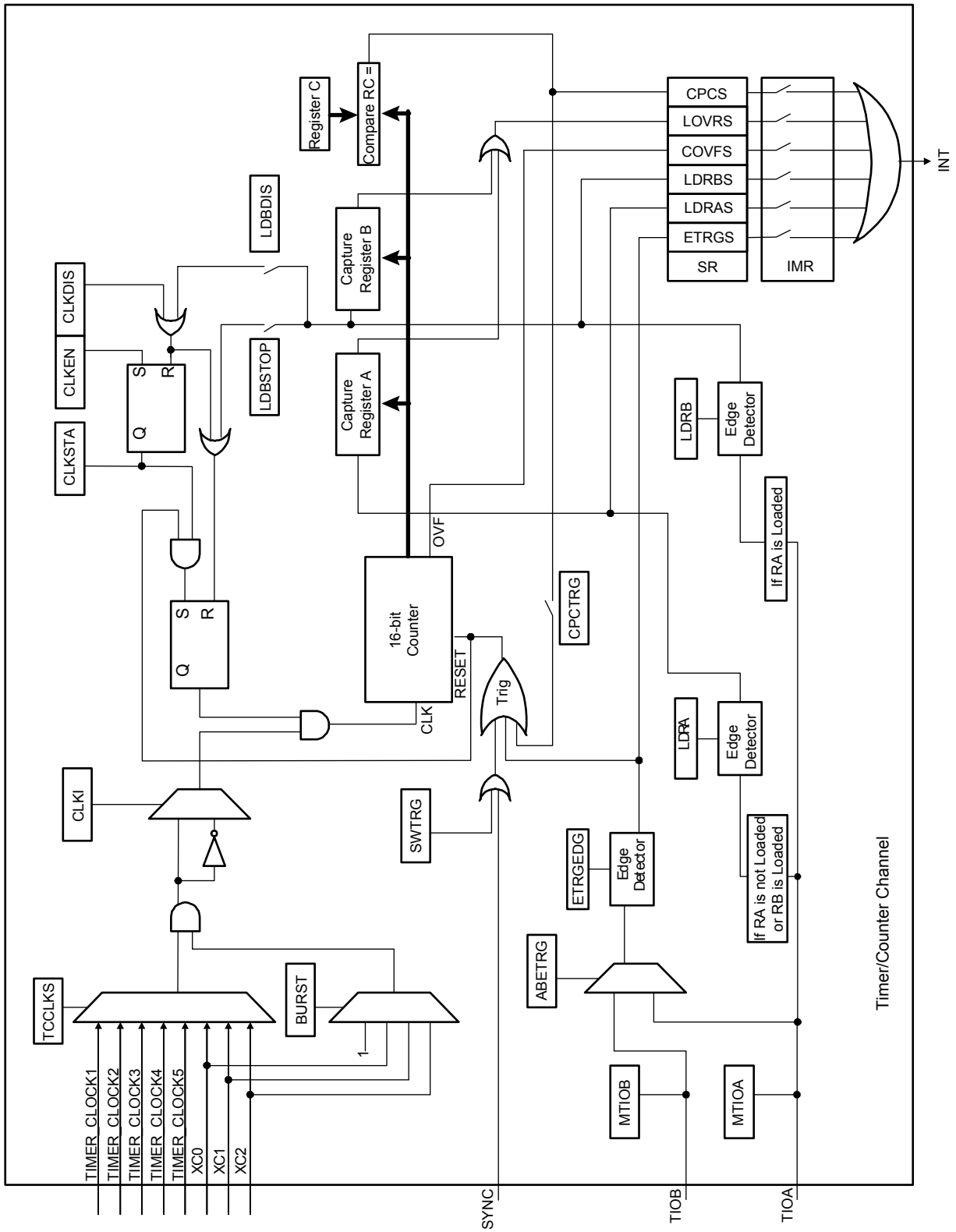
Loading RA or RB before the read of the last value loaded sets the Load Overrun Status bit in SRn (SRn.LOVRS). In this case, the old value is overwritten.

26.6.2.2 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 26-4. Capture Mode



26.6.3 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 26-5 on page 537 shows the configuration of the TC channel when programmed in Waveform operating mode.

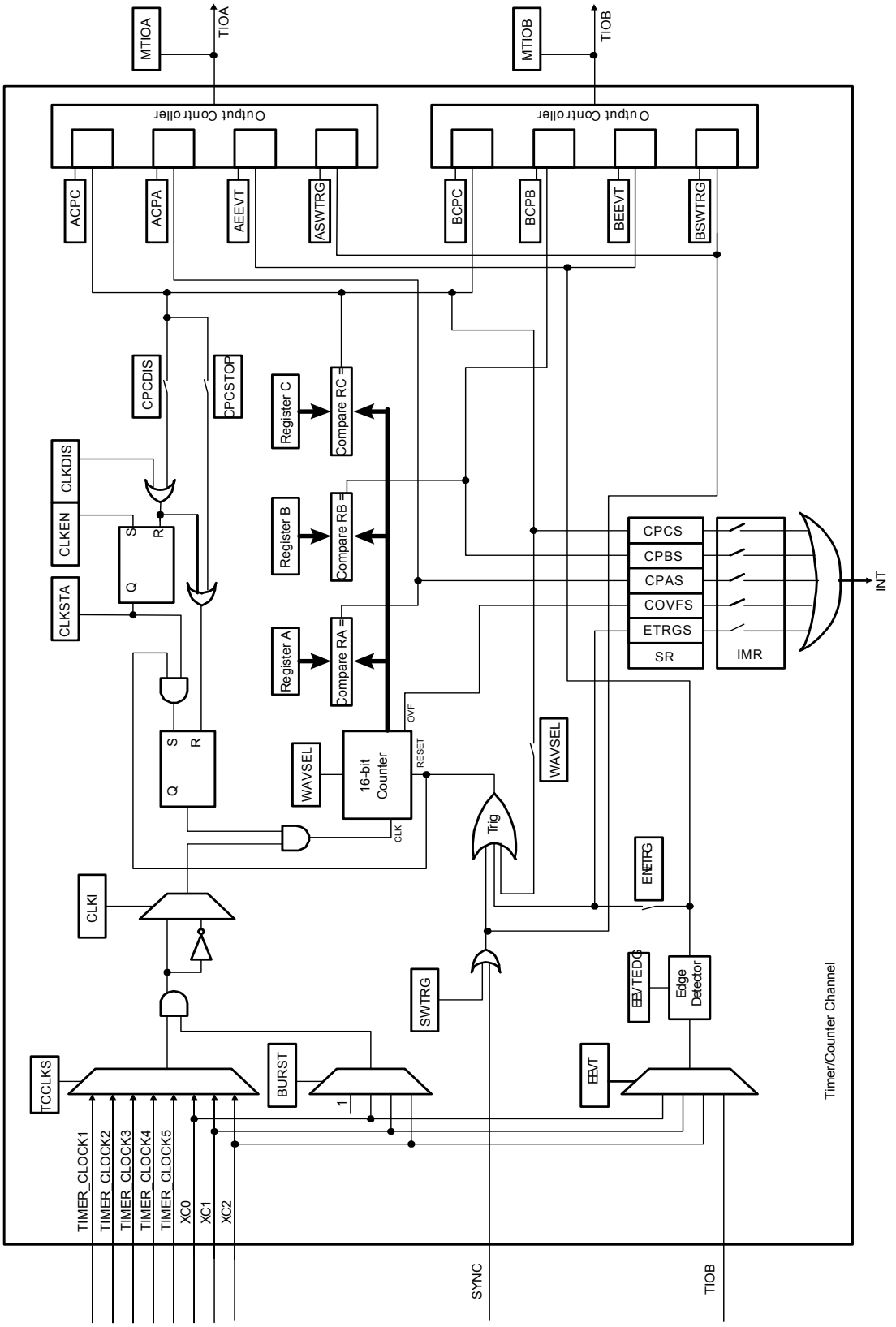
26.6.3.1 Waveform selection

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 26-5. Waveform Mode



26.6.3.2 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 26-6 on page 538.

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 26-7 on page 539.

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 26-6. WAVSEL= 0 Without Trigger

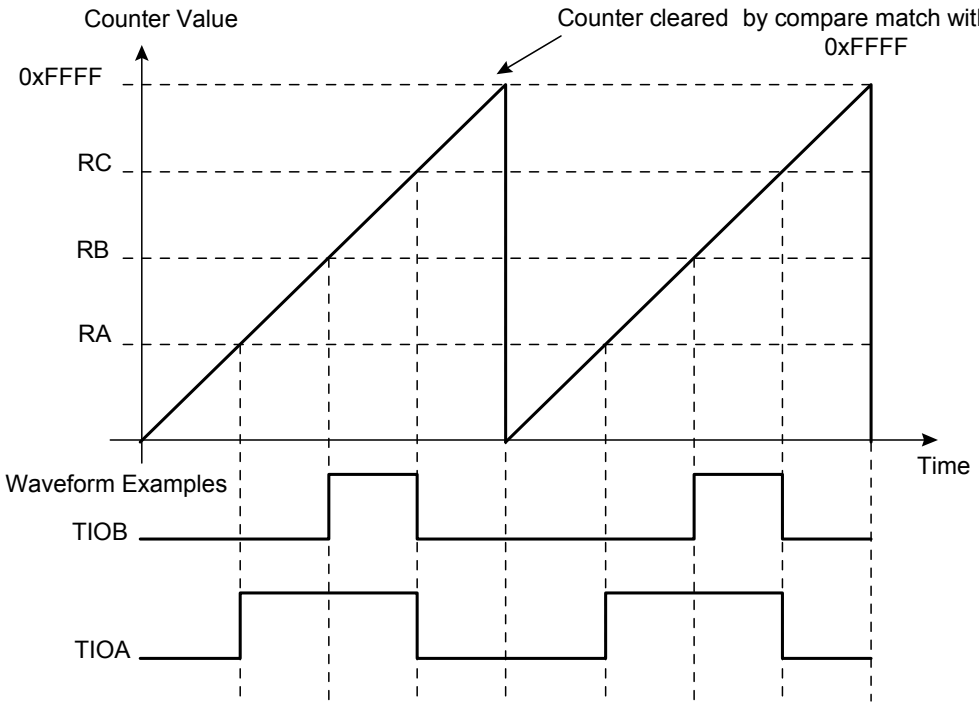
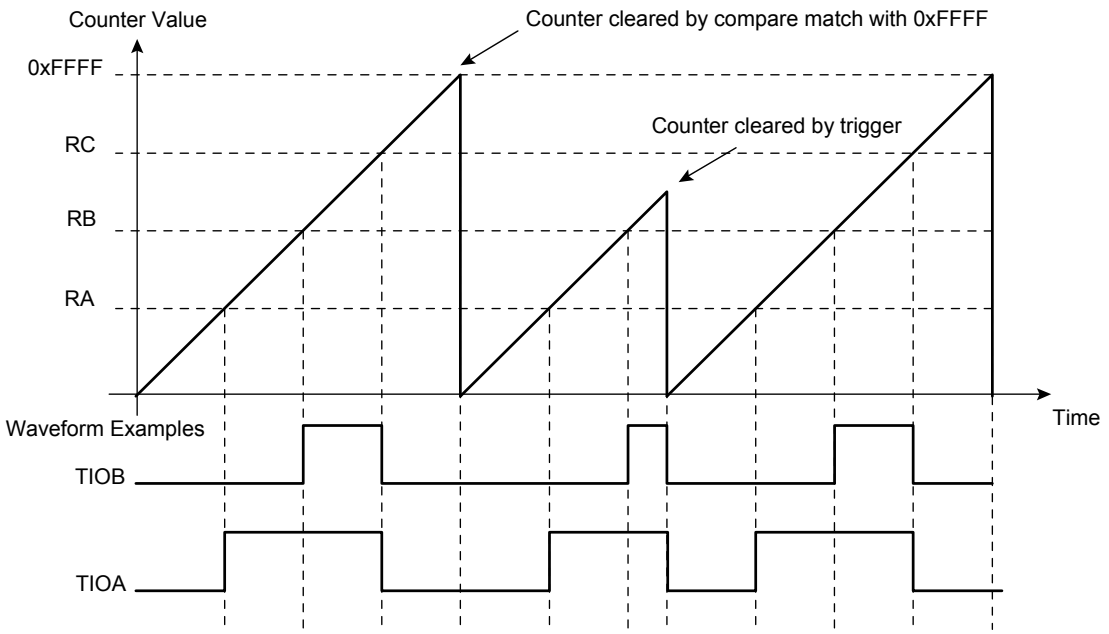


Figure 26-7. WAVSEL= 0 With Trigger



26.6.3.3 *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 26-8 on page 540.

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 26-9 on page 540.

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

Figure 26-8. WAVSEL = 2 Without Trigger

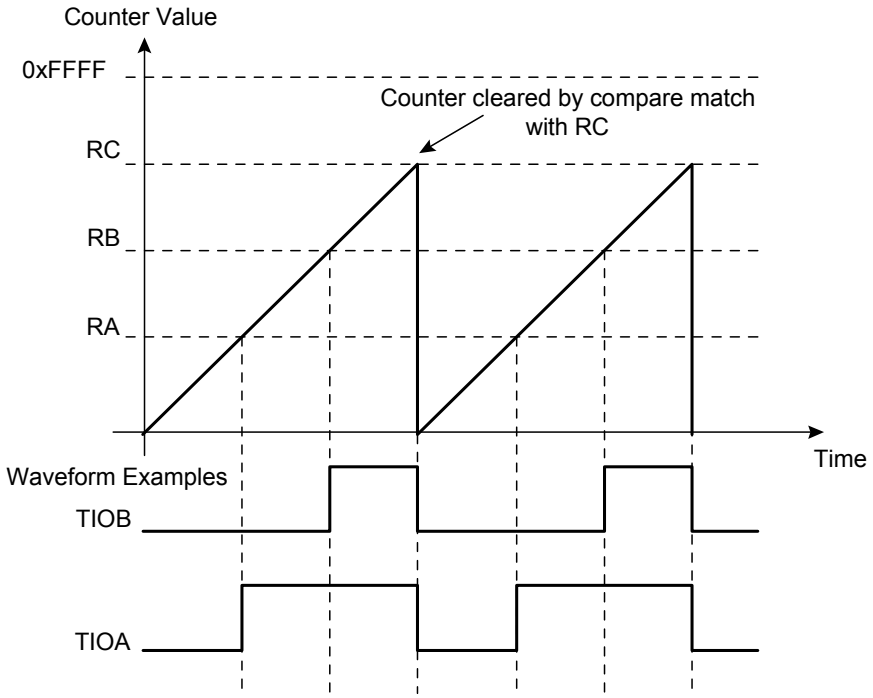
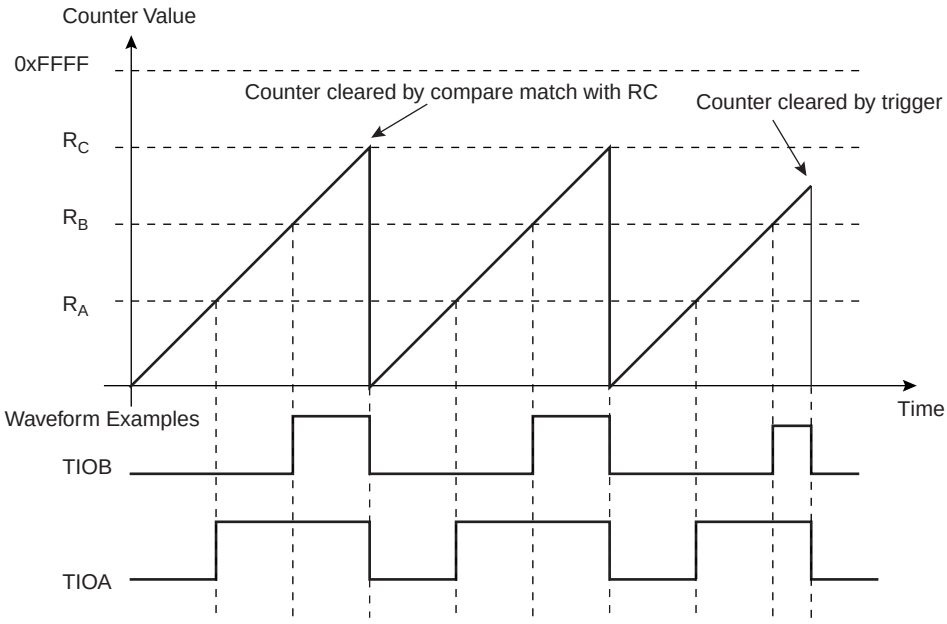


Figure 26-9. WAVSEL = 2 With Trigger



26.6.3.4 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 26-10 on page 541.

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 26-11 on page 541.

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 26-10. WAVSEL = 1 Without Trigger

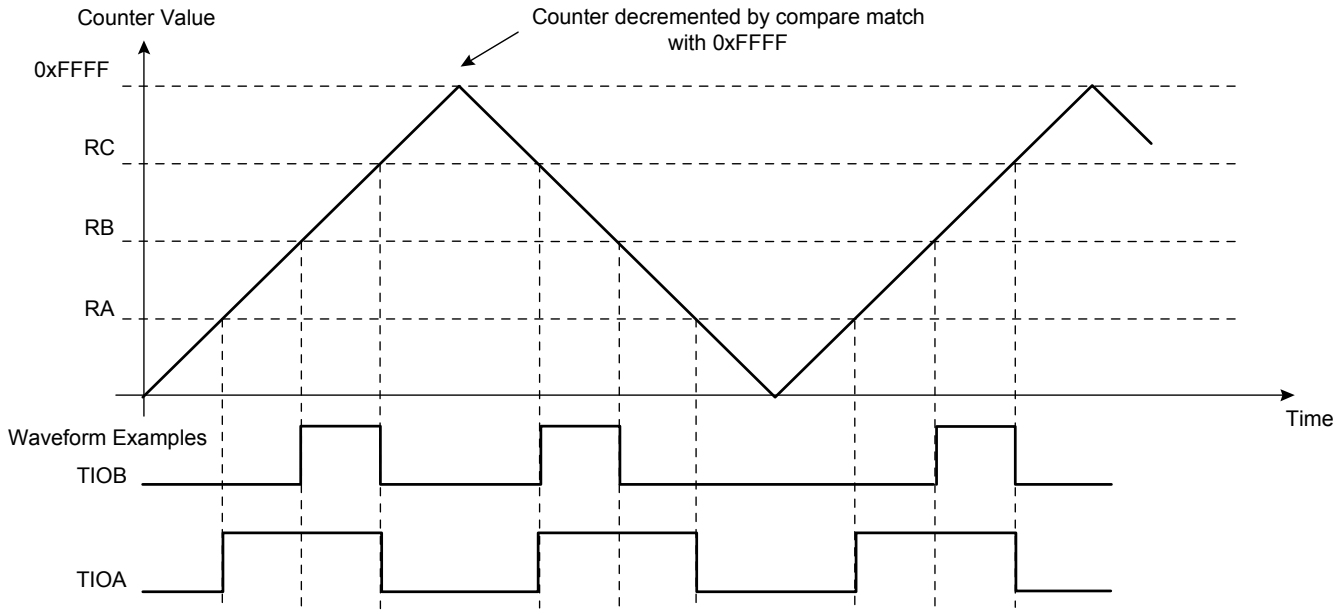
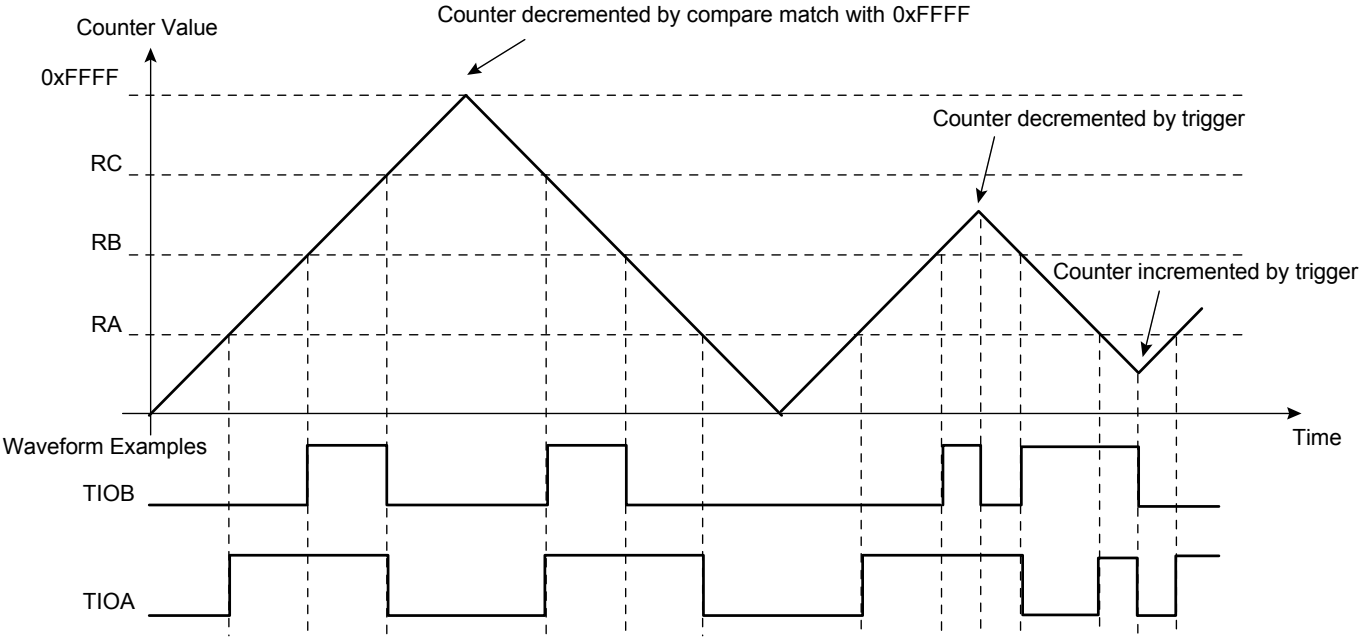


Figure 26-11. WAVSEL = 1 With Trigger



26.6.3.5 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 26-12 on page 542.

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 26-13 on page 543.

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

Figure 26-12. WAVSEL = 3 Without Trigger

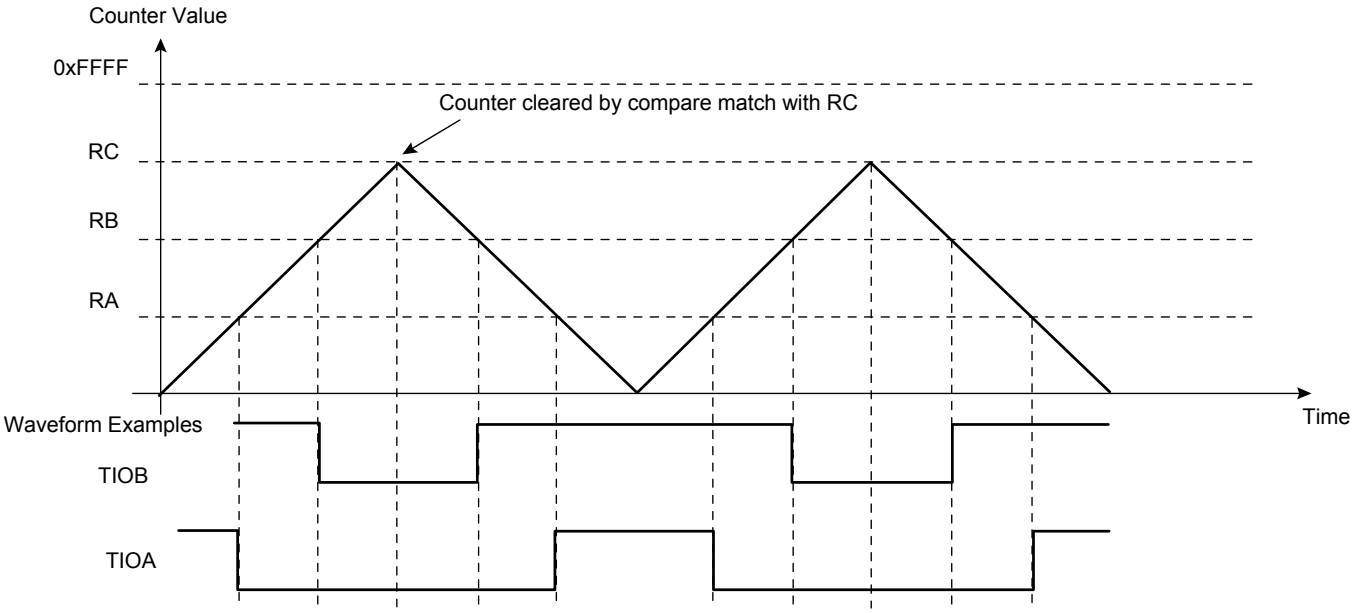
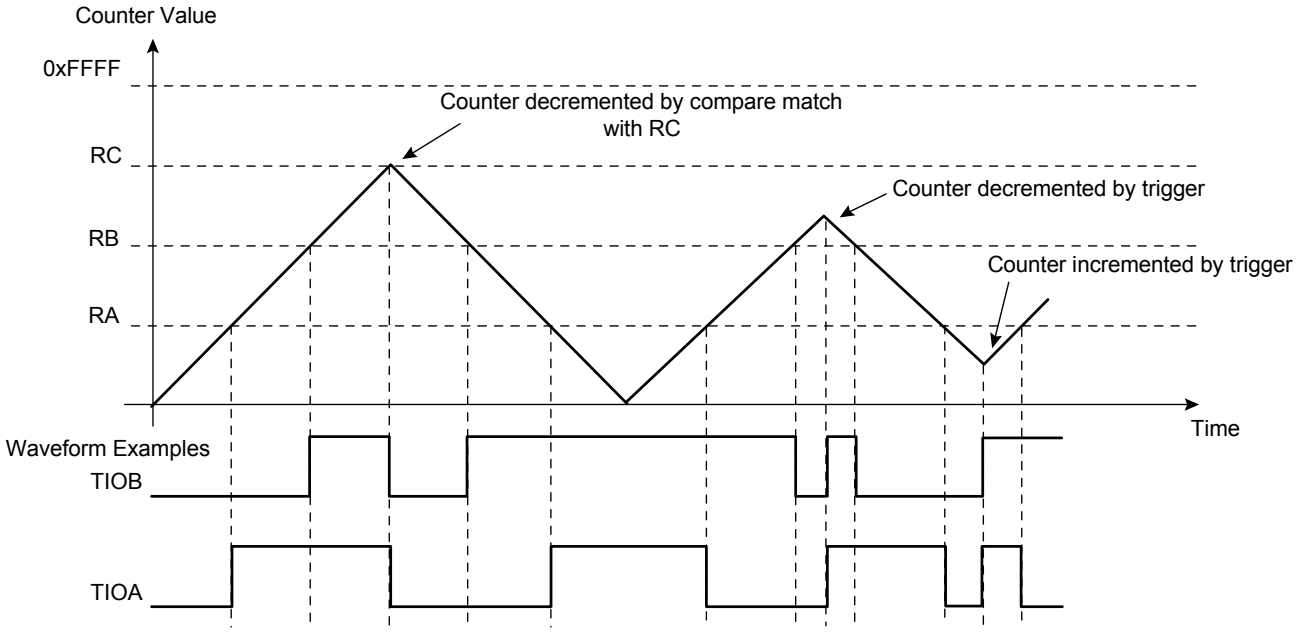


Figure 26-13. WAVSEL = 3 With Trigger



26.6.3.6 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.

26.6.3.7 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)

26.7 User Interface

Table 26-3. TC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Channel 0 Control Register	CCR0	Write-only	0x00000000
0x04	Channel 0 Mode Register	CMR0	Read/Write	0x00000000
0x10	Channel 0 Counter Value	CV0	Read-only	0x00000000
0x14	Channel 0 Register A	RA0	Read/Write(1)	0x00000000
0x18	Channel 0 Register B	RB0	Read/Write(1)	0x00000000
0x1C	Channel 0 Register C	RC0	Read/Write	0x00000000
0x20	Channel 0 Status Register	SR0	Read-only	0x00000000
0x24	Interrupt Enable Register	IER0	Write-only	0x00000000
0x28	Channel 0 Interrupt Disable Register	IDR0	Write-only	0x00000000
0x2C	Channel 0 Interrupt Mask Register	IMR0	Read-only	0x00000000
0x40	Channel 1 Control Register	CCR1	Write-only	0x00000000
0x44	Channel 1 Mode Register	CMR1	Read/Write	0x00000000
0x50	Channel 1 Counter Value	CV1	Read-only	0x00000000
0x54	Channel 1 Register A	RA1	Read/Write(1)	0x00000000
0x58	Channel 1 Register B	RB1	Read/Write(1)	0x00000000
0x5C	Channel 1 Register C	RC1	Read/Write	0x00000000
0x60	Channel 1 Status Register	SR1	Read-only	0x00000000
0x64	Channel 1 Interrupt Enable Register	IER1	Write-only	0x00000000
0x68	Channel 1 Interrupt Disable Register	IDR1	Write-only	0x00000000
0x6C	Channel 1 Interrupt Mask Register	IMR1	Read-only	0x00000000
0x80	Channel 2 Control Register	CCR2	Write-only	0x00000000
0x84	Channel 2 Mode Register	CMR2	Read/Write	0x00000000
0x90	Channel 2 Counter Value	CV2	Read-only	0x00000000
0x94	Channel 2 Register A	RA2	Read/Write(1)	0x00000000
0x98	Channel 2 Register B	RB2	Read/Write(1)	0x00000000
0x9C	Channel 2 Register C	RC2	Read/Write	0x00000000
0xA0	Channel 2 Status Register	SR2	Read-only	0x00000000
0xA4	Channel 2 Interrupt Enable Register	IER2	Write-only	0x00000000
0xA8	Channel 2 Interrupt Disable Register	IDR2	Write-only	0x00000000
0xAC	Channel 2 Interrupt Mask Register	IMR2	Read-only	0x00000000
0xC0	Block Control Register	BCR	Write-only	0x00000000
0xC4	Block Mode Register	BMR	Read/Write	0x00000000
0xF8	Features Register	FEATURES	Read-only	-(2)
0xFC	Version Register	VERSION	Read-only	-(2)

Notes:
1. Read-only if CMRn.WAVE is zero.
2. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

26.7.1 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.

26.7.2 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

26.7.3 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

26.7.4 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.

26.7.5 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.

26.7.6 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.

26.7.7 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.

26.7.8 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.

26.7.9 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.

26.7.10 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.

26.7.11 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.

26.7.12 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.

26.7.13 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

26.7.14 Features Register

Name: FEATURES

Access Type: Read-only

Offset: 0xF8

Reset Value: -

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

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

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

7	6	5	4	3	2	1	0
CTRSIZE							

- **BRPBHSB: Bridge type is PB to HSB**
  1: Bridge type is PB to HSB.
  0: Bridge type is not PB to HSB.
- **UPDNIMPL: Up/down is implemented**
  1: Up/down counter capability is implemented.
  0: Up/down counter capability is not implemented.
- **CTRSIZE: Counter size**
  This field indicates the size of the counter in bits.

26.7.15 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0xFC

Reset Value: -

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

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

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

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

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

26.8 Module Configuration

26.8.1 Clock Connections

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

Table 26-4. Timer/Counter Clock Connections

Module	Source	Name	Connection
TC	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	
		XC1	
		XC2	

Table 26-5. Register Reset Values

Register	Reset Value
VERSION	0x00000223
FEATURES	0x00000310

27. Capacitive Touch Module (CAT)

Rev: 4.0.0.0

27.1 Features

- **QTouch® method allows N touch sensors to be implemented using 2N physical pins**
- **One autonomous QTouch sensor operates without DMA or CPU intervention**
- **All QTouch sensors can operate in DMA-driven mode without CPU intervention**
- **External synchronization to reduce 50 or 60 Hz mains interference**
- **Spread spectrum sensor drive capability**

27.2 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.

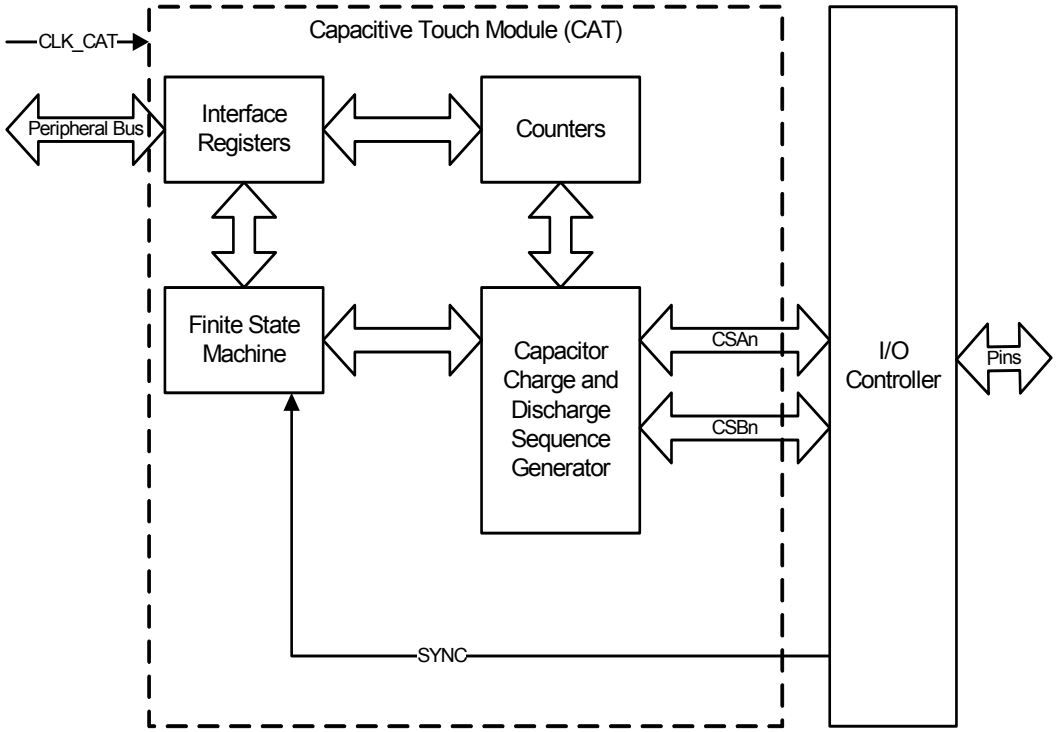
The module implements the QTouch method of capturing signals from capacitive touch sensors. The QTouch method is generally suitable for small numbers of sensors since it requires 2 physical pins per sensor.

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.

All of the QTouch sensors can operate in a DMA-driven mode, known as DMATouch, that allows detection of touch without CPU intervention. The module also implements one autonomous QTouch sensor that is capable of detecting touch without DMA or CPU intervention. This allows proximity or activation detection in low-power sleep modes.

27.3 Block Diagram

Figure 27-1. CAT Block Diagram



27.4 I/O Lines Description

Table 27-1. I/O Lines Description

Name	Description	Type
CSAn	Capacitive sense A line n	I/O
CSBn	Capacitive sense B line n	I/O
SYNC	Synchronize signal	Input

27.5 Product Dependencies

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

27.5.1 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.

Table 27-2. Pin Selection Guide

CAT Module Pin Name	QTouch Method Pin Name		Selection Bit in PINMODEx Register
CSA0	SNS0		SP0
CSB0	SNSK0		SP0
CSA1	SNS1		SP1
CSB1	SNSK1		SP1
CSA2	SNS2		SP2
CSB2	SNSK2		SP2
CSA3	SNS3		SP3
CSB3	SNSK3		SP3
CSA4	SNS4		SP4
CSB4	SNSK4		SP4
CSA5	SNS5		SP5
CSB5	SNSK5		SP5
CSA6	SNS6		SP6
CSB6	SNSK6		SP6
CSA7	SNS7		SP7
CSB7	SNSK7		SP7
CSA8	SNS8		SP8
CSB8	SNSK8		SP8
CSA9	SNS9		SP9
CSB9	SNSK9		SP9
CSA10	SNS10		SP10
CSB10	SNSK10		SP10
CSA11	SNS11		SP11
CSB11	SNSK11		SP11
CSA12	SNS12		SP12
CSB12	SNSK12		SP12
CSA13	SNS13		SP13
CSB13	SNSK13		SP13
CSA14	SNS14		SP14
CSB14	SNSK14		SP14
CSA15	SNS15		SP15
CSB15	SNSK15		SP15
CSA16	SNS16		SP16
CSB16	SNSK16		SP16

27.5.2 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.

27.5.3 Interrupts

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

27.5.4 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.Two additional handshakes support DMATouch by regulating transfers from memory to the DMATouch State Write Register (DMATSW) and from the DMATouch State Read Register (DMATSR) to memory. The Peripheral DMA Controller must be configured properly and enabled in order to perform direct memory access transfers to/from the CAT module.

27.5.5 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.

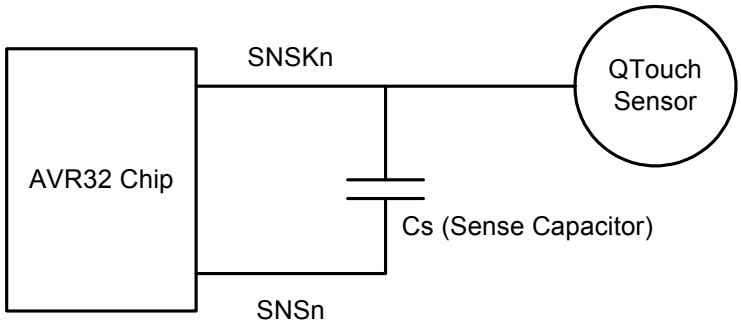
27.6 Functional Description

27.6.1 Acquisition Types

The CAT module can perform several types of QTouch acquisition from capacitive touch sensors: autonomous QTouch (one sensor only), DMATouch, QTouch group A, and QTouch group B. 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 sensors:

Figure 27-2. CAT Touch 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/DMA 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 DMATouch capability, it is first necessary to set up the pin mode registers (PINMODE0, PINMODE1, and PINMODE2) so that the desired pins are specified as DMA-Touch. The Autonomous/DMA Touch Configuration Registers (ATCFG0 through 3) must also be configured with appropriate values. One channel of the Peripheral DMA Controller must be set up to transfer state words from a block of memory to the DMATSW register, and another channel must be set up to transfer state words from the DMATSR register back to the same block of memory. The module can then be enabled using the CTRL register. After the module is enabled, the module will acquire count values from each DMATouch sensor. Once the module has acquired a count value for a sensor, it will use a handshake interface to signal the Peripheral DMA controller to transfer a state word to the DMATSW register. The module will use the count value to update the state word, and then the updated state word will be transferred to the DMATSR register. Another handshake interface will signal the Peripheral DMA controller to transfer the contents of the DMATSR register back to memory. The status of the DMATouch sensors can be determined at any time by reading the DMATouch Sensor Status Register (DMATSS).

In order to use the QTouch group A, or QTouch group B acquisition capabilities, it is first necessary to set up the pin mode registers (PINMODE0, PINMODE1, and PINMODE2) and configuration registers (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: QTouch group A, or QTouch group B. The CPU can initiate acquisition by writing to the AISR.

While 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 ACOUNT to memory, the Peripheral DMA Controller can perform long acquisition sequences and store results in memory without CPU intervention.

27.6.2 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.

Each QTouch prescaler is controlled by the DIV field in the appropriate Configuration Register 0 (ATCFG0, TGACFG0, or TGBCFG0). 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.

27.6.3 Capacitive Count Acquisition

For the 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.

27.6.4 Autonomous QTouch and DMATouch

For autonomous QTouch and DMATouch, a complete detection algorithm is implemented within the CAT module. The additional parameters needed to control the detection algorithm must be specified by the user in the ATCFG2 and ATCFG3 registers.

Autonomous QTouch and DMATouch 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.

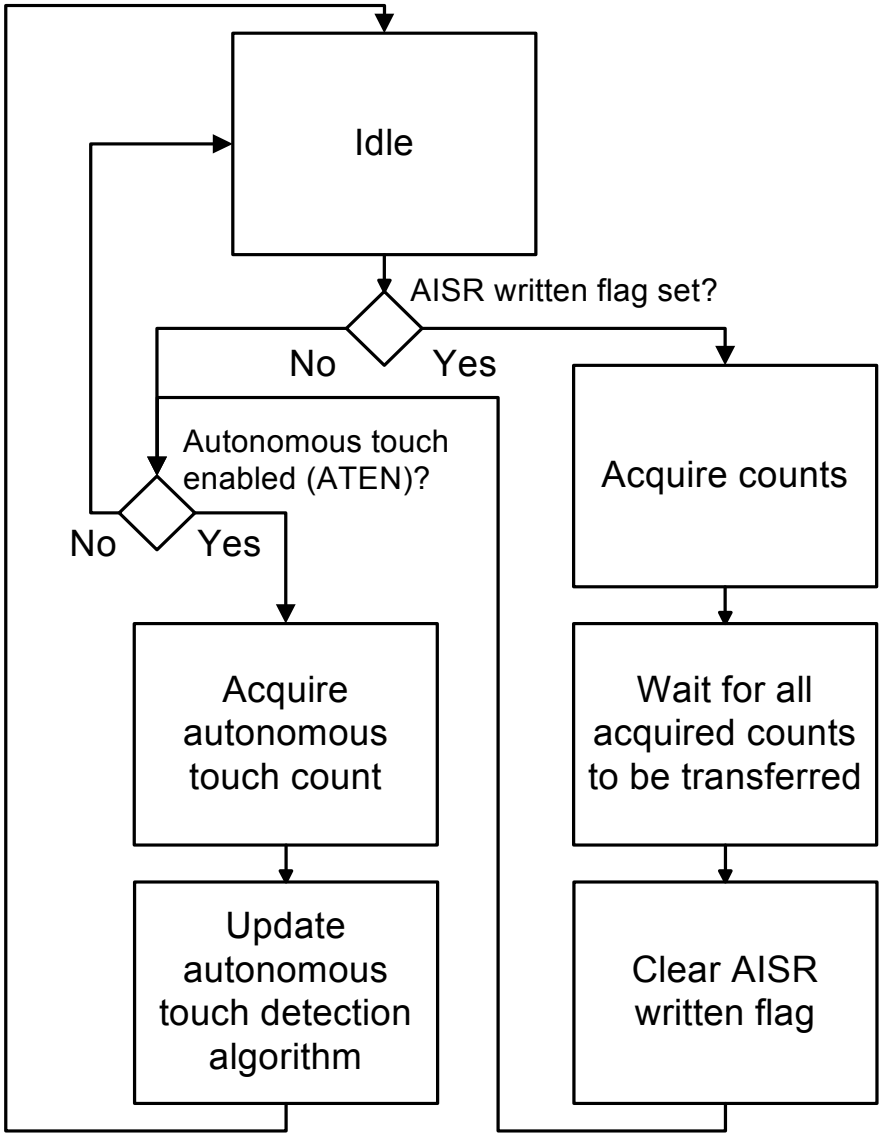
To avoid false positives a detect integration filtering technique can be used. The number of successive detects required is specified in the FILTER field of the ATCFG2 register.

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 and DMATouch sensors 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.

Figure 27-3. CAT Acquisition and Processing Sequence



27.6.5 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 (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.

27.6.6 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 (ATCFG1, TGACFG1, or TGBCFG1).

27.6.7 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.

27.7 User Interface

Table 27-3. 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/DMA Touch Configuration Register 0	ATCFG0	Read/Write	0x00000000
0x14	Autonomous/DMA Touch Configuration Register 1	ATCFG1	Read/Write	0x00000000
0x18	Autonomous/DMA Touch Configuration Register 2	ATCFG2	Read/Write	0x00000000
0x1C	Autonomous/DMA 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
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
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
0x74	Pin Mode Register 2	PINMODE2	Read/Write	0x00000000
0x78	DMATouch State Write Register	DMATSW	Write-only	0x00000000
0x7C	DMATouch State Read Register	DMATSR	Read-only	0x00000000
0xA0	DMATouch Sensor Status Register	DMATSS	Read-only	0x00000000
0xF8	Parameter Register	PARAMETER	Read-only	-(1)
0xFC	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.

27.7.1 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.

27.7.2 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).

27.7.3 Pin Mode Registers 0, 1, and 2

Name: PINMODE0, PINMODE1, and PINMODE2

Access Type: Read/Write

Offset: 0x08, 0x0C, 0x74

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 (PINMODE2.SP[n] PINMODE1.SP[n] PINMODE0.SP[n]) bits have the following definitions:

  000: Sense pair n disabled.

  001: Sense pair n is assigned to QTouch Group A.

  010: Sense pair n is assigned to QTouch Group B.

  011:Reserved.

  100: Sense pair n is assigned to the DMATouch Group.

  101: Reserved.

  110: Reserved.

  111: Reserved.

27.7.4 Autonomous/DMA 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 and DMATouch sensors, specifies how many sample clock cycles should be used for transferring charge to the sense capacitor.
- **SELEN: Settle Length**
  For the autonomous QTouch sensor and DMATouch sensors, specifies how many sample clock cycles should be used for settling after charge transfer.

27.7.5 Autonomous/DMA 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 and DMATouch sensors, 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 and DMATouch sensors, 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 and DMATouch sensors, specifies that spread spectrum sensor drive shall be used.
- **DILEN: Discharge Length**
  For the autonomous QTouch sensor and DMATouch sensors, 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 and DMATouch sensors, specifies how many counts the maximum acquisition should be.

27.7.6 Autonomous/DMA Touch Configuration Register 2

Name: ATCFG2

Access Type: Read/Write

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
-				FILTER			

15	14	13	12	11	10	9	8
OUTSENS							

7	6	5	4	3	2	1	0
SENSE							

- **FILTER: Autonomous Touch Filter Setting**
  For the autonomous QTouch sensor and DMATouch sensors, specifies how many positive detects in a row the CAT needs to have on the 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 and DMATouch sensors, specifies how sensitive the out-of-touch detector should be.
- **SENSE: Sensitivity**
  For the autonomous QTouch sensor and DMATouch sensors, specifies how sensitive the touch detector should be.

27.7.7 Autonomous/DMA 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 and DMATouch sensors, 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 and DMATouch sensors, 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 and DMATouch sensors, 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.

27.7.8 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.

27.7.9 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.

27.7.10 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
DMATSC	-	-	-	-	-	DMATSR	DMATSW
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ACQDONE	ACREADY
7	6	5	4	3	2	1	0
-	-	-	MBLREQ	ATSTATE	ATSC	ATCAL	ENABLED

- **DMATSC: DMATouch Sensor State Change**
  0: No change in the DMATSS register.
  1: One or more bits have changed in the DMATSS register.
- **DMATSR: DMATouch State Read Register Ready**
  0: A new state word is not available in the DMATSR register.
  1: A new state word is available in the DMATSR register.
- **DMATSW: DMATouch State Write Register Request**
  0: The DMATouch algorithm is not requesting that a state word be written to the DMATSW register.
  1: The DMATouch algorithm is requesting that a state word be written to the DMATSW register.
- **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.

27.7.11 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
DMATSC	-	-	-	-	-	-	-
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.

27.7.12 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
DMATSC	-	-	-	-	-	-	-
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.

27.7.13 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
DMATSC	-	-	-	-	-	-	-
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.

27.7.14 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
DMATSC	-	-	-	-	-	-	-
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.

27.7.15 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: Undefined behavior.

  11: Undefined behavior.

  A read of this register will return the value that was previously written.

27.7.16 Acquired Count Register

Name: ACOUNT

Access Type: Read-only

Offset: 0x54

Reset Value: 0x00000000

31	30	29	28	27	26	25	24

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]							

- **SPORX: Sensor pair index**
  The sensor pair index (for QTouch 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.

27.7.17 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.

27.7.18 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.

27.7.19 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.

27.7.20 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.

27.7.21 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.

27.7.22 DMATouch State Write Register

Name: DMATSW

Access Type: Write-only

Offset: 0x78

Reset Value: 0x00000000

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

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

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

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

- **NOTINCAL: Not in Calibration Mode**
  0: Calibration should be performed on the next iteration of the DMATouch algorithm.
  1: Calibration should not be performed on the next iteration of the DMATouch algorithm.
- **DETCNT: Detection Count**
  This count value is updated and used by the DMATouch algorithm in order to detect when a button has been pushed.
- **BASECNT: Base Count**
  This count value represents the average expected acquired count when the sensor/button is not pushed.

27.7.23 DMA Touch State Read Register

Name: DMATSR

Access Type: Read/Write

Offset: 0x7C

Reset Value: 0x00000000

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

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

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

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

- **NOTINCAL: Not in Calibration Mode**
  0: Calibration should be performed on the next iteration of the DMATouch algorithm.
  1: Calibration should not be performed on the next iteration of the DMATouch algorithm.
- **DETCNT: Detection Count**
  This count value is updated and used by the DMATouch algorithm in order to detect when a button has been pushed.
- **BASECNT: Base Count**
  This count value represents the average expected acquired count when the sensor/button is not pushed.

27.7.24 DMATouch Sensor Status Register

Name: DMATSS

Access Type: Read-only

Offset: 0xA0

Reset Value: 0x00000000

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

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

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

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

- **SS: Sensor Status**
  0: The DMATouch sensor is not active, i.e. the button is currently not pushed.
  1: The DMATouch sensor is active, i.e. the button is currently pushed.

27.7.25 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0xF8

Reset Value: -

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

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.

27.7.26 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.

27.8 Module Configuration

The specific configuration of 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 27-4. Module Configuration

Feature	CAT
Number of touch sensors	Allows up to 25 buttons for 64 pins packages. Allows up to 17 buttons for 48 pins packages.

Table 27-5. Module clock name

Module name	Clock name
CAT	CLK_CAT

Table 27-6. Register Reset Values

Register	Reset Value
VERSION	0x00000400
PARAMETER	0x01FFFFFF

28. ADC Interface (ADCIFD)

Rev. 1.0.0.0

28.1 Feature

- **Multi-channel single-ended Analog-to-Digital Converter with up to 10-bit resolution**
- **Sequencer handling multiple conversions**
- **Numerous trigger sources**
  - **Software**
  - **Embedded 16-bit timer for periodic trigger**
  - **Continuous trigger**
  - **External trigger, rising, falling or any-edge trigger**
  - **Chip dependent Internal trigger**
- **Multiple sequencer modes:**
  - **Run the whole sequence on a start-of-conversion**
  - **Run a single conversion on a start-of-conversion**
- **ADC Power Reduction Mode for low power ADC applications**
- **Window monitor, with selectable channel**
- **Programmable ADC startup time**

28.2 Overview

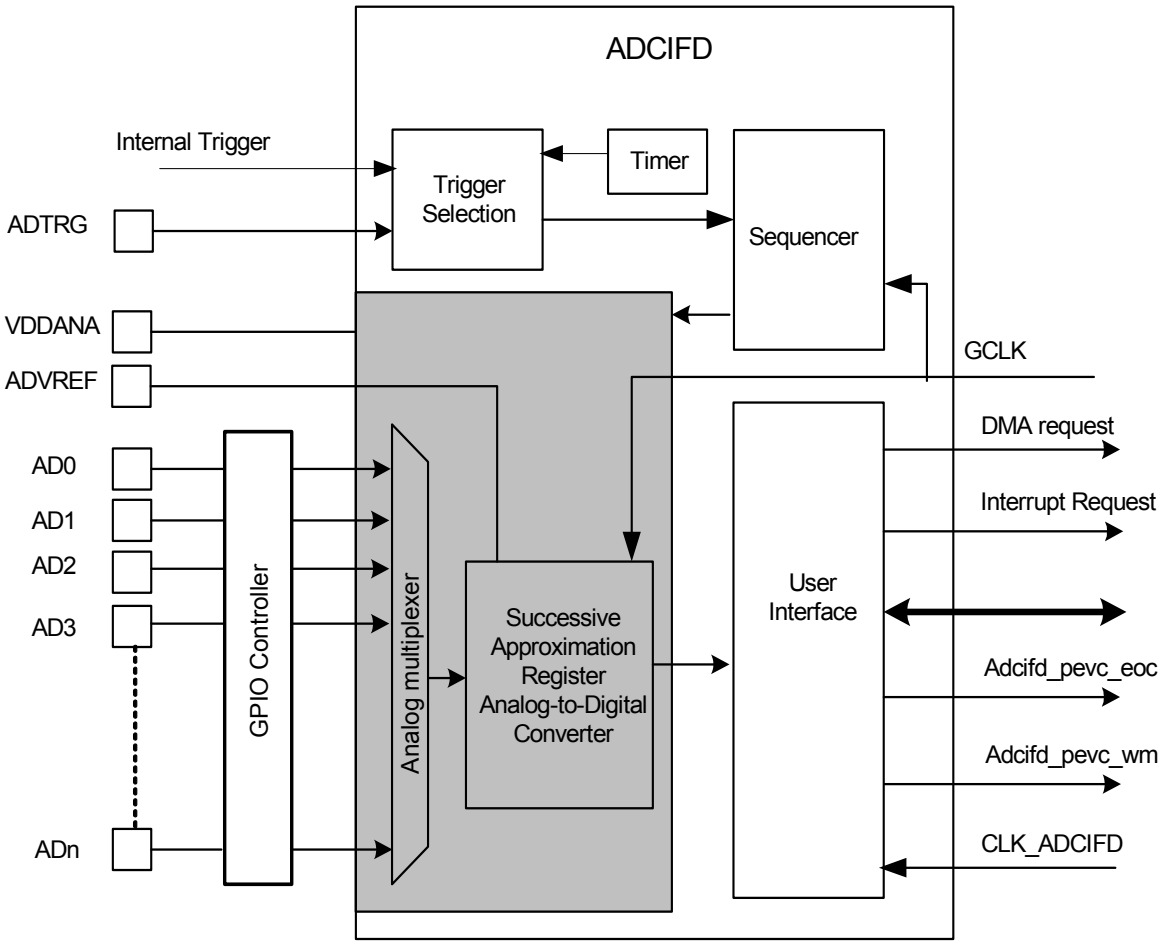
The ADC interface (ADCIFD) is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). It also integrates an analog multiplexer, making possible the analog-to-digital conversions of multiple analog lines. The conversions extend from 0V to ADVREF.

The ADC supports 8-bit and 10-bit resolution mode, and conversion results are reported in a common register for all channels. Conversions can be started for all enabled channels, either by a software trigger, by detection of a level change on the external trigger pin (ADTRG), or by an integrated programmable timer.

The ADCIFD also integrates an ADC Power Reduction Mode and a Window Monitor mode, and connects with one Peripheral DMA Controller channel. These features reduce both power consumption and processor intervention.

28.3 Block diagram

Figure 28-1. ADCIFD block diagram



28.4 I/O Lines Description

Table 28-1. I/O Lines decription table

Name	Description	Type
AD0-AD7	Analog input channels	Analog
ADVREF	Reference voltage	Analog
VDDANA	Analog power supply	Power
GNDANA	Analog ground	Power
ADTRG	External trigger	Digital

28.5 Product dependencies

28.5.1 I/O Lines

The pins used for interfacing the ADCIFD may be multiplexed with I/O Controller lines. The programmer must first program the I/O controller to assign the desired ADCIFD pins to their peripheral function. If I/O lines of the ADCIFD are not used by the application, they can be used for other purposes by the I/O controller.

Not all ADCIFD outputs may be enabled. If an application requires only four channels, then only four ADCIFD lines will be assigned to ADCIFD outputs.

28.5.2 Power management

If the CPU enters a power reduction mode that disables clocks used by the ADCIFD, the ADCIFD will stop functioning and resume operation after the system wakes up from power reduction mode. Before entering a power reduction mode where the clock to the ADCIFD is stopped, make sure the Analog-to-Digital Converter cell is put in an inactive state.

28.5.3 Clocks

The clock for the ADCIFD bus interface (CLK_ADCIFD) 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 ADCIFD before disabling the clock, to avoid freezing the ADCIFD in an undefined state. Additionally, the ADCIFD 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 ADCIFD can be used.

28.5.4 Interrupt controller

The ADCIFD interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the ADCIFD requires the Interrupt Controller to be programmed first.

Section 28.6.5).

28.5.5 Debug operation

When an external debugger forces the CPU into debug mode:

- the ADCIFD continues normal operation if the bit related to ADCIFD in PDBG register is ‘0’. PDC access continues normal operation and may interfere with debug operation.
- the ADCIFD is frozen if the bit related to ADCIFD in PDBG register is ‘1’. When the ADCIFD is frozen, ADCIFD PB registers can still be accessed. Then, reading registers may modify status bits such as LOVR like in normal operation. PDC access are pending.

28.6 Functional description

28.6.1 Initializing the ADCIFD

To initialize the module the user first needs to configure the ADCIFD clocks (please refer to Section 28.5.3). Then he needs to configure the Power Reduction Mode field (PRM) in the Configuration Register (CFG) and the STARTUP field in the Timing Configuration Register (TIM) (Please refer to Section 28.6.7). Then he can write a one to the Enable (EN) bit in the Control Register (CR). The user must check that ADCIFD has started correctly, firstly by checking that the Enable bit (EN) located in the Status Register (SR) is set. Secondly, If the Power Reduction Mode is off, he must wait for the startup-done bit (SUD) also located in the Status Register (SR)

to be set. If the power reduction mode is on, only SR.EN can tell if the ADCIFD is ready for operation since startup-time will be performed only when a sequencer trigger event occurs. Please note that all ADCIFD controls will be ignored until SR.EN goes to '1'.

Before the ADCIFD can be used, the I/O Controller must be configured correctly and the Reference Voltage (ADVREF) signal must be connected. Refer to I/O Controller section for details. Note that once configured, ADCIFD configuration registers should not be written during operation since they are permanently used by the ADCIFD. The user must ensure that ADCIFD is stopped during configuration unless he knows what he is doing.

28.6.2 Basic Operation

To convert analog values to digital values the user must first initialize the ADCIFD as described in Section 28.6.1. When the ADCIFD is initialized the sequencer must be configured by writing the Number of Conversions in the Sequence (CNVNB) in the Sequencer Configuration Register (SEQCFG) and by writing the index channels in the Channel Selection Per Low/High conversion registers, respectively CSPLC and CSPHC. Configuring channel N for a given conversion instructs the ADCIFD to convert the analog voltage applied to AD pin N. To start converting data the user can either manually start a conversion sequence by write a one to the sequencer trigger event (STRIG) bit in the Control Register (CR) or configure an automatic trigger to initiate the conversions. The automatic trigger can be configured to trig on many different conditions. Refer to Section 28.6.13 for details. The result of the conversions are stored in the Last Converted Value register (LCV) 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 LOVR bit in the Status Register (SR). To use an interrupt handler the user must enable the End Of Conversion (EOC) 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 Status Clear Register (SCR). To use a Peripheral DMA Controller channel the user must configure the Peripheral DMA Controller appropriately. The 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 LOVR bit in the SR will be set.

28.6.3 ADC resolution

The ADC supports 8-bit and 10-bit resolution. Resolution can be changed by writing the resolution field (RS) in the Sequencer Configuration register (SEQCFG). By default, after a reset, the resolution is set to 10-bit. Note that an external decoupling capacitor connected to ADVREF and GNDANA is mandatory to achieve maximum resolution.

28.6.4 ADC Sequencer operating modes

28.6.4.1 General

The ADC sequencer consists in a mult-conversion sequencer. A sequence consists in a set of conversions to perform successively. The maximum number of conversions is 16, the actual number of conversions is given by the CNVNB field in the SEQCFG register (SEQCFG.CNVNB). After a conversion, the digital value of the selected channel is stored in the Last Converted Value register (LCV). It is also possible to sample the same channel multiple times, allowing the user to perform "oversampling", which gives increased resolution over traditional single-sampled conversion results.

28.6.4.2 Sequencer Behavior on a STRIG event

Thanks to the STRIGB field in the Sequencer Configuration Register (SEQCFG), two different behaviors are possible:

- 0: All sequence conversions are performed on a STRIG event
- 1: The sequencer runs across the sequence conversion per conversion

28.6.4.3 Sequencer start/stop mode

Thanks to the SA bit in the SEQCFG register (SEQCFG.SA), the behavior of the sequencer at the end of a sequence can be changed.

- 0: The sequencer waits for software acknowledge (acknowledge is done by writing a 1 in the SEOS bit of the SCR register (SEQCFG.SEOS)).
- 1: The sequencer will restart automatically a new sequence on a new STRIG event. Results will be overwritten if not processed.

The LOVR bit in SR register (SR.LOVR) indicates that an overrun error occurred. This means that the LCV register is updated with a new conversion result but previous one has not been read . Events such as end-of-sequence or end-of-conversion can be catched by interrupt servicing or polling routines thanks to the SEOS and SEOC bits in the SR register (SR.SEOS and SR.SEOC).

28.6.5 ADC clock configuration

The ADC analog cell clock frequency (GCLK) should be programmed to provide an ADC clock frequency accordingly 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.

The ADC cell converts an input voltage in 10 GCLK periods and takes at least SHTIM+1 GCLK periods to sample.

Thus, the maximum achievable ADC sampling frequency is: $\frac{F(GCLK)}{10 + (SHTIM + 1)}$

28.6.6 Power Reduction Mode

The Power Reduction Mode maximizes power saving by automatically deactivating the Analog-to-Digital Converter cell when it is not being used for conversions. The Power Reduction Mode is enabled by writing a one to the Power Reduction Mode (PRM) bit in the Configuration register (CFG.PRM). When a trigger occurs while the Power Reduction 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 event.

Before entering power reduction mode the user must make sure the ADCIFD is idle and that the Analog-to-Digital Converter cell is inactive. To deactivate the Analog-to-Digital Converter cell the PRM bit in the ADC Configuration Register (CFG) must be written to one and the ADCIFD must be idle. To make sure the ADCIFD is idle, write a zero to the Trigger Selection (TRGSEL) field in the Sequencer Configuration Register (SEQCFG) and wait for the sequencer busy (SBUSY) 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 timing register (TIM) will apply on the next conversion.

28.6.7 Power-up and 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 timing register (TIM) to get correct conversion results. The TIM.STARTUP field expects the startup time to be represented as the number of GCLK cycles between 8 and 256 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 CFG.PRM is zero. If CFG.PRM 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.

28.6.8 Operation Start/Stop

To reset ADCIFD to its initial state, user can enable the ADCIFD after it was previously disabled thanks to the Enable bit EN in the Control register (CR.EN). Another way to reset ADCIFD is to write a one in the SWRST field of the Control Register (CR.SWRST). In both cases configuration registers won't be affected.

28.6.9 Sample and hold time

A minimal Sample and Hold Time is necessary for the ADCIFD 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 by default is one GCLK period and can be increased by programming the SHTIM field in the ADC timing register (TIM). A null value means that no additional GCLK period are waited to charge the input sampling capacitor, the maximum achievable additional GCLK period number is 15.

28.6.10 Analog reference

Please refer to the Electrical Characteristics chapter.

Please, note that it is recommended to insert a decoupling capacitor between ADVREF and GNDANA externally to achieve maximum precision.

28.6.11 Conversion range and sampling rates

The conversion voltage amplitude range is [0, ADVREF].

28.6.12 Conversion results

If the Half Word Left Adjust (HWLA) bit in the SEQCFG register is set, then the result will be left adjusted on the 16 lower bits of the LCV register. Otherwise, results will be right-adjusted.All

$$Code = \frac{V(AD)}{V(ADVREF)} \times 2^{SRES + HWLA(16 - SRES)}$$

conversion results are signed in two's complement representation. Extra bits depending on resolution and left adjust settings are padded with zeroes.

28.6.13 Sequencer trigger event (STRIG)

The sources must be configured through the TRGSEL field of the SEQCFG register (SEQCFG.TRGSEL). Selecting the event controller source allows any event controller source to generate a sequencer trigger event (STRIG). By configuring the continuous mode, STRIG will be generated continuously.

The ADC can serve a maximum of one STRIG every 10 GCLK periods. Extra STRIG will be ignored. User will be informed thanks to the Sequencer Missed Trigger Event (SMTRG) field of the SR register (SR.SMTRG). If the STRIG frequency provided by the event controller exceeds the ADC capability, the event controller will generate an underrun status.

28.6.14 Internal Timer

The ADCIFD embeds an internal timer used as a trigger source which can be configured by setting the ITMC field of the ITIMER register (ITIMER.ITMC).

*Internal Timer Trigger Period= (ITMC+1)*T(GCLK)*

Once set as a STRIG source, the internal timer has to be started by writing a '1' in the TSTART bit of the CR register (CR.TSTART). It can be stopped in the same way by writing a '1' in the TSTOP bit of the CR register (CR.TSTOP). The current status of the internal timer can be read from the Timer Busy field of the SR register (SR.TBUSY): 0 means stopped, 1 means running. In addition when the internal timer is running, if ITIMER.ITMC is written to change the internal timer timeout frequency, the internal counter is cleared to avoid rollover phenomena.

Note: It is possible to generate an internal timer event each GCLK period by writing 0x0 in ITIMER.ITMC and by selecting the internal timer as a STRIG source

28.6.15 Peripheral DMA Controller (PDC) capability

There is one PDC channel. The LCV register contains the last converted value of the sequencer according to the conversion result format. The LCV register is updated each time the sequencer ends a conversion. If the last converted value has not been read, there's an overrun, the LOVR bit in the SR register indicates that at least one overrun error occurred. The LOVR bit of the SR register is cleared by writing a '1' in the LOVR fields of the SCR register.

Note: PDC transfers are 16 bits wide.

28.6.16 Window monitor

The window monitor monitors ADC results and make the ADCIFD behave as an analog comparator. Configuration is done by writing appropriately the Window Configuration Register (WCFG) and the Window Thresholds Register (WTH). When writing a one in the Monitor Filter Mode bit in the WCFG register (WCFG.MFM) , conversions are filtered using its index in the sequence. Otherwise, no filtering is applied, monitoring is performed on every conversion. Index is given by writing the field Source in the WCFG register (WCFG.SRC). Supported modes are selected by writing the Window Mode field in the WCFG register, please refer to the Table 28-2 below.

Thresholds are given by writing the Low Threshold (LT) and High Threshold (HT) in WTH. Please note that the significant WTH.LT and WT.HT bits are given by the precision selected in the SEQCFG.SRES field. That means that if you are in 8-bit mode, only the 8 lower bits will be considered.

Table 28-2. Window Monitor Modes

WM field in WCFGy			Modes
0	0	0	No window mode (default)
0	0	1	Mode 1 : active when result > LT
0	1	0	Mode 2 : active when result < HT
0	1	1	Mode 3 : active when LT < result < HT
1	0	0	Mode 4 : active when (!(LT < result < HT))
1	0	1	reserved
1	1	0	reserved
1	1	1	reserved

Note: Comparisons are performed regardless with the SEQCFG.HWLA setting (half word left adjust).

28.6.17 Interrupts

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 Status Register (SR). To clear an active interrupt request, write a one to the corresponding bit in the Clear Register (CR).

The Status Register (SR) fields in common with IER/IDR/IMR show the status since the last write to the Interrupt Clear Register. Other SR fields show the status at the time being read.

Table 28-3. ADCIFD interrupt group

Line	Line Description	Related Status
0	Sequencer	Sequencer start of sequence (SSOS) Sequencer end of sequence (SEOS) Sequencer end of conversion (SEOC) Sequencer (last converted value) overrun (LOVR) Sequencer missed trigger event (SMTRG)
1	Timing	Start-up done Timer time-out
2	Window	Window monitor

28.6.18 Conversion Performances

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

28.7 User Interface

Table 28-4. ADCIFD Register Memory Map

Offset	Register	Name	Access	Reset State
0x0000	CR Register	CR	Write-Only	0x00000000
0x0004	CFG Register	CFG	Read/Write	0x00000000
0x0008	SR Register	SR	Read-Only	0x00000000
0x000C	SCR Register	SCR	Write-Only	0x00000000
0x0010	SSR Register	SSR	Write-Only	0x00000000
0x0014	SEQCFG Register	SEQCFG	Read/Write	0x00000000
0x0018	CSPHC Register	MSPHC	Read/Write	0x00000000
0x001C	CSPLC Register	MSPLC	Read/Write	0x00000000
0x0020	TIM Register	CKDIV	Read/Write	0x00000000
0x0024	ITIMER Register	ITIMER	Read/Write	0x00000000
0x0028	WCFG Register	WCFG	Read/Write	0x00000000
0x002C	WTH Register	WTH	Read/Write	0x00000000
0x0030	LCV Register	LCV	Read-Only	0x00000000
0x0034	IER Register	IER	Read/Write	0x00000000
0x0038	IDR Register	IDR	Read/Write	0x00000000
0x004C	IMR Register	IMR	Read/Write	0x00000000
0x0040	VERSION Register	VERSION	Read-Only	-(1)
0x0044	PARAMETER Register	PARAMETER	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.

28.7.1 Control Register

Name : CR

Access Type : Write-Only

Offset : 0x00

Reset Value : 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	-	-	-	TSTART	TSTOP	STRIG	SWRST

Writing a zero to any of those bits in this register has no effect.

- **DIS:ADCIFD disable**
  Writing a one to this bit disables the ADCIFD
  Reading this bit always returns 0

Note: Changes do not apply immediately, ADCIFD status can be checked by reading the EN field of the SR register

- **EN:ADCIFD enable**
  Writing a one to this bit enables the ADCIFD
  Reading this bit always returns 0

Note: Changes do not apply immediately, ADCIFD status can be checked by reading the EN field of the SR register

- **TSTART:Internal timer start bit**
  Writing a one to this bit starts the internal timer
  Reading this bit always returns 0

Note: The internal timer status can be read in the RUNT field of the SR register

- **TSTOP:Internal timer stop bit**
  Writing a one to this bit stops the internal timer
  Reading this bit always returns 0

Note: The internal timer status can be read in the RUNT field of the SR register

- **STRIG:Sequencer trigger**
  Writing a one to this bit generates a sequencer trigger event
  Reading this bit always returns 0

- **SWRST: Software reset**
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets the ADCIFD, simulating a hardware reset. Using that control ensures that ADCIFD internal features will return to their initial states. Configuration registers won't be affected.

28.7.2 Configuration Register

Name : CFG

Access Type : Read/Write

Offset : 0x04

Reset Value : 0x00000000

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

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

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

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

- **PRM: Power reduction mode**
  1: Power reduction mode
  0: Normal mode

Note: When enabled, start-up time is required before each new conversion

28.7.3 Status Register

Name : SR

Access Type : Read-Only

Offset : 0x08

Reset Value : 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	CBUSY	SBUSY	TBUSY	EN

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

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

7	6	5	4	3	2	1	0
TTO	SUTD	SMTRG	WM	LOVR	SEOC	SEOS	SSOS

- **CBUSY: Conversion busy**
  1: ADCIFD is converting
  0: ADCIFD is not converting
- **SBUSY: Sequencer busy**
  1: ADCIFD sequencer is running
  0: ADCIFD sequencer is ready
- **TBUSY: Timer busy**
  1: ADCIFD internal timer is running
  0: ADCIFD internal timer is stopped
- **EN: Enable Status**
  1: ADCIFD is ready for operation
  0: ADCIFD is not ready
- **CSCNV:Current Sequencer Conversion**
  This field is set to the sequencer current conversion identifier
- **TTO: Timer time-out**
  This bit is set when the internal timer times out
  This bit is cleared when the corresponding bit in SCR is written to one
- **SUTD:Start-up time done**
  This bit is set when a start-up done event occurs
  This bit is cleared when the corresponding bit in SCR is written to one
- **SMTRG:Sequencer missed trigger event**
  This bit is set when a sequencer trigger event is missed
  This bit is cleared when the corresponding bit in SCR is written to one
- **WM:Window monitor**
  This bit is set when the watched result value goes to the defined window

This bit is cleared when the corresponding bit in SCR is written to one

- **LOVR:Sequencer last converted value overrun**
  This bit is set when an overrun error occurs on the LCV register
  This bit is cleared when the corresponding bit in SCR is written to one
- **SEOC:Sequencer end of conversion**
  This bit is set when an end of conversion occurs
  This bit is cleared when the corresponding bit in SCR is written to one
- **SEOS:Sequencer end of sequence**
  This bit is set when an end of sequence occurs
  This bit is cleared when the corresponding bit in SCR is written to one
- **SSOS:Sequencer start of sequence**
  This bit is set when a start of sequence occurs
  This bit is cleared when the corresponding bit in SCR is written to one

28.7.4 Status Clear Register

Name : SCR

Access Type : Write-Only

Offset : 0x0C

Reset Value : 0x00000000

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

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

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

7	6	5	4	3	2	1	0
TTO	SUTD	SMTRG	WM	LOVR	SEOC	SEOS	SSOS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit clears the corresponding SR bit

28.7.5 Status Set Register

Name : SSR

Access Type : Write-Only

Offset : 0x10

Reset Value : 0x00000000

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

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

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

7	6	5	4	3	2	1	0
TTO	SUTD	SMTRG	WM	LOVR	SEOC	SEOS	SSOS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit sets the corresponding SR bit

28.7.6 Sequencer Configuration Register

Name : SEQCFG

Access Type : Read/Write

Offset : 0x14

Reset Value : 0x00000000

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

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

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

7	6	5	4	3	2	1	0
-	-	-	-	-	STRGM	HWLA	SA

- **CNVNB:Number of conversions in a sequence**

  The number of conversions to perform in the sequence is CNVNB+1

- **SRES:Resolution**

SRES	Resolution
0	10-bits
1	8-bits

- **TRGSEL:Trigger selection**

TRGSEL	Trigger
0	Software
1	internal ADC timer
2	internal trigger source (refer to module configuration section)
3	Continuous mode
4	External trigger pin rising edge
5	External trigger pin falling edge
6	External trigger pin both edges

- **STRGM: Sequencer trigger mode**

  1 : The sequencer runs across the sequence conversion per conversion

  0 : The sequencer runs the complete sequence when a trigger event occurs

- **HWLA:Half Word Left Adjust**
  1 : enables the HWLA mode
  0 : disables the HWLA mode
- **SA:Software Acknowledge**
  1 : enables the SA mode
  0 : disables the SA mode

28.7.7 Channel Selection Per Low Conversion

Name : CSPLC

Access Type : Read/Write

Offset : 0x18

Reset Value : 0x00000000

31	30	29	28	27	26	25	24
CNV7				CNV6			

23	22	21	20	19	18	17	16
CNV5				CNV4			

15	14	13	12	11	10	9	8
CNV3				CNV2			

7	6	5	4	3	2	1	0
CNV1				CNV0			

- **CNV7: Channel selection for the 8th sequencer conversion slot**
- **CNV6: Channel selection for the 7th sequencer conversion slot**
- **CNV5: Channel selection for the 6th sequencer conversion slot**
- **CNV4: Channel selection for the 5th sequencer conversion slot**
- **CNV3: Channel selection for the 4th sequencer conversion slot**
- **CNV2: Channel selection for the 3rd sequencer conversion slot**
- **CNV1: Channel selection for the 2nd sequencer conversion slot**
- **CNV0: Channel selection for the 1st sequencer conversion slot**

28.7.8 Channel Selection Per High Conversion

Name : CSPHC

Access Type : Read/Write

Offset : 0x1C

Reset Value : 0x00000000

31	30	29	28	27	26	25	24
CNV15				CNV14			

23	22	21	20	19	18	17	16
CNV13				CNV12			

15	14	13	12	11	10	9	8
CNV11				CNV10			

7	6	5	4	3	2	1	0
CNV9				CNV8			

- **CNV15: Channel selection for the 15th sequencer conversion slot**
- **CNV14: Channel selection for the 14th sequencer conversion slot**
- **CNV13: Channel selection for the 13th sequencer conversion slot**
- **CNV12: Channel selection for the 12th sequencer conversion slot**
- **CNV11: Channel selection for the 11th sequencer conversion slot**
- **CNV10: Channel selection for the 10th sequencer conversion slot**
- **CNV9: Channel selection for the 9th sequencer conversion slot**
- **CNV8: Channel selection for the 8th sequencer conversion slot**

28.7.9 Timing Configuration Register

Name : TIM

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
-	-	-	-	SHTIM			

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

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

- **SHTIM: Sample and hold time**
  Sample and hold time in number of GCLK clock cycles : SHTIM+1
- **STARTUP: Startup time**
  Number of GCLK clock cycles to wait for : (STARTUP+1)*8

28.7.10 Internal timer register

Name : ITIMER

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
ITMC							

7	6	5	4	3	2	1	0
ITMC							

- **ITMC:Internal Timer Max Counter**

  f(itimer_timeout)=f(GCLK)/(ITMC+1)

28.7.11 Window monitor configuration

Name : WCFG

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
-	-	-	-	-	-	-	MFM

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

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

- **MFM: Monitor Filter Mode**
  Writing a zero to this field allows the window monitor to monitor all conversions
  Writing a one to this field allows the Window Monitor to filter conversions by index thanks to WCFG.SRC
- **SRC: Source**
  Index of the conversion result to monitor
- **WM: Window Monitor Mode**

WM	Window monitor mode
0	OFF
1	Mode 1 : RES(SRC) > LT
2	Mode 2 : RES(SRC) < HT
3	Mode 3 : LT<RES(SRC)<HT
4	Mode 4 : (LT>=RES(SRC)) || (RES(SRC)>=HT)
5	Reserved
6	Reserved
7	Reserved

28.7.12 Window Monitor Threshold Configuration

Name : WTH

Access Type : Read/Write

Offset : 0x2C

Reset Value : 0x00000000

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

23	22	21	20	19	18	17	16
HT							

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

7	6	5	4	3	2	1	0
LT							

- **HT:High Threshold**
- **LT:Low Threshold**

28.7.13 Sequencer last converted value

Name : LCV

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
LCV							

7	6	5	4	3	2	1	0
LCV							

- **LCV:Last converted value**

  This field is set by hardware to the last sequencer converted value depending on precision and on the choosen left adjustment mode (SEQCFG.HWLA).

28.7.14 Interrupt enable register

Name : IER

Access Type : Write-Only

Offset : 0x34

Reset Value : 0x00000000

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

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

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

7	6	5	4	3	2	1	0
TTO	SUTD	SMTRG	WM	LOVR	SEOC	SEOS	SSOS

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.

28.7.15 Interrupt disable register

Name : IDR

Access Type : Write-Only

Offset : 0x38

Reset Value : 0x00000000

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

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

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

7	6	5	4	3	2	1	0
TTO	SUTD	SMTRG	WM	LOVR	SEOC	SEOS	SSOS

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.

28.7.16 Interrupt mask register

Name : IMR

Access Type : Read-Only

Offset : 0x3C

Reset Value : 0x00000000

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

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

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

7	6	5	4	3	2	1	0
TTO	SUTD	SMTRG	WM	LOVR	SEOC	SEOS	SSOS

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.

28.7.17 Module Version

Name : VERSION

Access Type : Read-Only

Offset : 0x40

Reset Value : -

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

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

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

7	6	5	4	3	2	1	0
VERSION							

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

28.7.18 Parameter Register

Name : PARAMETER

Access Type : Read-Only

Offset : 0x44

Reset Value : -

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

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

15	14	13	12	11	10	9	8
N							

7	6	5	4	3	2	1	0
M							

- **N: Number of channels**
- **M: Number of sequencer states**

28.8Module configuration

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

Table 28-5. Module configuration

Feature	Connected to
Internal Trigger	TC0, output A0

Table 28-6. ADCIFD Clocks

Clock Name	Description
CLK_ADCIFD	Clock for the ADCIFD bus interface
GCLK_ADCIFD	Conversion clock. The generic clock used for the ADCIFD is GCLK8

Table 28-7. Register Reset Values

Register	Reset Value
VERSION	0x0000 0100
PARAMETER	0x0000 0808

29. Glue Logic Controller (GLOC)

Rev: 1.0.1.0

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

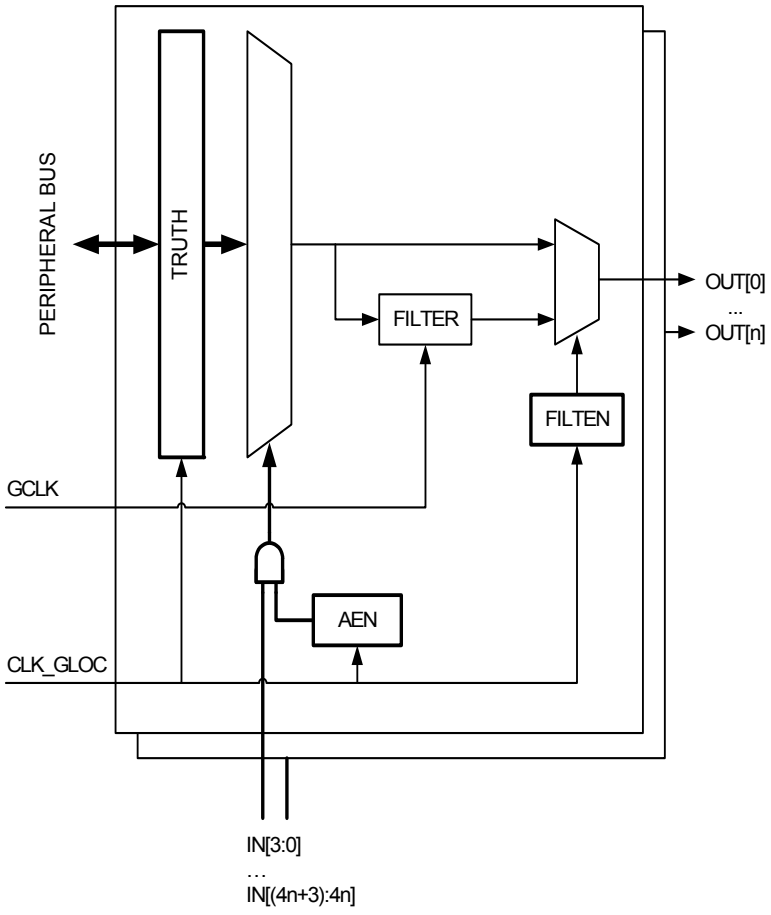
The Glue Logic Controller (GLOC) contains programmable logic which can be connected to the device pins. This allows the user to eliminate logic gates for simple glue logic functions on the PCB.

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 inputs and outputs actually in use. Pullups for pins configured to be used by the GLOC will be disabled.

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 Control Register (CR).

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 corresponding 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 CRn. 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	Control Register n	CRn	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 Control Register n

Name: CRn

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	4

Table 29-5. GLOC Clock Name

Clock Name	Description
CLK_GLOC	Clock for the GLOC bus interface
GCLK_GLOC	Generic clock used for the output filter feature. The generic clock used for the GLOC is GCLK0

Table 29-6. Register Reset Values

Register	Reset Value
VERSION	0x00000101
PARAMETER	0x00000004

30. aWire UART (AW)

Rev: 2.3.0.0

30.1 Features

- **Asynchronous receiver or transmitter when the aWire system is not used for debugging.**
- **One- or two-pin operation supported.**

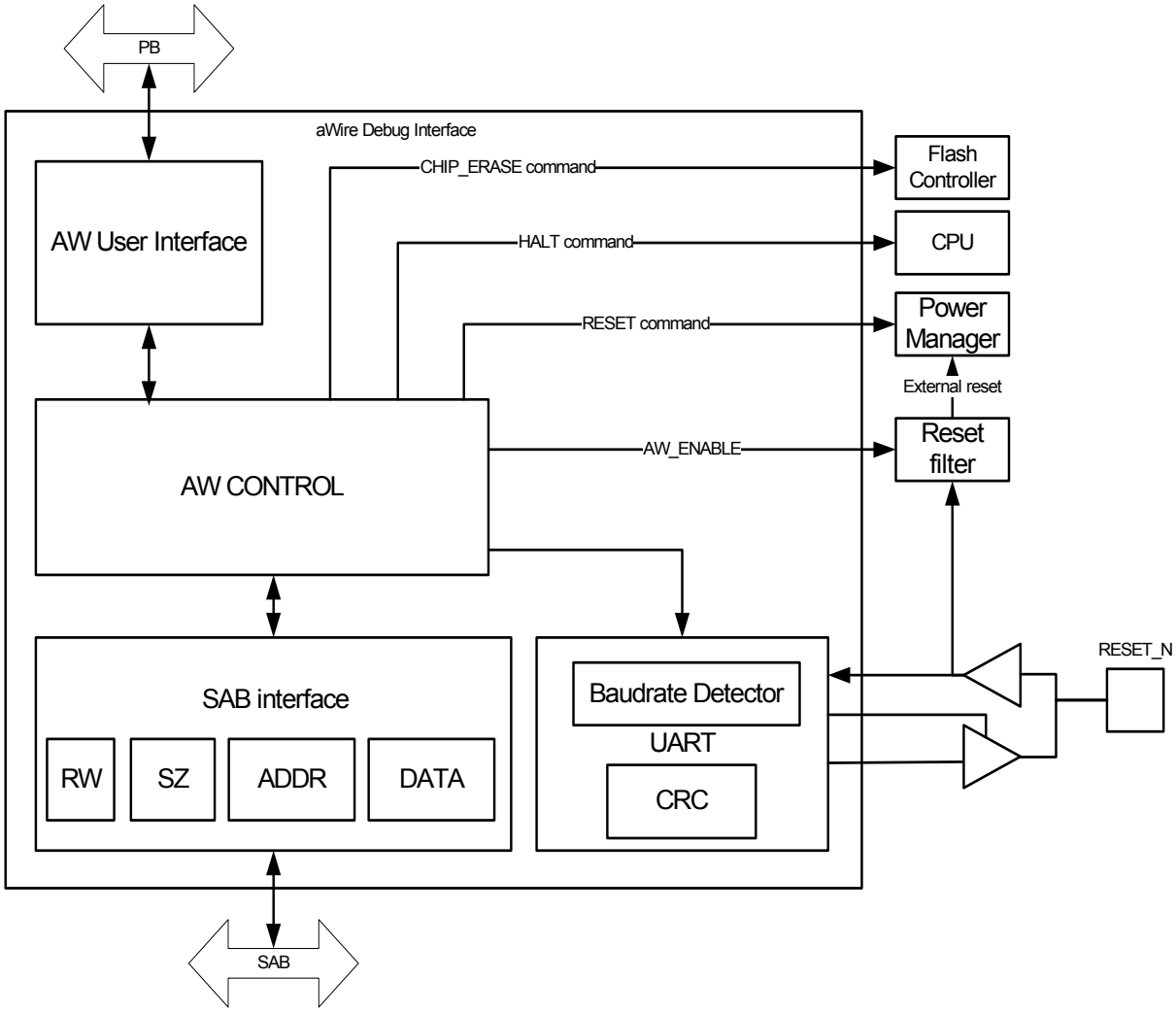
30.2 Overview

If the AW is not used for debugging, the aWire UART can be used by the user to send or receive data with one start bit, eight data bits, no parity bits, and one stop bit. This can be controlled through the aWire UART user interface.

This chapter only describes the aWire UART user interface. For a description of the aWire Debug Interface, please see the Programming and Debugging chapter.

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.

30.6 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 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	0x00000230

31. Programming and Debugging

31.1 Overview

The UC3D 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 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 AVR32 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
Security bit	FLASHCDW security bit set	Programming and debugging not possible, very restricted access.

Below follows a more in depth description of what locations are accessible when the security measures are active.

Table 31-3. Security Bit SAB Restrictions

Name	Address start	Address end	Access
OCD DCCPU, OCD DCEMU, OCD DCSR	0x100000110	0x100000118	Read/Write
User page	0x580800000	0x581000000	Read
Other accesses	-	-	Blocked

Table 31-4. User Code Programming SAB Restrictions

Name	Address start	Address end	Access
OCD DCCPU, OCD DCEMU, OCD DCSR	0x100000110	0x100000118	Read/Write
User page	0x580800000	0x581000000	Read

Table 31-4. User Code Programming SAB Restrictions

Name	Address start	Address end	Access
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: 2.1.2.0

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 UC3D is facilitated by a powerful On-Chip Debug (OCD) system. The user accesses this through an external debug tool which connects to the JTAG or aWire port and the Auxiliary (AUX) port if implemented. The AUX port is primarily used for trace functions, and an aWire- or JTAG-based debugger is sufficient for basic debugging.

The debug system is based on the Nexus 2.0 standard, class 2+, which includes:

- Basic run-time control
- Program breakpoints
- Data breakpoints
- Program trace
- Ownership trace
- Data trace

In addition to the mandatory Nexus debug features, the UC3D 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 31-5 on page 668. 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 31-5. Auxiliary Port Signals

Pin Name	Pin Description	Direction	Active Level	Type
MCKO	Trace data output clock	Output		Digital
MDO[5:0]	Trace data output	Output		Digital
MSEO[1:0]	Trace frame control	Output		Digital
EVTI_N	Event In	Input	Low	Digital
EVTO_N	Event Out	Output	Low	Digital

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

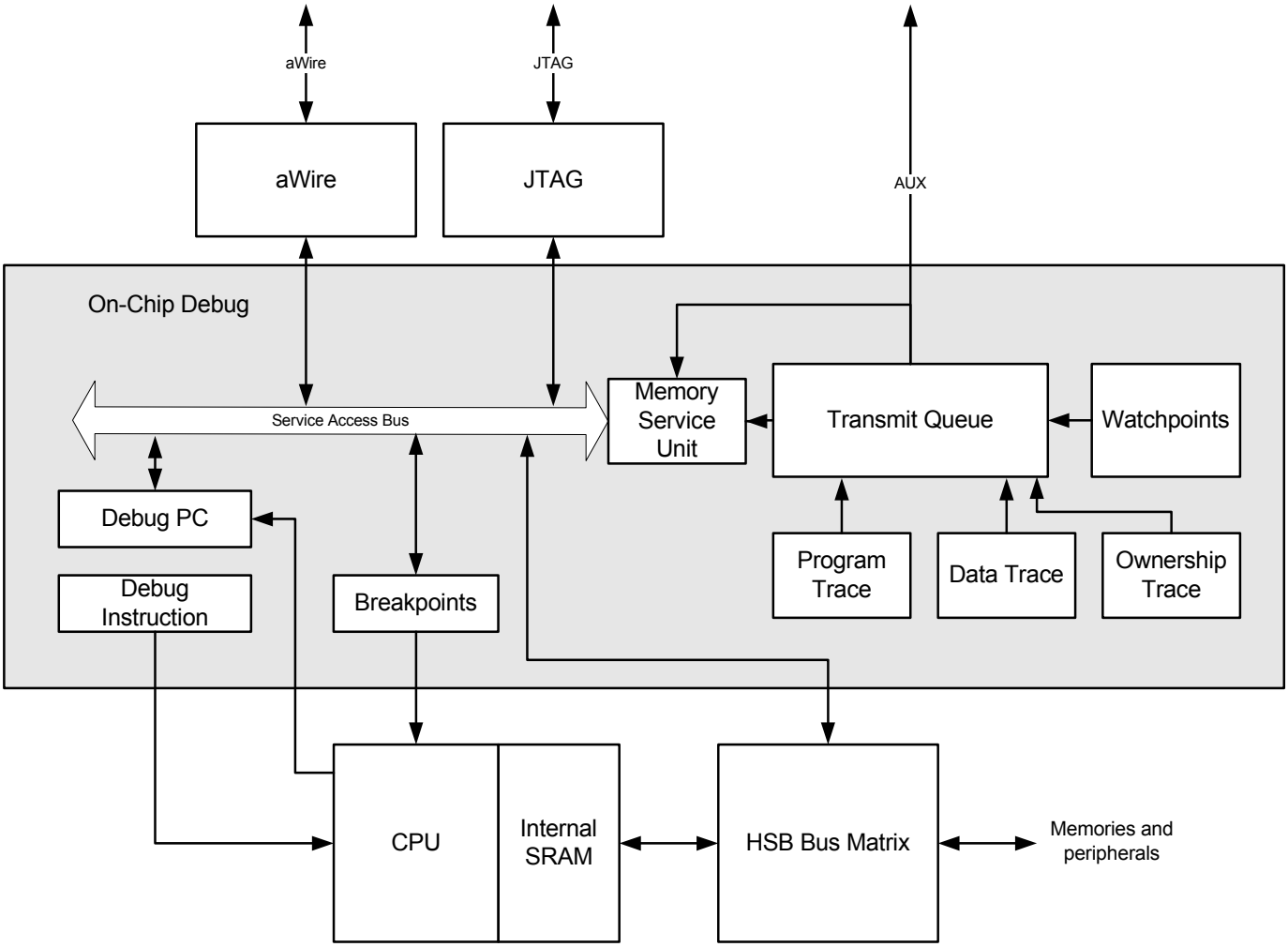
The OCD has a clock (CLK_OCD) running synchronously with the CPU clock. This clock is generated by the Power Manager. The clock is enabled at reset, and can be disabled by writing to the Power Manager.

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

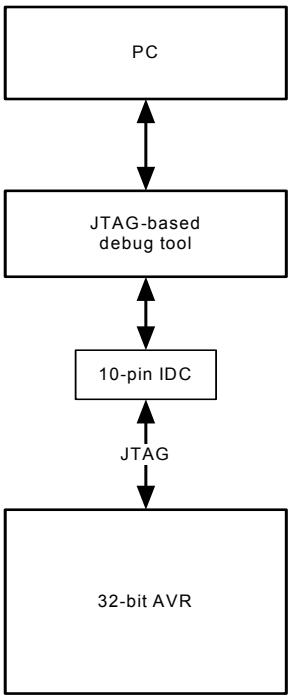
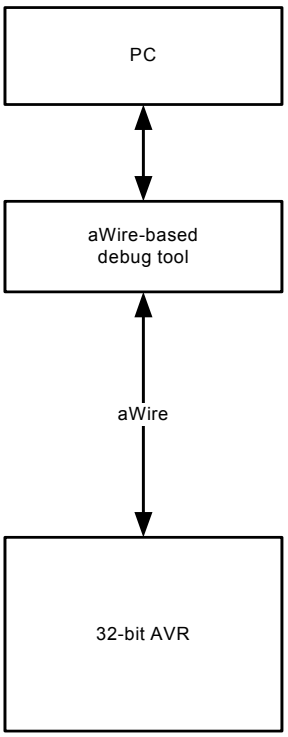


Figure 31-3. aWire-based Debugger



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 UC3D 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*

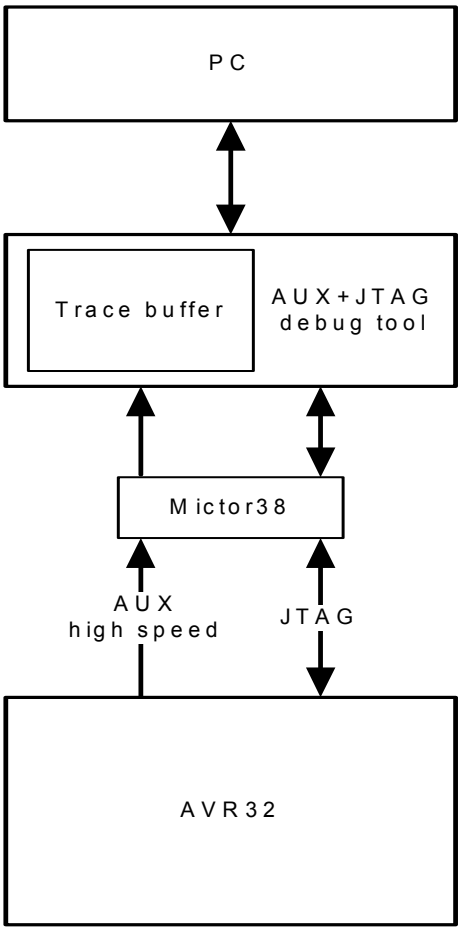
The MSU additionally supports NanoTrace. This is a 32-bit AVR-specific feature, in which trace data is output to memory instead of the AUX port. This allows the trace data to be extracted by the debugger through the SAB, enabling trace features for aWire- or JTAG-based debuggers. The user must write 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 UC3D 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 UC3D 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.

31.4 aWire Debug Interface (AW)

Rev.: 2.3.0.1

31.4.1 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.**

31.4.2 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 31-5 on page 676 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.

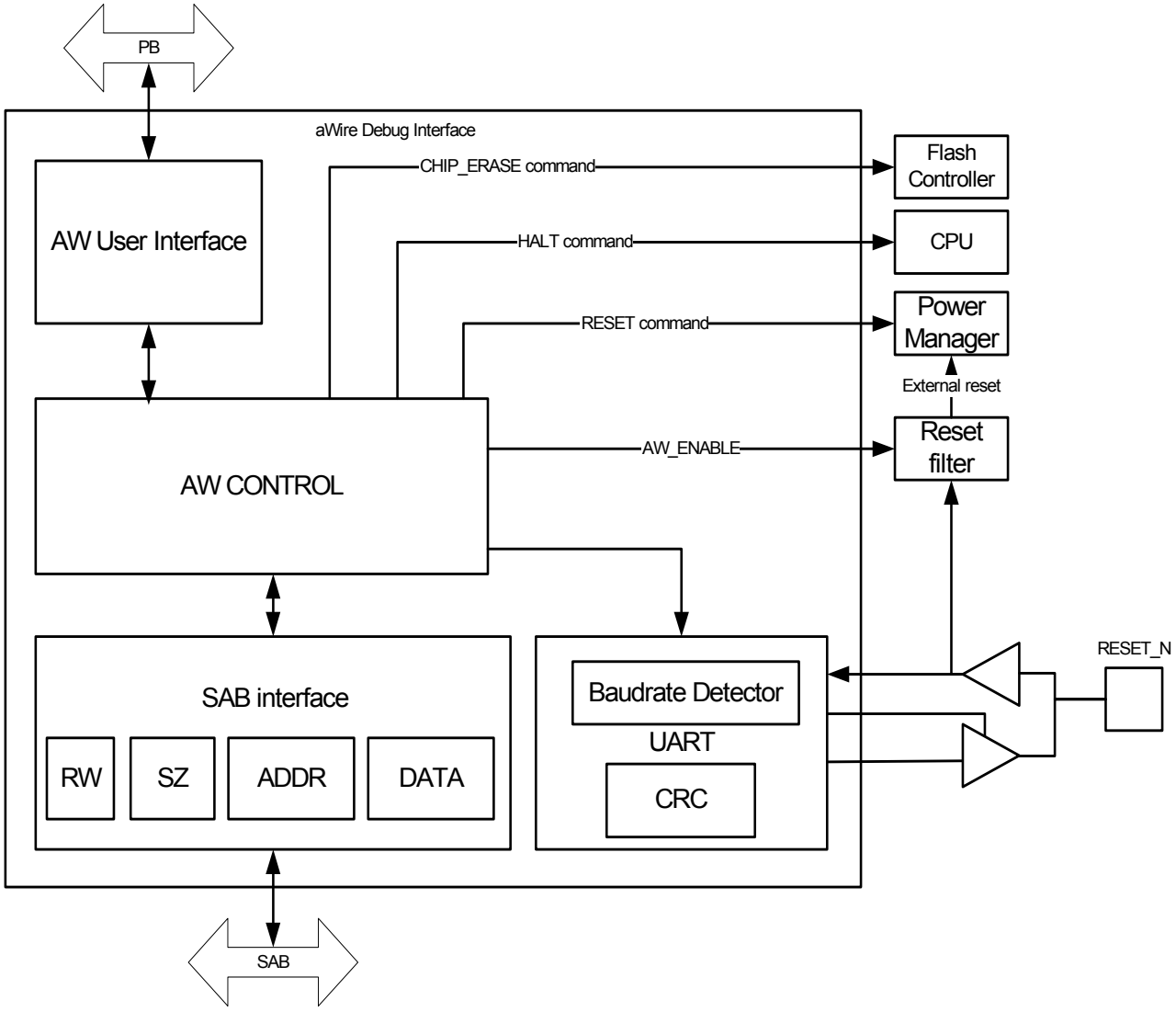
The Service Access Bus (SAB) interface contains address and data registers for the Service Access Bus, which gives access to On-Chip Debug, programming, and other functions in the device. The SAB offers several modes of access to the address and data registers, as discussed in Section 31.4.6.8.

Section 31.4.7 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.

31.4.3 Block Diagram

Figure 31-5. aWire Debug Interface Block Diagram



31.4.4 I/O Lines Description

Table 31-6. 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

31.4.5 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 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.

31.4.5.2 Power Management

When debugging through AW the system clocks are automatically turned on to allow debugging in sleep modes.

31.4.5.3 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.

31.4.6 Functional Description

31.4.6.1 aWire Communication Protocol

The AW is accessed through the RESET_N pin shown in Table 31-6 on page 676. 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 31-7. aWire Packet Format

Field	Number of bytes	Description	Comment	Optional
SYNC	1	Sync pattern (0x55).	Used by the receiver to set the baud rate clock.	No
COMMAND/ RESPONSE	1	Command from the master or response from the slave.	When the most significant bit is set the command/response has a length field. A response has the next most significant bit set. A command does not have this bit set.	No
LENGTH	2	The number of bytes in the DATA field.		Yes
DATA	LENGTH	Data according to command/ response.		Yes
CRC	2	CRC calculated with the FCS16 polynomial.	CRC value of 0x0000 makes the aWire disregard the CRC if the master does not support it.	No

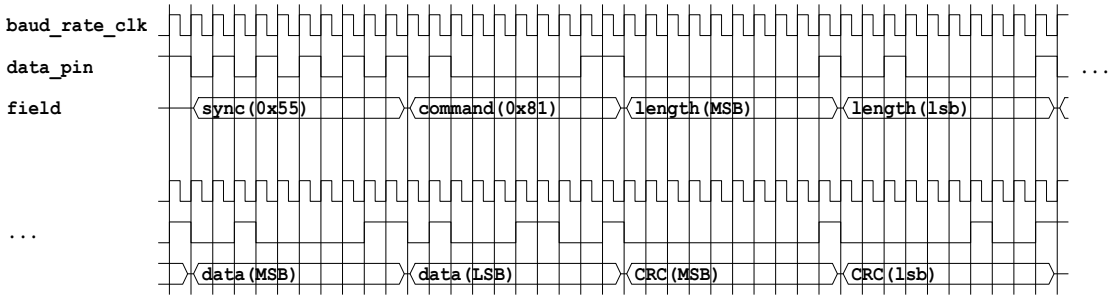
CRC calculation

The CRC is calculated from the command/response, length, and data fields. The polynomial used is the FCS16 (or CRC-16-CCIT) in reverse mode (0x8408) and the starting value is 0x0000.

Example command

Below is an example command from the master with additional data.

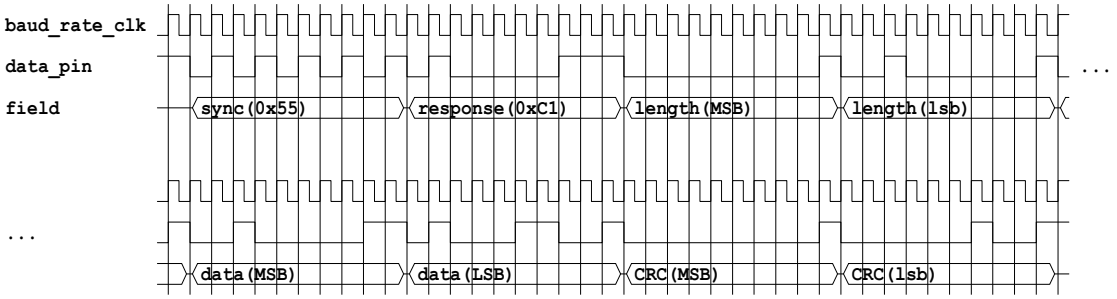
Figure 31-6. Example Command



Example response

Below is an example response from the slave with additional data.

Figure 31-7. 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.

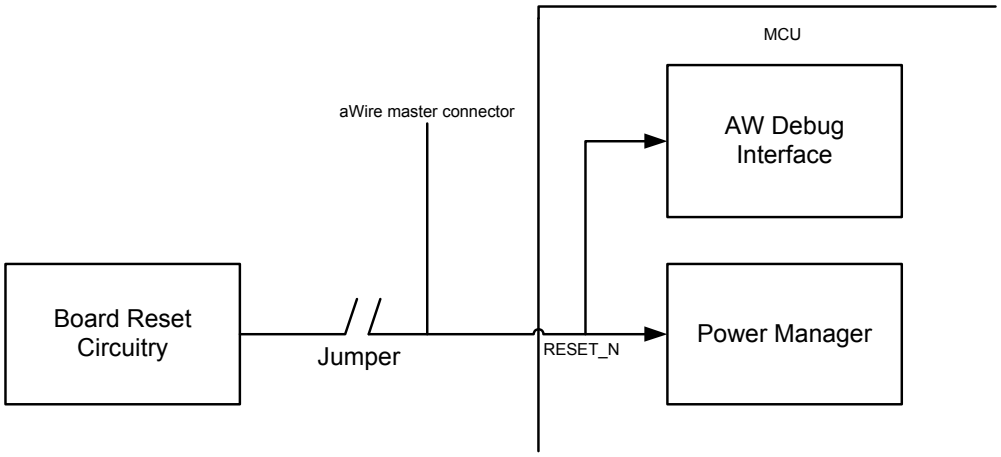
31.4.6.2 *The RESET_N pin*

During the direction change there can be a period when the line is not driven.

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 31-8. Reset Circuitry and aWire.



31.4.6.3 *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.

31.4.6.4 *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.

31.4.6.5 *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.

31.4.6.6 *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.

31.4.6.7 *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:

$$f_{brmax} = \frac{f_{aw}}{4}$$

31.4.6.8 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 31.4.8.5 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.

31.4.6.9 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.

31.4.7 aWire Command Summary

The implemented aWire commands are shown in the table below. The responses from the AW are listed in Section 31.4.8.

Table 31-8. 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.

Table 31-8. aWire Command Summary

COMMAND	Instruction	Description
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 31-9. 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.

31.4.7.1 AYA

This command asks the AW: “Are you alive”, where the AW should respond with an acknowledge.

Table 31-10. AYA Details

Command	Details
Command value	0x01
Additional data	N/A
Possible responses	0x40: ACK (Section 31.4.8.1) 0x41: NACK (Section 31.4.8.2)

31.4.7.2 JTAG_ID

This command instructs the AW to output the JTAG idcode in the following response.

Table 31-11. JTAG_ID Details

Command	Details
Command value	0x02
Additional data	N/A
Possible responses	0xC0: IDCODE (Section 31.4.8.3) 0x41: NACK (Section 31.4.8.2)

31.4.7.3 STATUS_REQUEST

Asks the AW for a status message.

Table 31-12. STATUS_REQUEST Details

Command	Details
Command value	0x03
Additional data	N/A
Possible responses	0xC4: STATUS_INFO (Section 31.4.8.7) 0x41: NACK (Section 31.4.8.2)

31.4.7.4 TUNE

Asks the AW for the current baud rate counter value.

Table 31-13. TUNE Details

Command	Details
Command value	0x04
Additional data	N/A
Possible responses	0xC3: BAUD_RATE (Section 31.4.8.6) 0x41: NACK (Section 31.4.8.2)

31.4.7.5 MEMORY_SPEED_REQUEST

Asks the AW for the relative speed between the aWire clock (RC120M) and the SAB interface.

Table 31-14. MEMORY_SPEED_REQUEST Details

Command	Details
Command value	0x05
Additional data	N/A
Possible responses	0xC5: MEMORY_SPEED (Section 31.4.8.8) 0x41: NACK (Section 31.4.8.2)

31.4.7.6 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 31-15. CHIP_ERASE Details

Command	Details
Command value	0x06
Additional data	N/A
Possible responses	0x40: ACK (Section 31.4.8.1) 0x41: NACK (Section 31.4.8.2)

31.4.7.7 DISABLE

Disables the AW. The AW will respond with an ACK response and then disable itself.

Table 31-16. DISABLE Details

Command	Details
Command value	0x07
Additional data	N/A
Possible responses	0x40: ACK (Section 31.4.8.1) 0x41: NACK (Section 31.4.8.2)

31.4.7.8 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 31-17. DISABLE Details

Command	Details
Command value	0x07
Additional data	N/A
Possible responses	0x40: ACK (Section 31.4.8.1) 0x41: NACK (Section 31.4.8.2)

31.4.7.9 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 31-18. 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 31-19. MEMORY_WRITE Details

Command	Details
Command value	0x80
Additional data	Size, Address and Data
Possible responses	0xC2: MEMORY_READWRITE_STATUS (Section 31.4.8.5) 0x41: NACK (Section 31.4.8.2)

31.4.7.10 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 31-20. 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 31-21. MEMORY_READ Details

Command	Details
Command value	0x81
Additional data	Size, Address and Length
Possible responses	0xC1: MEMDATA (Section 31.4.8.4) 0xC2: MEMORY_READWRITE_STATUS (Section 31.4.8.5) 0x41: NACK (Section 31.4.8.2)

31.4.7.11 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 31-22. 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 31.4.8.1) 0x41: NACK (Section 31.4.8.2)

31.4.7.12 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 31-23. 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 31.4.8.1) 0x41: NACK (Section 31.4.8.2)

31.4.7.13 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 31-24. SET_GUARD_TIME Details

Command	Details
Command value	0x84
Additional data	Guard time
Possible responses	0x40: ACK (Section 31.4.8.1) 0x41: NACK (Section 31.4.8.2)

31.4.8 aWire Response Summary

The implemented aWire responses are shown in the table below.

Table 31-25. 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.

31.4.8.1 ACK

The AW has received the command successfully and performed the operation.

Table 31-26. ACK Details

Response	Details
Response value	0x40
Additional data	N/A

31.4.8.2 NACK

The AW has received the command, but got a CRC mismatch.

Table 31-27. NACK Details

Response	Details
Response value	0x41
Additional data	N/A

31.4.8.3 IDCODE

The JTAG idcode for this device.

Table 31-28. IDCODE Details

Response	Details
Response value	0xC0
Additional data	JTAG idcode

31.4.8.4 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 31-29. MEMDATA Status Byte

status byte	Description
0x00	Read successful
0x01	SAB busy
0x02	Bus error (wrong address)
Other	Reserved

Table 31-30. MEMDATA Details

Response	Details
Response value	0xC1
Additional data	Data read, status byte, and byte count (2 bytes)

31.4.8.5 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 31-31. MEMORY_READWRITE_STATUS Status Byte

status byte	Description
0x00	Write successful
0x01	SAB busy
0x02	Bus error (wrong address)
Other	Reserved

Table 31-32. MEMORY_READWRITE_STATUS Details

Response	Details
Response value	0xC2
Additional data	Status byte and byte count (2 bytes)

31.4.8.6 BAUD_RATE

The current baud rate in the AW. See Section 31.4.6.7 for more details.

Table 31-33. BAUD_RATE Details

Response	Details
Response value	0xC3
Additional data	Baud rate

31.4.8.7 STATUS_INFO

A status message from AW.

Table 31-34. 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 31-35. STATUS_INFO Details

Response	Details
Response value	0xC4
Additional data	2 status bytes

31.4.8.8 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 31-36. MEMORY_SPEED Details

Response	Details
Response value	0xC5
Additional data	Clock cycle count (MS)

31.4.9 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.

31.5 JTAG and Boundary-scan (JTAG)

Rev: 2.2.2.4

31.5.1 Features

- **IEEE1149.1 compliant JTAG Interface**
- **Boundary-scan Chain for board-level testing**
- **Direct memory access and programming capabilities through JTAG Interface**

31.5.2 Overview

The JTAG Interface offers a four pin programming and debug solution, including boundary-scan support for board-level testing.

Figure 31-9 on page 692 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.

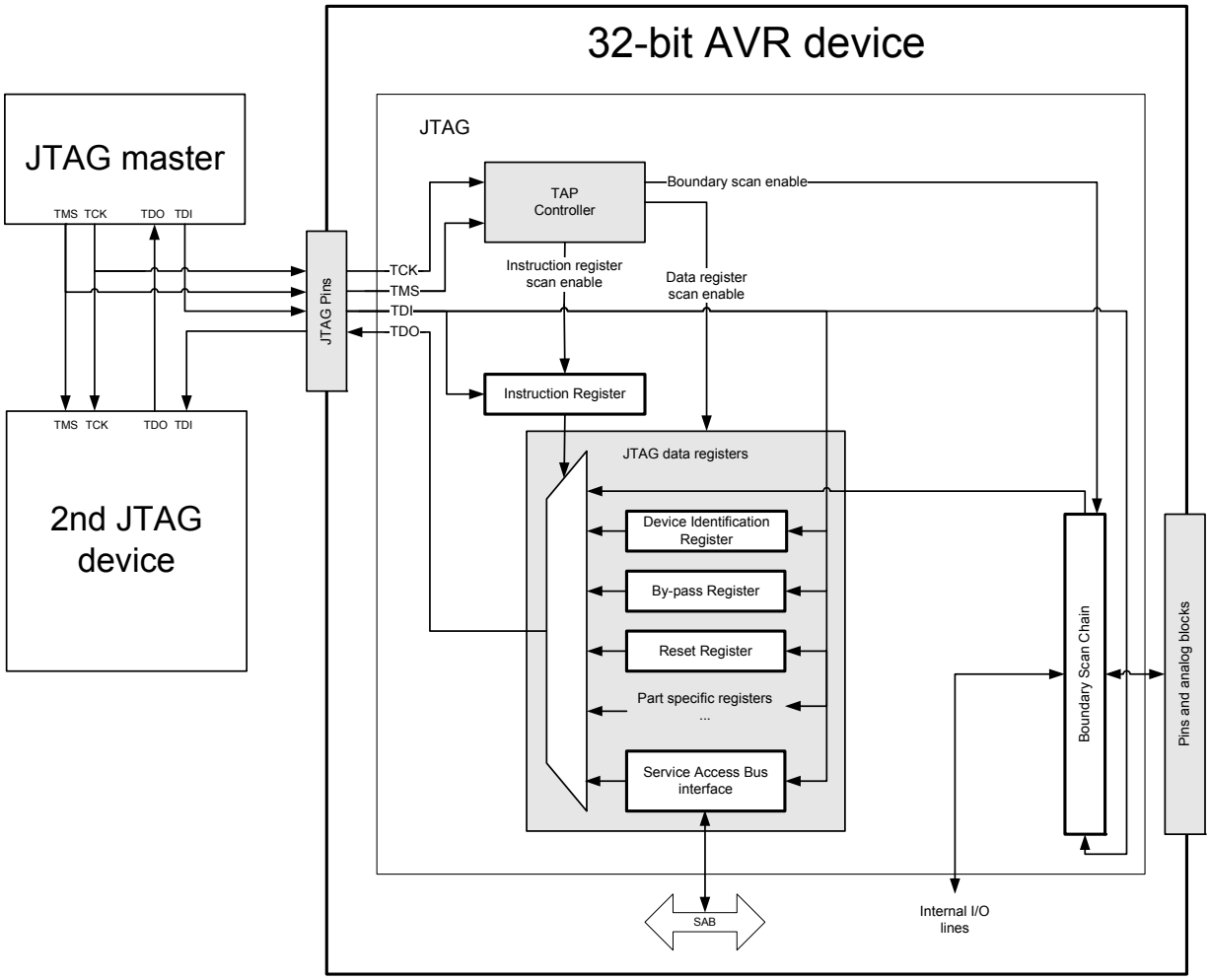
The Instruction Register holds JTAG instructions controlling the behavior of a Data Register. The Device Identification Register, Bypass Register, and the boundary-scan chain are the Data Registers used for board-level testing. The Reset Register can be used to keep the device reset during test or programming.

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 31.5.11.

Section 31.6 lists the supported JTAG instructions, with references to the description in this document.

31.5.3 Block Diagram

Figure 31-9. JTAG and Boundary-scan Access



31.5.4 I/O Lines Description

Table 31-37. 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	

31.5.5 Product Dependencies

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

31.5.5.1 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 31.5.7.

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.

31.5.5.2 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.

31.5.5.3 Clocks

The JTAG Interface uses the external TCK pin as clock source. This clock must be provided by the JTAG master.

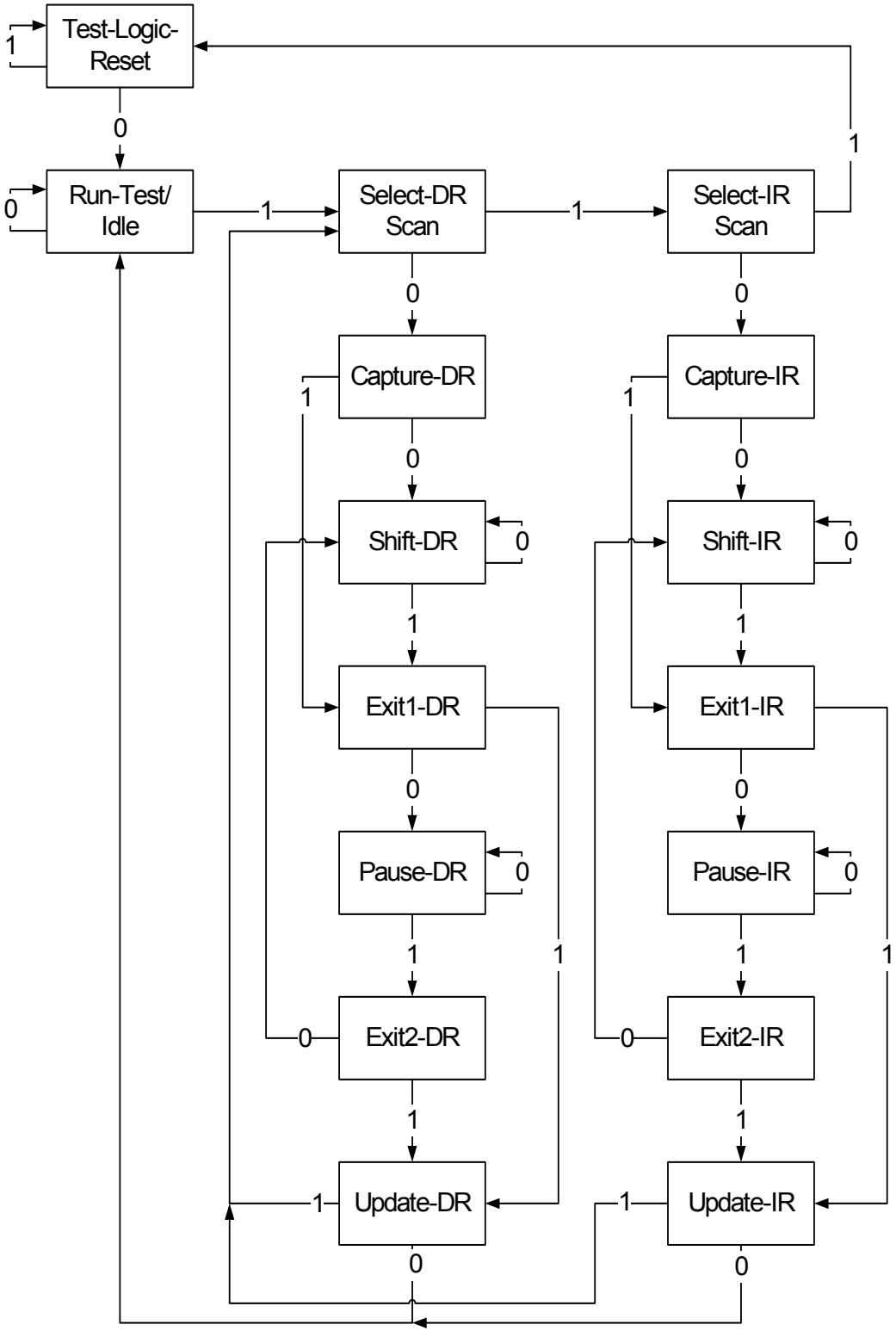
Instructions that use the SAB bus requires the internal main clock to be running.

31.5.6 JTAG Interface

The JTAG Interface is accessed through the dedicated JTAG pins shown in Table 31-37 on page 692. The TMS control line navigates the TAP controller, as shown in Figure 31-10 on page 694. 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 31.6.2, as well as a number of 32-bit AVR-specific private JTAG instructions described in Section 31.6.3. Each instruction selects a specific data register for the Shift-DR path, as described for each instruction.

Figure 31-10. TAP Controller State Diagram



31.5.7 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.

31.5.8 How to disable the module

To disable the JTAG pins the TCK pin must be held high while RESET_N pin is released.

31.5.9 Typical Sequence

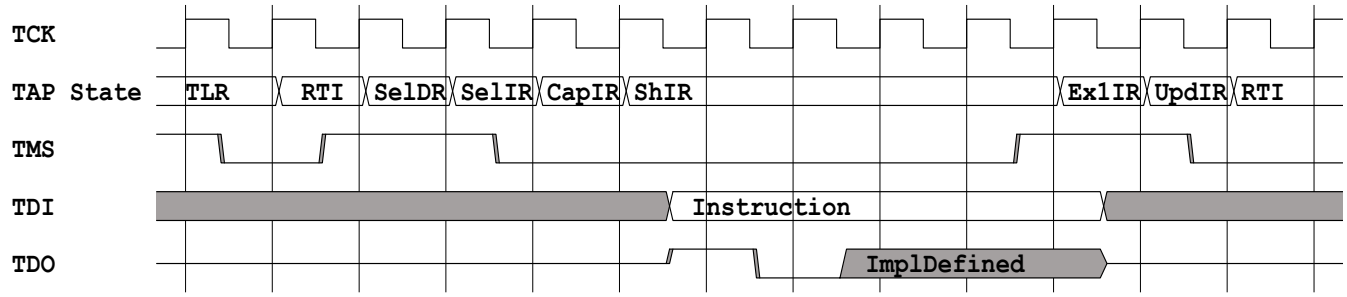
Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG Interface follows.

31.5.9.1 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 31.6 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 31-11. Scanning in JTAG Instruction



31.5.9.2 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.

31.5.10 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.

31.5.11 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.

31.5.11.1 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.

31.5.11.2 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.

31.5.11.3 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.

31.5.11.4 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.

31.5.11.5 *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.

31.5.11.6 *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 31.6.1.

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.

31.6 JTAG Instruction Summary

The implemented JTAG instructions in the 32-bit AVR are shown in the table below.

Table 31-38. 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

31.6.1 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.

31.6.1.1 Notation

Table 31-40 on page 700 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 31-39.

Table 31-39. 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 31.5.11.4 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 31.5.11.5 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 31-40. 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 31-40. 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**

31.6.2 Public JTAG Instructions

The JTAG standard defines a number of public JTAG instructions. These instructions are described in the sections below.

31.6.2.1 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 31-41. 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

31.6.2.2 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 31-42. 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.

31.6.2.3 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 31-43. 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.

31.6.2.4 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 31-44. 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.

31.6.2.5 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 31-45. CLAMP Details

Instructions	Details
IR input value	**00110** (0x06)
IR output value	p0001
DR Size	1
DR input value	x
DR output value	x

31.6.2.6 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 31-46. BYPASS Details

Instructions	Details
IR input value	**11111** (0x1F)
IR output value	p0001
DR Size	1
DR input value	x
DR output value	x

31.6.3 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.

31.6.3.1 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 31-47. 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**

31.6.3.2 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 31-48. 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**

31.6.3.3 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 31-49.

Table 31-49. 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 ddddddd** xxxxxxxx xxxxxxxx 1: Not allowed 2: xxxxxxxx xxxxxxxx **dddddddd ddddddd** 3: Not allowed
10	Word (32 bits)	Address modulo 4 : data alignment 0: **dddddddd ddddddd ddddddd ddddddd** 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 31-50. 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 31-50. 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**

31.6.3.4 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 31-51. 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 31-51. 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**

31.6.3.5 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 31-52. 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 31-52. MEMORY_BLOCK_ACCESS Details (Continued)

Instructions	Details
DR input value (Data write phase)	**ddddddddd ddddddddd ddddddddd ddddddddd** xx
DR output value (Data read phase)	eb **ddddddddd ddddddddd ddddddddd ddddddddd**
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.

31.6.3.6 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 31-53. 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

31.6.3.7 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 31-54. 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

31.6.3.8 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 31-55. AVR_RESET Details

Instructions	Details
IR input value	**01100** (0x0C)
IR output value	p0001

Table 31-55. AVR_RESET Details (Continued)

Instructions	Details
DR Size	Device specific.
DR input value	Device specific.
DR output value	Device specific.

31.6.3.9 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 31-56. 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

31.6.3.10 HALT

This instruction allows a programmer to easily stop the CPU to ensure that it does not execute invalid code during programming.

This instruction selects a 1-bit halt register. Setting this bit to one resets the device and halts the CPU. Setting this bit to zero resets the device and releases the CPU to run normally. The value shifted out from the data register is one if the CPU is halted.

The HALT instruction can be used in the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. Return to Run-Test/Idle.
5. Select the DR Scan path.

6. In Shift-DR: Scan in the value 1 to halt the CPU, 0 to start CPU execution.
7. Return to Run-Test/Idle.

Table 31-57. HALT Details

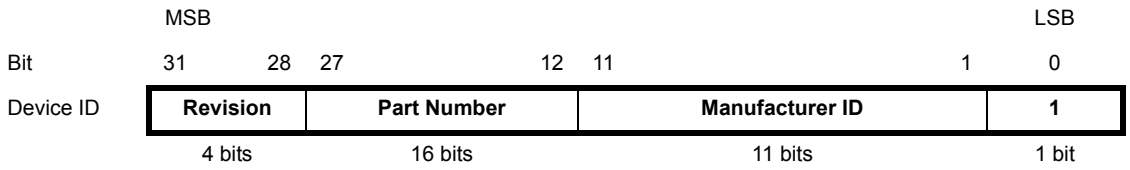
Instructions	Details
IR input value	**11100** (0x1C)
IR output value	p0001
DR Size	1 bit
DR input value	d
DR output value	d

31.6.4 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.

31.6.4.1 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 31-58. Note that if the flash controller is statically reset, the ID code will be undefined.

Table 31-58. Device and JTAG ID

Device Name	JTAG ID Code (R is the revision number)
ATUC256D3	0xr212303F
ATUC128D3	0xr212003F
ATUC64D3	0xr212103F
ATUC256D4	0xr213303F
ATUC128D4	0xr213003F
ATUC64D4	0xr213103F

31.6.4.2 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.

Bit	Reset Domain	Description
0	System	Resets the whole chip, except the JTAG itself.

31.6.4.3 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.

31.7 NanoTrace and Auxilliary Port

NanoTrace and Auxilliary(AUX) Port features have not been implemented in this chip. All references to these concepts are invalid.

31.8 Module Configuration

The bit mapping of the Peripheral Debug Register (PDBG) is described in the following table. Please refer to the On-Chip-Debug chapter in the AVR32UC Technical Reference Manual for details.

Table 31-59. Bit Mapping of the Peripheral Debug Register (PDBG)

bit	Peripheral
0	WDT
1	AST
2	ADC

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 152mA
Total DC output current on all I/O pins - VDDANA..... 152mA
Maximum operating voltage VDDCORE..... 1.95V
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 8

2. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pin. Refer to Section 3.2 on page 8 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

Symbol	Parameter	Voltage		
		Min	Max	Unit
V_{VDDIO}	DC supply peripheral I/Os	3.0	3.6	V
V_{VDDIN}	DC supply internal regulator, 3.3V single supply mode	3.0	3.6	V
$V_{VDDCORE}$	DC supply core	1.65	1.95	V
V_{VDDANA}	Analog supply voltage	3.0	3.6	V
$V_{ADVREFP}$	Analog reference voltage	2.6	V_{VDDANA}	V

32.4 Maximum Clock Frequencies

These parameters are given in the following conditions:

- $V_{VDDCORE}$ = 1.65 to 1.95V

- Temperature = -40°C to 85°C

Table 32-3. Clock Frequencies

Symbol	Parameter	Conditions	Min	Max	Units
f_{CPU}	CPU clock frequency			48	MHz
f_{PBA}	PBA clock frequency			48	MHz
f_{PBB}	PBB clock frequency			48	MHz
f_{GCLK0}	GCLK0 clock frequency	GLOC, GCLK0 pin		48	MHz
f_{GCLK1}	GCLK1 clock frequency	GCLK1 pin		48	MHz
f_{GCLK2}	GCLK2 clock frequency	GCLK2 pin		48	MHz
f_{GCLK3}	GCLK3 clock frequency	USB		48	MHz
f_{GCLK4}	GCLK4 clock frequency	PWMA		150	MHz
f_{GCLK5}	GCLK5 clock frequency	IISC		48	MHz
f_{GCLK6}	GCLK6 clock frequency	AST		80	MHz
f_{GCLK8}	GCLK8 clock frequency	ADCIFB		48	MHz

32.5 Power Consumption

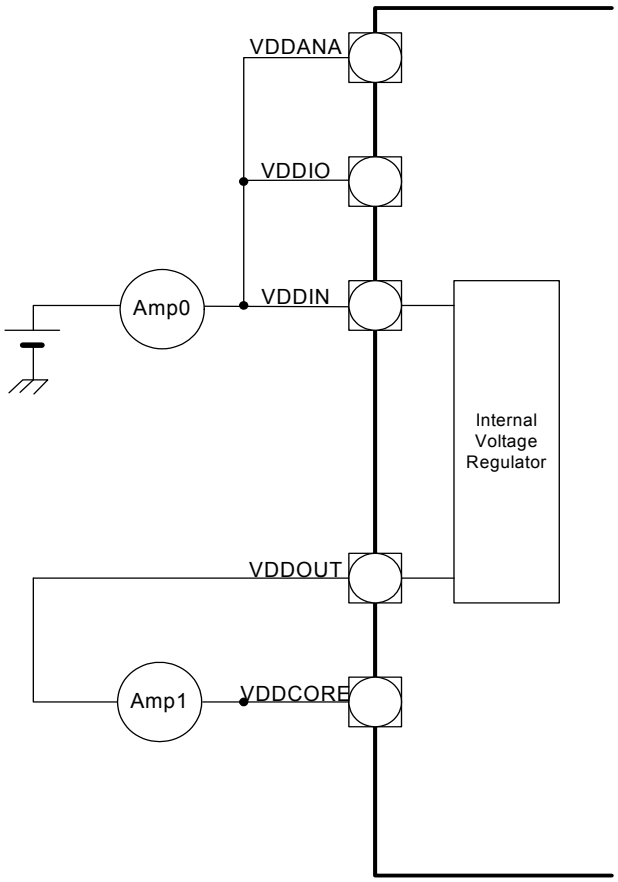
The values in Table 32-4 are measured values of power consumption under the following conditions, except where noted:

- Operating conditions internal core supply (Figure 32-1) - this is the default configuration
  - V_{VDDIN} = 3.3V
  - $V_{VDDCORE}$ = 1.8V, supplied by the internal regulator
  - Corresponds to the 3.3V supply mode with 1.8 V regulated I/O lines, please refer to the Supply and Startup Considerations section for more details
  - The following peripheral clocks running
- T_A = 25°C
- Oscillators
  - OSC0 running (external clock)as reference
  - PLL running at 48MHz with OSC0 as reference
- Clocks
  - PLL 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

Table 32-4. Power Consumption for Different Operating Modes

Mode	Conditions		Consumption Typ	Unit
Active	- CPU running a recursive Fibonacci Algorithm from flash and clocked from PLL0 at f MHz. - Voltage regulator is on. - XIN0: external clock. - All peripheral clocks activated with a division by 4. - GPIOs are inactive with internal pull-up, JTAG unconnected with external pull-up and Input pins are connected to GND		0.3105xf(MHz) + 0.2707	mA/MHz
	Same conditions at 48MHz		15.17	mA
Idle	See Active mode conditions		0.1165xf(MHz) + 0.1457	mA/MHz
	Same conditions at 48MHz		5.74	mA
Frozen	See Active mode conditions		0.0718xf(MHz) + 0.0903	mA/MHz
	Same conditions at 48MHz		3.54	mA
Standby	See Active mode conditions		0.0409xf(MHz) + 0.0935	mA/MHz
	Same conditions at 48MHz		2.06	mA
Stop	- CPU running in sleep mode - XIN0, Xin1 and XIN32 are stopped. - All peripheral clocks are desactived. - GPIOs are inactive with internal pull-up, JTAG unconnected with external pull-up and Input pins are connected to GND.	Voltage Regulator On	60	µA
		Voltage Regulator Off	51	µA
Deepstop	See Stop mode conditions	Voltage Regulator On	26	µA
		Voltage Regulator Off	17	µA
Static	See Stop mode conditions	Voltage Regulator On	13	µA
		Voltage Regulator Off	3.5	µA

Figure 32-1. Measurement Schematic, External Core Supply



32.5.1 Peripheral Power Consumption

The values in Table 32-5 are measured values of power consumption under the following conditions.

- Operating conditions external core supply (Figure 32-1)
  - V_{VDDIN} = 3.3V
  - $V_{VDDCORE}$ = 1.8V, supplied by the internal regulator
  - Corresponds to the 3.3V + 1.8V dual supply mode , please refer to the Supply and Startup Considerations section for more details
- T_A = 25°C
- Oscillators
  - OSC0 on external clock running
  - PLL running at 48MHz with OSC0 as reference
- Clocks
  - OSC0 external clock used as main clock source
  - CPU, HSB, and PB clocks undivided

- I/Os are inactive with internal pull-up

Consumption idle is the added current consumption when turning the module clock on and the module is uninitialized. Consumption active is the added current consumption when the module clock is turned on and when the module is doing a typical set of operations.

Table 32-5. Typical Current Consumption by Peripheral

Peripheral	Typ Consumption Active	Unit
ADCIFD[1]	3.6	µA/MHz
AST	4.5	
AW USART	9.8	
CAT	14	
EIC	2.3	
FREQM	1.1	
GLOC	1.3	
GPIO	10.6	
IISC	4.7	
PWMA	5.6	
SPI	6.3	
TC	7.3	
TWIM	4.5	
TWIS	2.8	
USART	3.9	
WDT	1.8	

Notes: 1. Includes the current consumption on VDDANA and ADVREFP.

32.6 I/O Pin Characteristics

Table 32-6. Normal I/O Pin Characteristics[1]

Symbol	Parameter	Condition		Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance			9	15	25	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3.0V	[3]	-0.3		+0.8	V
			[4]	-0.3		+0.4	V
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V	[3]	+2		V_{VDD} + 0.3	V
			[4]	+1.6		V_{VDD} + 0.3	V
V_{OL}	Output low-level voltage	V_{VDD} = 3.0V, I_{OL} = 4mA				0.4	V
V_{OH}	Output high-level voltage	V_{VDD} = 3.0V, I_{OH} = 4mA		V_{VDD} - 0.4			V

Table 32-6. Normal I/O Pin Characteristics[1]

Symbol	Parameter	Condition		Min	Typ	Max	Units
I_{OL}	Output low-level current	V_{VDD} = 3.0V	[5]			4	mA
			[6]			8	mA
I_{OH}	Output high-level current	V_{VDD} = 3.0V	[5]			4	mA
			[6]			8	mA
F_{MAX}	Output frequency[2]	V_{VDD} = 3.0V, load = 10 pF	[5]			195	MHz
			[6]			348	MHz
		V_{VDD} = 3.0V, load = 30 pF	[5]			78	MHz
			[6]			149	MHz
t_{RISE}	Rise time[2]	V_{VDD} = 3.0V, load = 10 pF	[5]			2.21	ns
			[6]			1.26	ns
		V_{VDD} = 3.0V, load = 30 pF	[5]			5.45	ns
			[6]			2.88	ns
t_{FALL}	Fall time[2]	V_{VDD} = 3.0V, load = 10 pF	[5]			2.57	ns
			[6]			1.44	ns
		V_{VDD} = 3.0V, load = 30 pF	[5]			6.41	ns
			[6]			3.35	ns
I_{LEAK}	Input leakage current	Pull-up resistors disabled				1	µA
C_{IN}	Input capacitance,	[7]			2		pF
		PA09, PA10			16.5		pF
		PA11, PA12, PA18, PA19			18.5		pF
		PB14, PB15			5		pF

Notes:
1. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pin. Refer to Section 3.2 on page 8 for details.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.
3. This applies to all normal drive pads except PB13, PB17 and PB18.
4. This applies to PB13, PB17 and PB18 pads only.
5. This applies to all normal drive pad except PA00, PA01, PA02, PA03, PA04, PA05, PA06, PA07, PA08, PA09, PA10, PA11, PA12, PA13, PA18, PA19, PA27, PA30, PA31, PB13, PB16 and RESET_N.
6. This applies to PA00, PA01, PA02, PA03, PA04, PA05, PA06, PA07, PA08, PA09, PA10, PA11, PA12, PA13, PA18, PA19, PA27, PA30, PA31, PB13, PB16 and RESET_N pads only.
7. This applies to all normal drive pads except PA09, PA10, PA11, PA12, PA18, PA19, PB14, PB15.

Table 32-7. High-drive I/O Pin Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance		9	15	25	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3.0V	-0.3		+0.8	V
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V	+2		V_{VDD} + 0.3	V
V_{OL}	Output low-level voltage	V_{VDD} = 3.0V, I_{OL} = 6mA			0.4	V

Table 32-7. High-drive I/O Pin Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{OH}	Output high-level voltage	V_{VDD} = 3.0V, I_{OH} = 6mA	V_{VDD} - 0.4			V
I_{OL}	Output low-level current	V_{VDD} = 3.0V			16	mA
I_{OH}	Output high-level current	V_{VDD} = 3.0V			16	mA
F_{MAX}	Output frequency	V_{VDD} = 3.0V, load = 10 pF			471	MHz
		V_{VDD} = 3.0V, load = 30 pF			249	MHz
t_{RISE}	Rise time, all High-drive I/O pins	V_{VDD} = 3.0V, load = 10 pF			0.86	ns
		V_{VDD} = 3.0V, load = 30 pF			1.70	ns
t_{FALL}	Fall time	V_{VDD} = 3.0V, load = 10 pF			1.06	ns
		V_{VDD} = 3.0V, load = 30 pF			2.01	ns
I_{LEAK}	Input leakage current	Pull-up resistors disabled			1	µA
C_{IN}	Input capacitance,	TQFP48 package		2		pF

Notes: 1. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pin. Refer to Section 3.2 on page 8 for details.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

Table 32-8. PB14-DP, PB15-DM Pins Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance		50	100	150	kOhm

Table 32-9. PB16-VBUS Pin Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
R_{PULLUP}[3]	Pull-up resistance					kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3.0V	-0.3		+0.8	V
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V	+2		V_{VDD} + 0.3	V
I_{LEAK}	Input leakage current	5.5V, pull-up resistors disabled			1	µA
C_{IN}	Input capacitance	48 pin packages		0.6		pF

Notes: 1. V_{VDD} corresponds to either V_{VDDIN} or V_{VDDIO}, depending on the supply for the pin. Refer to Section 3.2 on page 8 for details.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.
3. PB16-VBUS pad has no pull-up resistance

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 32-10. 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 32-2. The user must choose a crystal oscillator where the crystal load capacitance C_L is within the range given in the table. The exact value of C_L can be found in the crystal datasheet. The capacitance of the external capacitors (C_{LEXT}) can then be computed as follows:

$$C_{LEXT} = 2(C_L - C_i) - C_{PCB}$$

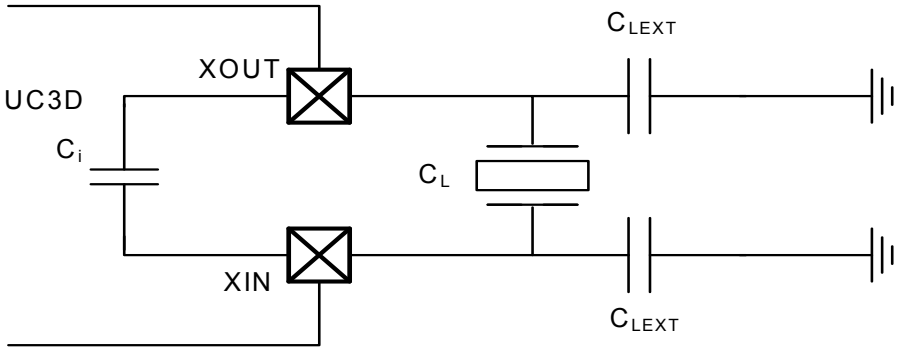
where C_{PCB} is the capacitance of the PCB.

Table 32-11. Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal oscillator frequency		0.4		20	MHz
C_L	Crystal load capacitance		6		18	pF
C_i	Internal equivalent load capacitance			1.7		pF
$t_{STARTUP}$	Startup time	400 KHz Resonator SCIF.OSCCTRL.GAIN = 0[1]		198		µs
		2 MHz Quartz SCIF.OSCCTRL.GAIN = 0[1]		4666		
		8 MHz Quartz SCIF.OSCCTRL.GAIN = 1[1]		975		
		12 MHz Quartz SCIF.OSCCTRL.GAIN = 2[1]		615		
		16 MHz Quartz SCIF.OSCCTRL.GAIN = 2[1]		1106		
		20 MHz Quartz SCIF.OSCCTRL.GAIN = 3[1]		1109		

Notes: 1. Please refer to the SCIF chapter for details.

Figure 32-2. Oscillator Connection



32.7.2 32KHz Crystal Oscillator (OSC32K) Characteristics

32.7.2.1 Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN32.

Table 32-12. Digital Clock Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{CPXIN}	XIN32 clock frequency			32.768	5000	KHz
t_{CPXIN}	XIN32 clock duty cycle		40		60	%

Figure 32-2 and the equation above also applies to the 32KHz oscillator connection. The user must choose a crystal oscillator where the crystal load capacitance C_L is within the range given in the table. The exact value of C_L can then be found in the crystal datasheet.

Table 32-13. 32 KHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal oscillator frequency			32.768	5000	KHz
t_{ST}	Startup time	R_S = 50kOhm, C_L = 9pF		2		s
C_L	Crystal load capacitance		6		15	pF
C_i	Internal equivalent load capacitance			1.4		pF

32.7.3 Phase Locked Loop (PLL) Characteristics

Table 32-14. Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
F_{OUT}	VCO Output Frequency		80		240	MHz
F_{IN}	Input Frequency		4		16	MHz
I_{PLL}	Current Consumption	Active mode F_{VCO}@80 MHz		240		µA
		Active mode F_{VCO}@240 MHz		600		
$t_{STARTUP}$	Startup time, from enabling the PLL until the PLL is locked	Wide Bandwith mode disabled		15		µs
		Wide Bandwith mode enabled		45		

32.7.4 120 MHz RC Oscillator (RC120M) Characteristics

Table 32-15. Internal 120 MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency[1]		88	120	152	MHz
I_{RC120M}	Current consumption			1.85		mA
$t_{STARTUP}$	Startup time			3		µs

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

32.7.5 System RC Oscillator (RCSYS) Characteristics

Table 32-16. System RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency	Calibrated point Ta = 85°C	110	115.2	116	kHz
		Ta = 25°C	105	109	115	kHz
		Ta = -40°C	100	104	108	kHz

32.8 Flash Characteristics

Table 32-17 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 32-17. Maximum Operating Frequency

Flash Wait States	Maximum Operating Frequency
1	48 MHz
0	24 MHz

Table 32-18. Flash Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_{FPP}	Page programming time	f_{CLK_HSB}= 48MHz		5		ms
T_{FPE}	Page erase time			5		
T_{FFP}	Fuse programming time			1		
T_{FEA}	Full chip erase time (EA)			6		
T_{FCE}	JTAG chip erase time (CHIP_ERASE)	f_{CLK_HSB}= 115kHz		310		

Table 32-19. 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 32-20. Electrical Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{VDDIN}	Input voltage range		3	3.3	3.6	V
$V_{VDDCORE}$	Output voltage	V_{VDDIN} >= 3V	1.75	1.8	1.85	V
	Output voltage accuracy	I_{OUT} = 0.1mA to 100mA, V_{VDDIN}>3V		2		%
I_{OUT}	DC output current	V_{VDDIN}=3.3V			100	mA
I_{VREG}	Static current of internal regulator	Low power mode		10		µA

32.9.1.2 Decoupling Requirements

Table 32-21. Decoupling Requirements

Symbol	Parameter	Condition	Typ	Techno.	Units
C_{IN1}	Input regulator capacitor 1		1	NPO	nF

Table 32-21. Decoupling Requirements

Symbol	Parameter	Condition	Typ	Techno.	Units
C_{IN2}	Input regulator capacitor 2		4.7	X7R	nF
C_{OUT1}	Output regulator capacitor 1		470	NPO	nF
C_{OUT2}	Output regulator capacitor 2		2.2	X7R	µF

32.9.2 ADC Characteristics

Table 32-22. Channel Conversion Time and ADC Clock

Parameter	Conditions	Min.	Typ.	Max.	Unit
ADC Clock Frequency	10-bit resolution mode			5	MHz
	8-bit resolution mode			8	MHz
Startup Time	Return from Idle Mode			20	µs
Track and Hold Acquisition Time		600			ns
Conversion Time	ADC Clock = 5 MHz			2	µs
	ADC Clock = 8 MHz			1.25	µs
Throughput Rate	ADC Clock = 5 MHz			384 (1)	kSPS
	ADC Clock = 8 MHz			533 (2)	kSPS

1. Corresponds to 13 clock cycles: 3 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.
2. Corresponds to 15 clock cycles: 5 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.

Table 32-23. ADC Power Consumption

Parameter	Conditions	Min.	Typ.	Max.	Unit
Current Consumption on VDDANA (1)	On 13 samples with ADC clock = 5 MHz			1.25	mA

1. Including internal reference input current

Table 32-24. Analog Inputs

Parameter	Conditions	Min.	Typ.	Max.	Unit
Input Voltage Range		0		VDDANA	V
Input Leakage Current				1	µA
Input Capacitance			7		pF
Input Resistance			370	810	Ohm

Table 32-25. Transfer Characteristics in 8-bit mode

Parameter	Conditions	Min.	Typ.	Max.	Unit
Resolution			8		Bit
Absolute Accuracy	ADC Clock = 5 MHz			0.8	LSB
	ADC Clock = 8 MHz			1.5	LSB
Integral Non-linearity	ADC Clock = 5 MHz		0.35	0.5	LSB
	ADC Clock = 8 MHz		0.5	1.0	LSB

Table 32-25. Transfer Characteristics in 8-bit mode

Parameter	Conditions	Min.	Typ.	Max.	Unit
Differential Non-linearity	ADC Clock = 5 MHz		0.3	0.5	LSB
	ADC Clock = 8 MHz		0.5	1.0	LSB
Offset Error	ADC Clock = 5 MHz	-0.5		0.5	LSB
Gain Error	ADC Clock = 5 MHz	-0.5		0.5	LSB

Table 32-26. Transfer Characteristics in 10-bit mode

Parameter	Conditions	Min.	Typ.	Max.	Unit
Resolution			10		Bit
Absolute Accuracy	ADC Clock = 5 MHz			3	LSB
Integral Non-linearity	ADC Clock = 5 MHz		1.5	2	LSB
Differential Non-linearity	ADC Clock = 5 MHz		1	2	LSB
	ADC Clock = 2.5 MHz		0.6	1	LSB
Offset Error	ADC Clock = 5 MHz	-2		2	LSB
Gain Error	ADC Clock = 5 MHz	-2		2	LSB

32.9.3 BOD

The values in Table 32-27 describe the values of the BODLEVEL in the flash General Purpose Fuse register.

Table 32-27. BODLEVEL Values

BODLEVEL Value	Min	Typ	Max	Units
000000b (00)		1.44		V
010111b (23)		1.52		V
011111b (31)		1.61		V
100111b (39)		1.71		V

Table 32-28. BOD Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{HYST}	BOD hysteresis	T=25C°		10		mV
t_{DET}	Detection time	Time with VDDCORE < BODLEVEL necessary to generate a reset signal		1		µs
I_{BOD}	Current consumption			16		µA
$t_{STARTUP}$	Startup time			5		µs

32.9.4 Reset Sequence

Table 32-29. Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{DDRR}	VDDCORE rise rate to ensure power-on-reset		2.5			V/ms
V_{DDFR}	VDDCORE fall rate to ensure power-on-reset		0.01		400	V/ms
V_{POR+}	Rising threshold voltage: voltage up to which device is kept under reset by POR on rising VDDCORE	Rising VDDCORE: $V_{RESTART}$ -> V_{POR+}	1.4	1.55	1.65	V
V_{POR-}	Falling threshold voltage: voltage when POR resets device on falling VDDCORE	Falling VDDCORE: 1.8V -> V_{POR+}	1.2	1.3	1.4	V
$V_{RESTART}$	On falling VDDCORE, voltage must go down to this value before supply can rise again to ensure reset signal is released at V_{POR+}	Falling VDDCORE: 1.8V -> $V_{RESTART}$	-0.1		0.5	V
T_{POR}	Minimum time with VDDCORE < V_{POR-}	Falling VDDCORE: 1.8V -> 1.1V		15		µs
T_{RST}	Time for reset signal to be propagated to system			200	400	µs
T_{SSU1}	Time for Cold System Startup: Time for CPU to fetch its first instruction (RCosc not calibrated)		480		960	µs
T_{SSU2}	Time for Hot System Startup: Time for CPU to fetch its first instruction (RCosc calibrated)			420		µs

Figure 32-3. MCU Cold Start-Up RESET_N tied to VDDIN

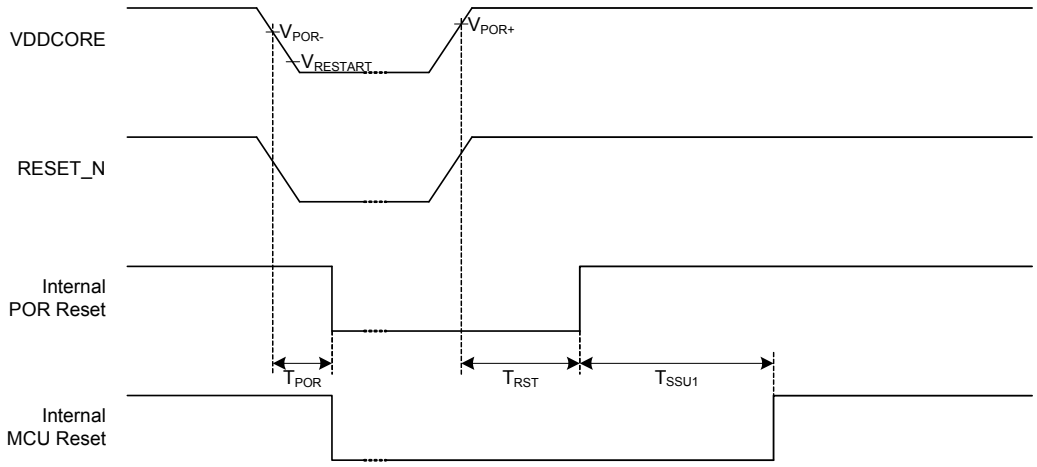


Figure 32-4. MCU Cold Start-Up RESET_N Externally Driven

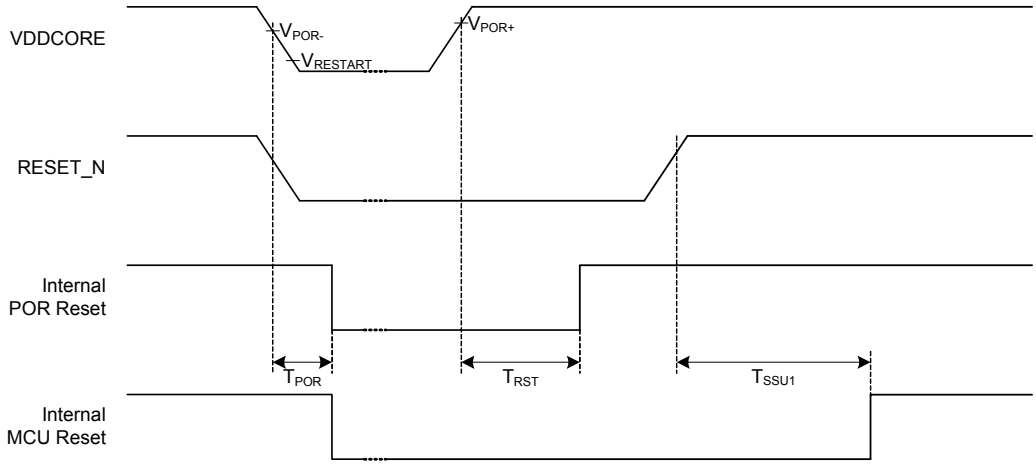
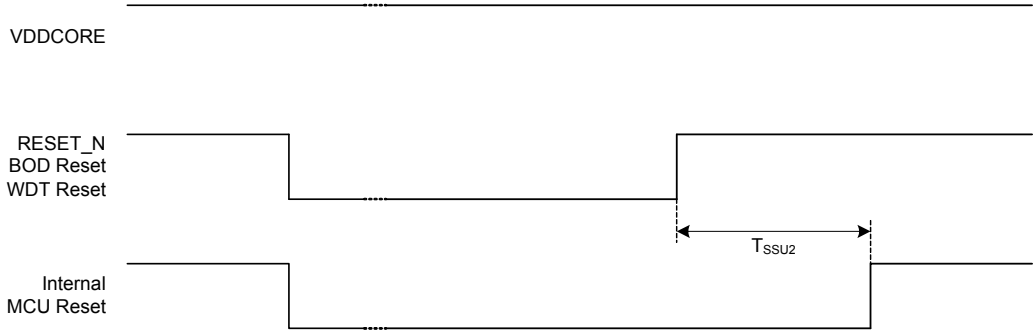


Figure 32-5. MCU Hot Start-Up

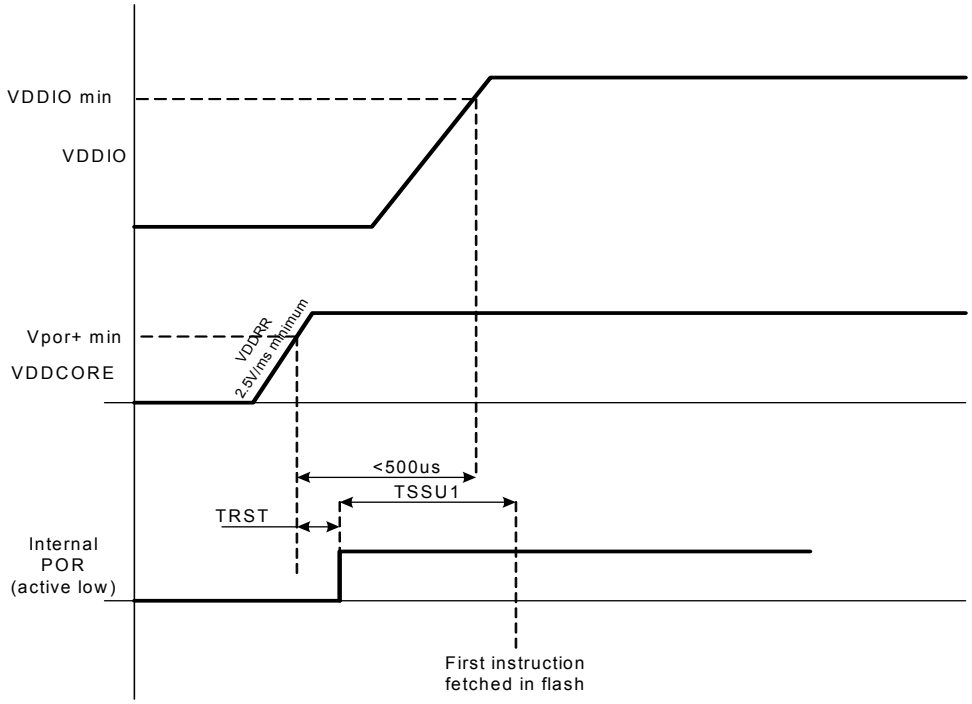


In dual supply configuration, the power up sequence must be carefully managed to ensure a safe startup of the device in all conditions.

The power up sequence must ensure that the internal logic is safely powered when the internal reset (Power On Reset) is released and that the internal Flash logic is safely powered when the CPU fetch the first instructions.

Therefore VDDCORE rise rate (VDDRR) must be equal or superior to 2.5V/ms and VDDIO must reach VDDIO mini value before 500 us (< TRST + TSSU1) after VDDCORE has reached V_{POR+} min value.

Figure 32-6. Dual Supply Configuration



32.9.5 RESET_N Characteristics

Table 32-30. RESET_N Waveform Parameters

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
t_{RESET}	RESET_N minimum pulse width		10			ns

32.10 USB Transceiver Characteristics

32.10.1 Electrical Characteristics

Electrical Parameters

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%		39		Ω

The USB on-chip buffers comply with the Universal Serial Bus (USB) v2.0 standard. All AC parameters related to these buffers can be found within the USB 2.0 electrical specifications.

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	65.1	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP48	23.4	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	QFN48	29.2	°C/W
θ_{JC}	Junction-to-case thermal resistance		QFN48	2.7	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	TQFP64	63.1	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP64	23.0	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	QFN64	26.9	°C/W
θ_{JC}	Junction-to-case thermal resistance		QFN64	2.7	

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. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

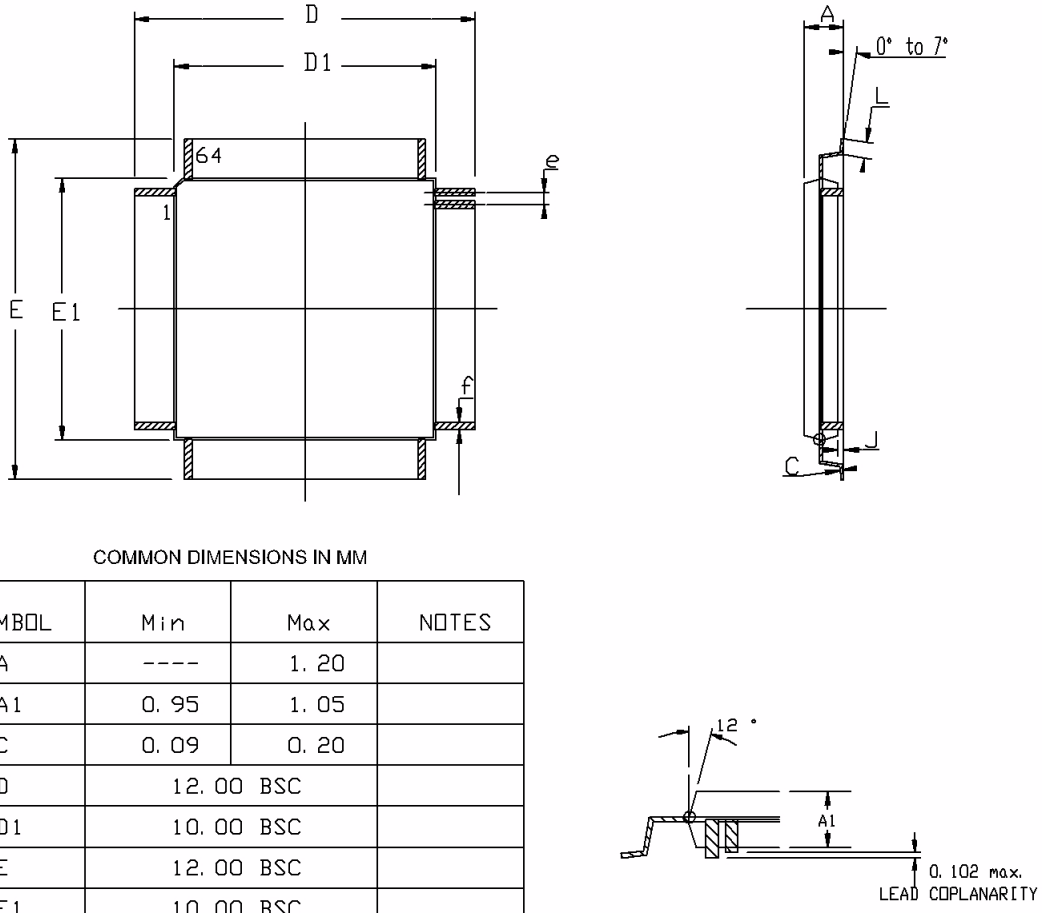
where:

- 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 717.
- 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-64 package drawing



COMMON DIMENSIONS IN MM

SYMBOL	Min	Max	NOTES
A	----	1.20	
A1	0.95	1.05	
C	0.09	0.20	
D	12.00 BSC		
D1	10.00 BSC		
E	12.00 BSC		
E1	10.00 BSC		
J	0.05	0.15	
L	0.45	0.75	
e	0.50 BSC		
f	0.17	0.27	

Table 33-2. Device and Package Maximum Weight

Weight	300 mg

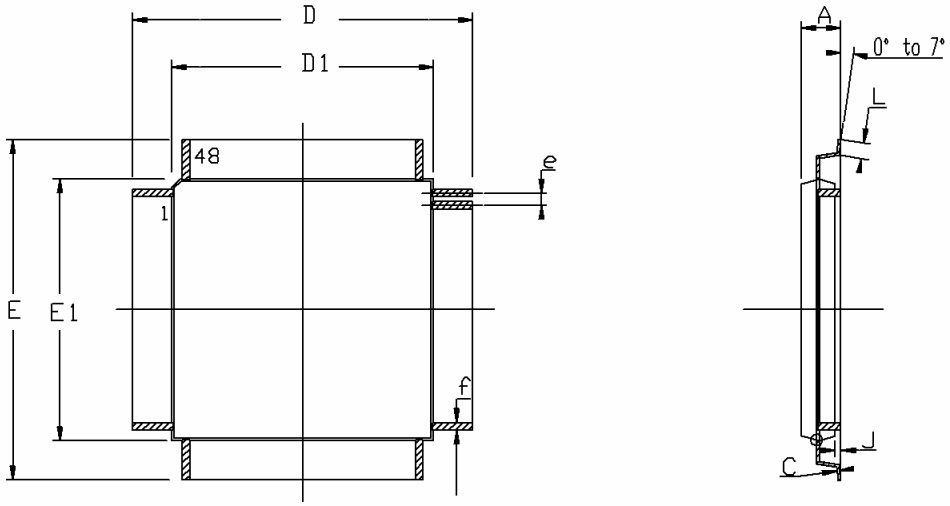
Table 33-3. Package Characteristics

Moisture Sensitivity Level	Jedec J-STD-20D-MSL3

Table 33-4. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	e3

Figure 33-2. 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	
e	0.50 BSC		
f	0.17	0.27	

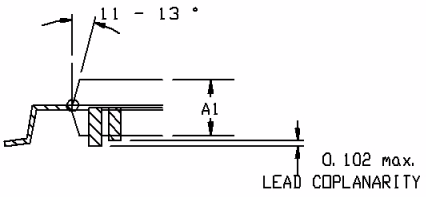


Table 33-5. Device and Package Maximum Weight

Weight	100 mg

Table 33-6. Package Characteristics

Moisture Sensitivity Level	Jedec J-STD-20D-MSL3

Table 33-7. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	e3

Figure 33-3. QFN-48 Package Drawing

Table 33-8. Device and Package Maximum Weight

Weight	100 mg

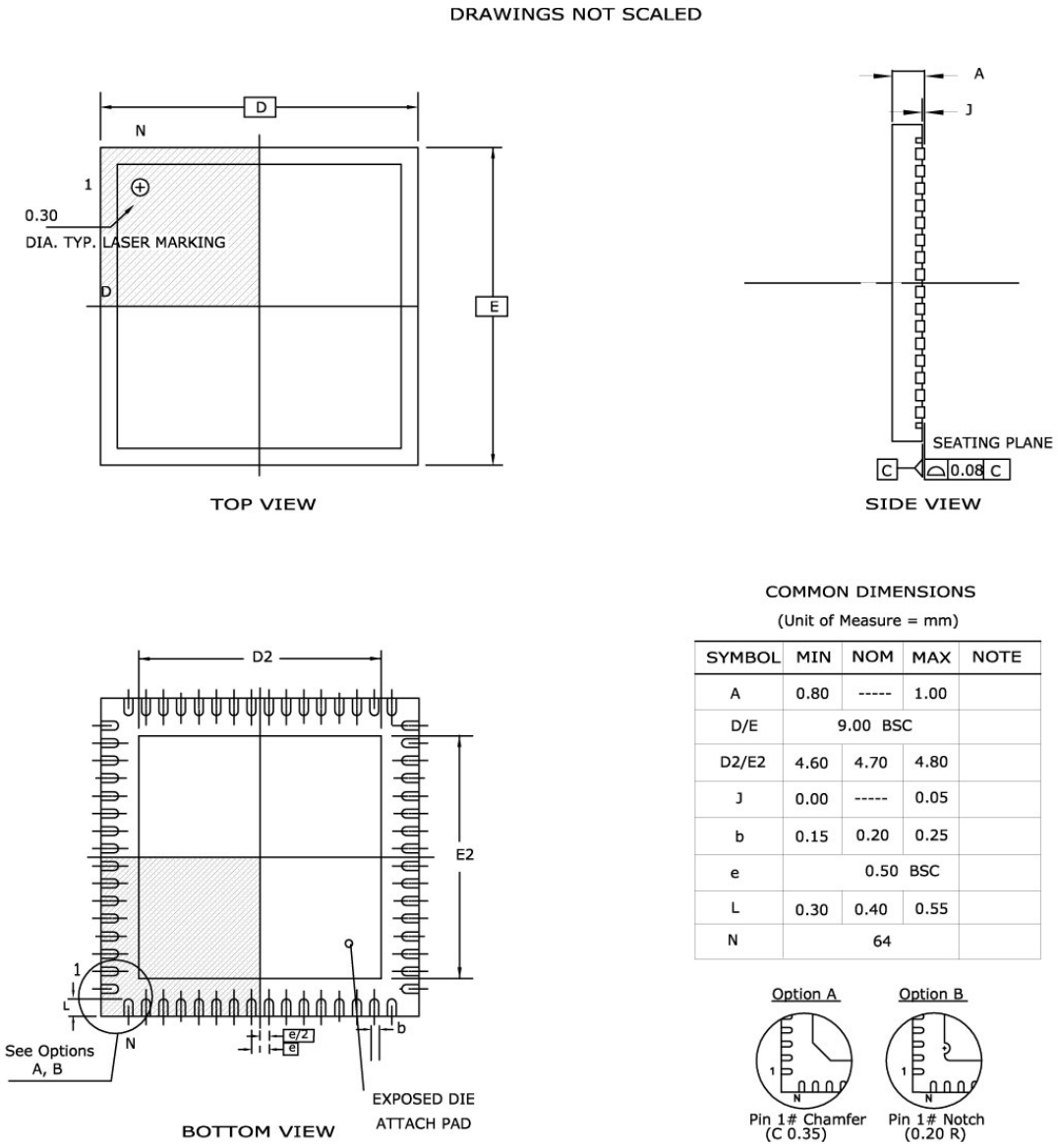
Table 33-9. Package Characteristics

Moisture Sensitivity Level	Jedec J-STD-20D-MSL3

Table 33-10. Package Reference

JEDEC Drawing Reference	M0-220
JESD97 Classification	e3

Figure 33-4. QFN-64 package drawing



SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	-----	1.00	
D/E		9.00 BSC		
D2/E2	4.60	4.70	4.80	
J	0.00	-----	0.05	
b	0.15	0.20	0.25	
e		0.50 BSC		
L	0.30	0.40	0.55	
N		64		

Notes : 1. This drawing is for general information only. Refer to JEDEC Drawing MO-220, Variation VMMD-4, for proper dimensions, tolerances, datums, etc.
2. Dimension b applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip.
If the terminal has the optical radius on the other end of the terminal, the dimension should not be measured in that radius area.

Table 33-11. Device and Package Maximum Weight

Weight	200 mg

Table 33-12. Package Characteristics

Moisture Sensitivity Level	Jedec J-STD-20D-MSL3

Table 33-13. Package Reference

JEDEC Drawing Reference	M0-220
JESD97 Classification	e3

33.3 Soldering Profile

Table 33-14 gives the recommended soldering profile from J-STD-20.

Table 33-14. 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°C min, 200°C max
Temperature 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
ATUC128D3	ATUC128D3-A2UT	Tray	TQFP 64	JESD97 Classification E3	Industrial (-40·C to 85·C)
	ATUC128D3-A2UR	Tape & Reel	TQFP 64		
	ATUC128D3-Z2UT	Tray	QFN 64		
	ATUC128D3-Z2UR	Tape & Reel	QFN 64		
ATUC128D4	ATUC128D4-AUT	Tray	TQFP 48		
	ATUC128D4-AUR	Tape & Reel	TQFP 48		
	ATUC128D4-Z1UT	Tray	QFN 48		
	ATUC128D4-Z1UR	Tape & Reel	QFN 48		
ATUC64D3	ATUC64D3-A2UT	Tray	TQFP 64	JESD97 Classification E3	Industrial (-40·C to 85·C)
	ATUC64D3-A2UR	Tape & Reel	TQFP 64		
	ATUC64D3-Z2UT	Tray	QFN 64		
	ATUC64D3-Z2UR	Tape & Reel	QFN 64		
ATUC64D4	ATUC64D4-AUT	Tray	TQFP 48		
	ATUC64D4-AUR	Tape & Reel	TQFP 48		
	ATUC64D4-Z1UT	Tray	QFN 48		
	ATUC64D4-Z1UR	Tape & Reel	QFN 48		

35. Errata

35.1 Rev. C

35.1.1 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. **PCS field in receive data register is inaccurate**
   The PCS field in the SPI_RDR register does not accurately indicate from which slave the received data is read.
   Fix/Workaround
   None.

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 the SR.TDRE bit whereas the write data command is filtered when SPI is disabled. Writing to TDR when SPI is disabled will not clear SR.TDRE. If SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR until its buffer is empty, and this data will be lost.
   Fix/Workaround
   Disable the PDCA, add two NOPs, and disable the SPI. To continue the transfer, enable the SPI and PDCA.

5. **SPI bad serial clock generation on 2nd chip_select when SCBR=1, CPOL=1, and NCPHA=0**
   When multiple chip selects (CS) are in use, if one of the baudrates equal 1 while one (CSRn.SCBR=1) of the others do not equal 1, and CSRn.CPOL=1 and CSRn.NCPHA=0, then an additional pulse will be generated on SCK.
   Fix/Workaround
   When multiple CS are in use, if one of the baudrates equals 1, the others must also equal 1 if CSRn.CPOL=1 and CSRn.NCPHA=0.

6. **Timer Counter**

7. **Channel chaining skips first pulse for upper channel**
   When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
   Fix/Workaround
   Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

35.1.2 TWIS

1. **Clearing the NAK bit before the BTF bit is set locks up the TWI bus**
   When the TWIS is in transmit mode, clearing the NAK Received (NAK) bit of the Status Register (SR) before the end of the Acknowledge/Not Acknowledge cycle will cause the TWIS to attempt to continue transmitting data, thus locking up the bus.
   Fix/Workaround
   Clear SR.NAK only after the Byte Transfer Finished (BTF) bit of the same register has been set.

35.1.3 PWMA

1. **The SR.READY bit cannot be cleared by writing to SCR.READY**
   The Ready bit in the Status Register will not be cleared when writing a one to the corresponding bit in the Status Clear register. The Ready bit will be cleared when the Busy bit is set.
   Fix/Workaround
   Disable the Ready interrupt in the interrupt handler when receiving the interrupt. When an operation that triggers the Busy/Ready bit is started, wait until the ready bit is low in the Status Register before enabling the interrupt.

35.2 Rev. B

35.2.1 Power Manager

1. **TWIS may not wake the device from sleep mode**
   If the CPU is put to a sleep mode (except Idle and Frozen) directly after a TWI Start condition, the CPU may not wake upon a TWIS address match. The request is NACKed.
   Fix/Workaround
   When using the TWI address match to wake the device from sleep, do not switch to sleep modes deeper than Frozen. Another solution is to enable asynchronous EIC wake on the TWIS clock (TWCK) or TWIS data (TWD) pins, in order to wake the system up on bus events.

35.2.2 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. **PCS field in receive data register is inaccurate**
   The PCS field in the SPI_RDR register does not accurately indicate from which slave the received data is read.
   Fix/Workaround
   None.

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 the SR.TDRE bit whereas the write data command is filtered when SPI is disabled. Writing to TDR when SPI is disabled will not clear SR.TDRE. If SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR until its buffer is empty, and this data will be lost.
   Fix/Workaround
   Disable the PDCA, add two NOPs, and disable the SPI. To continue the transfer, enable the SPI and PDCA.

5. **SPI bad serial clock generation on 2nd chip_select when SCBR=1, CPOL=1, and NCPHA=0**
   When multiple chip selects (CS) are in use, if one of the baudrates equal 1 while one (CSRn.SCBR=1) of the others do not equal 1, and CSRn.CPOL=1 and CSRn.NCPHA=0, then an additional pulse will be generated on SCK.
   Fix/Workaround
   When multiple CS are in use, if one of the baudrates equals 1, the others must also equal 1 if CSRn.CPOL=1 and CSRn.NCPHA=0.

6. **Timer Counter**

7. **Channel chaining skips first pulse for upper channel**
   When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
   Fix/Workaround
   Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

35.2.3 TWIS

1. **Clearing the NAK bit before the BTF bit is set locks up the TWI bus**
   When the TWIS is in transmit mode, clearing the NAK Received (NAK) bit of the Status Register (SR) before the end of the Acknowledge/Not Acknowledge cycle will cause the TWIS to attempt to continue transmitting data, thus locking up the bus.
   Fix/Workaround
   Clear SR.NAK only after the Byte Transfer Finished (BTF) bit of the same register has been set.

35.2.4 PWMA

1. **The SR.READY bit cannot be cleared by writing to SCR.READY**
   The Ready bit in the Status Register will not be cleared when writing a one to the corresponding bit in the Status Clear register. The Ready bit will be cleared when the Busy bit is set.
   Fix/Workaround
   Disable the Ready interrupt in the interrupt handler when receiving the interrupt. When an operation that triggers the Busy/Ready bit is started, wait until the ready bit is low in the Status Register before enabling the interrupt.

35.3 Rev. A

35.3.1 GPIO

1. **Clearing Interrupt flags can mask other interrupts**
   When clearing interrupt flags in a GPIO port, interrupts on other pins of that port, happening in the same clock cycle will not be registered.
   Fix/Workaround
   Read the PVR register of the port before and after clearing the interrupt to see if any pin change has happened while clearing the interrupt. If any change occurred in the PVR between the reads, they must be treated as an interrupt.

35.3.2 Power Manager

1. **Clock Failure Detector (CFD) can be issued while turning off the CFD**
   While turning off the CFD, the CFD bit in the Status Register (SR) can be set. This will change the main clock source to RCSYS.
   Fix/Workaround
   Solution 1: Enable CFD interrupt. If CFD interrupt is issues after turning off the CFD, switch back to original main clock source.
   Solution 2: Only turn off the CFD while running the main clock on RCSYS.

2. **Requesting clocks in idle sleep modes will mask all other PB clocks than the requested**
   In idle or frozen sleep mode, all the PB clocks will be frozen if the TWIS or the AST needs to wake the CPU up.
   Fix/Workaround
   Disable the TWIS or the AST before entering idle or frozen sleep mode.

3. **SPI**

4. **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).

5. **PCS field in receive data register is inaccurate**
   The PCS field in the SPI_RDR register does not accurately indicate from which slave the received data is read.
   Fix/Workaround
   None.

6. **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.

7. **Disabling SPI has no effect on the SR.TDRE bit**
   Disabling SPI has no effect on the SR.TDRE bit whereas the write data command is filtered when SPI is disabled. Writing to TDR when SPI is disabled will not clear SR.TDRE. If SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR until its buffer is empty, and this data will be lost.

Fix/Workaround
Disable the PDCA, add two NOPs, and disable the SPI. To continue the transfer, enable the SPI and PDCA.

8. **SPI bad serial clock generation on 2nd chip_select when SCBR=1, CPOL=1, and NCPHA=0**
When multiple chip selects (CS) are in use, if one of the baudrates equal 1 while one (CSRn.SCBR=1) of the others do not equal 1, and CSRn.CPOL=1 and CSRn.NCPHA=0, then an additional pulse will be generated on SCK.
Fix/Workaround
When multiple CS are in use, if one of the baudrates equals 1, the others must also equal 1 if CSRn.CPOL=1 and CSRn.NCPHA=0.

9. **I/O Pins**

10. **Current leakage through pads PA09, PA10 and PB16**
Pads PA09 (TWI), PA10 (TWI) and PB16 (USB VBUS) are not fully 5V tolerant. A leakage current can be observed when a 5V voltage is applied onto those pads inputs. Their behavior is normal at 3.3V
Fix/Workaround
None for pads PA09 and PA10. A voltage divider can be used for PB16 (VBUS) to bring the input voltage down into the 3.3V range.

11. **Current leakage through pads PB13, PB17 and PB18**
For applications in which UC3D is considered as a drop in replacement solution to UC3B, pads PB13, PB17 and PB18 can no longer be used as VDDCORE supply pins.Maintaining a 1.8V voltage on those inputs will however lead to a current over consumption through the pins.
Fix/Workaround
Do not connect PB13, PB17 and PB18 when using UC3D as a drop in replacement for a UC3B specific application.

12. **IO drive strength mismatch with UC3B specification for pads PA11, PA12, PA18 and PA19**
For applications in which UC3D is considered as a drop in replacement solution to UC3B, GPIOs PA11, PA12, PA18 and PA19 are not completely compatible in terms of drive strength. Those pads have a 8 mA current capability on UC3B, while this is limited to 4 mA in UC3D.
Fix/Workaround
None.

13. **WDT**

14. **Clearing the Watchdog Timer (WDT) counter in second half of timeout period will issue a Watchdog reset**
If the WDT counter is cleared in the second half of the timeout period, the WDT will immediately issue a Watchdog reset.
Fix/Workaround
Use twice as long timeout period as needed and clear the WDT counter within the first half of the timeout period. If the WDT counter is cleared after the first half of the timeout period, you will get a Watchdog reset immediately. If the WDT counter is not cleared at all, the time before the reset will be twice as long as needed.

35.3.3 Timer Counter

1. **Channel chaining skips first pulse for upper channel**
   When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
   Fix/Workaround
   Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

35.3.4 TWIS

1. **TWIS stretch on Address match error**
   When the TWIS stretches TWCK due to a slave address match, it also holds TWD low for the same duration if it is to be receiving data. When TWIS releases TWCK, it releases TWD at the same time. This can cause a TWI timing violation.
   Fix/Workaround
   None.

2. **Clearing the NAK bit before the BTF bit is set locks up the TWI bus**
   When the TWIS is in transmit mode, clearing the NAK Received (NAK) bit of the Status Register (SR) before the end of the Acknowledge/Not Acknowledge cycle will cause the TWIS to attempt to continue transmitting data, thus locking up the bus.
   Fix/Workaround
   Clear SR.NAK only after the Byte Transfer Finished (BTF) bit of the same register has been set.

3. **CAT**

4. **CAT module does not terminate QTouch burst on detect**
   The CAT module does not terminate a QTouch burst when the detection voltage is reached on the sense capacitor. This can cause the sense capacitor to be charged more than necessary. Depending on the dielectric absorption characteristics of the capacitor, this can lead to unstable measurements.
   Fix/Workaround
   Use the minimum possible value for the MAX field in the ATCFG1, TG0CFG1, and TG1CFG1 registers.

35.3.5 PWMA

1. **The SR.READY bit cannot be cleared by writing to SCR.READY**
   The Ready bit in the Status Register will not be cleared when writing a one to the corresponding bit in the Status Clear register. The Ready bit will be cleared when the Busy bit is set.
   Fix/Workaround
   Disable the Ready interrupt in the interrupt handler when receiving the interrupt. When an operation that triggers the Busy/Ready bit is started, wait until the ready bit is low in the Status Register before enabling the interrupt.

2.

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. A – 11/2009

1. Initial revision.

36.2 Rev. B – 04/2011

1. Minor.

36.3 Rev. C – 07/2011

1. Final revision.

36.4 Rev. D – 11/2011

1. Adding errata for silicon Revision C .
2. Fixed PLLOPT field description in SCIF chapter

Table of Contents

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.

© 2011 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.

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for 32-bit Microcontrollers - MCU ***category:***

Click to view products by Microchip ***manufacturer:***

Other Similar products are found below :

MB91F575BHSPMC-GSE1 MB91F594BSPMC-GSE1 PIC32MX120F032B-50I/ML MB91F464AAPMC-GSE2 MB91F577BHSPMC-GSE1 MB91F528USCPMC-GSE2 MB91F248PFV-GE1 MB91F594BPMC-GSE1 MB91243PFV-GS-136E1 MB91F577BHSPMC1-GSE1 PIC32MM0032GPL020-E/ML PIC32MM0016GPL028-E/SS PIC32MM0016GPL028-E/ML PIC32MM0032GPL028-E/ML PIC32MM0032GPL028-E/M6 SPC5604BF2VLL6 MB91F526KSEPMC-GSE1 TLE9872QTW40XUMA1 FT902L-T R5F564MLCDFB#31 R5F564MLCDFC#31 R5F523E5ADFL#30 R5F524TAADFF#31 MCF51AC256ACPUE PIC32MX150F128D-I/ML PIC32MX230F064D-I/PT PIC32MM0064GPL028-I/ML PIC32MM0064GPL028-I/SP PIC32MM0064GPL028-I/SO PIC32MX120F032D-I/TL PIC32MX130F064D-I/ML PIC32MZ2064DAB169-I/HF PIC32MZ2064DAB288-I/4J ATUC256L4U-AUT R5F56318CDBG#U0 PIC32MX150F128C-I/TL PIC32MX130F064C-ITL PIC32MX230F064D-IML PIC32MX154F128D-I/PT PIC32MX154F128B-V/SO AT32UC3L0128-AUT PIC32MX254F128B-I/SO PIC32MX230F128H-I/MR PIC32MX150F128D-50I/TL PIC32MZ1064DAB288-I/4J PIC32MZ1064DAB169-I/HF ATUC64D4-Z1UT AT32UC3A3128S-CTUT ATUC128L3U-Z3UT ATUC64L3U-Z3UT