Atmel

Atmel SAM D09

SMART ARM-Based Microcontroller

DATASHEET

Description

The Atmel® | SMART™ SAM D09 is a series of low-power microcontrollers using the 32-bit ARM® Cortex®-M0+ processor, and ranging from 14- to 24-pins with up to16KB Flash and 4KB of SRAM. The SAM D09 devices operate at a maximum frequency of 48MHz and reach 2.46 Coremark/MHz. They are designed for simple and intuitive migration with identical peripheral modules, hex compatible code, identical linear address map and pin compatible migration paths between all devices in the product series. All devices include intelligent and flexible peripherals, Atmel Event System for inter-peripheral signaling, and support for capacitive touch button, slider and wheel user interfaces. The SAM D09 series is compatible to the other product series in the SAM D family, enabling easy migration to larger device with added features.

The Atmel SAM D09 devices provide the following features: In-system programmable Flash, six-channel direct memory access (DMA) controller, 6 channel Event System, programmable interrupt controller, up to 22 programmable I/O pins, 32-bit real-time clock and calendar, two 16-bit Timer/Counters (TC), where each TC can be configured to perform frequency and waveform generation, accurate program execution timing or input capture with time and frequency measurement of digital signals. The TCs can operate in 8- or 16-bit mode, selected TCs can be cascaded to form a 32-bit TC, and one timer/counter has extended functions optimized for motor, lighting and other control applications. The series provide two Serial Communication Modules (SERCOM) that each can be configured to act as an USART, UART, SPI, I^2C, SMBus, PMBus and LIN slave; up to 10-channel 350ksps 12-bit ADC with programmable gain and optional oversampling and decimation supporting up to 16-bit resolution, programmable Watchdog Timer, brown-out detector and power-on reset and two-pin Serial Wire Debug (SWD) program and debug interface.

All devices have accurate and low-power external and internal oscillators. All oscillators can be used as a source for the system clock. Different clock domains can be independently configured to run at different frequencies, enabling power saving by running each peripheral at its optimal clock frequency, and thus maintaining a high CPU frequency while reducing power consumption.

The SAM D09 devices have two software-selectable sleep modes, idle and standby. In idle mode the CPU is stopped while all other functions can be kept running. In standby all clocks and functions are stopped expect those selected to continue running. The device supports SleepWalking. This feature allows the peripheral to wake up from sleep based on predefined conditions, and thus allows the CPU to wake up only when needed, e.g. when a threshold is crossed or a result is ready. The Event System supports synchronous and asynchronous events, allowing peripherals to receive, react to and send events even in standby mode.

The Flash program memory can be reprogrammed in-system through the SWD interface. The same interface can be used for non-intrusive on-chip debug and trace of application code. A boot loader running in the device can use any communication interface to download and upgrade the application program in the Flash memory.

The Atmel SAM D09 devices are supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, programmers and evaluation kits.

Features

- Processor
  - ARM Cortex-M0+ CPU running at up to 48MHz
    - Single-cycle hardware multiplier
    - Micro Trace Buffer
- Memories
  - 8/16KB in-system self-programmable Flash
  - 4KB SRAM Memory
- System
  - Power-on reset (POR) and brown-out detection (BOD)
  - Internal and external clock options with 48MHz Digital Frequency Locked Loop (DFLL48M) and 48MHz to 96MHz Fractional Digital Phase Locked Loop (FDPLL96M)
  - External Interrupt Controller (EIC)
  - 8 external interrupts
  - One non-maskable interrupt
  - Two-pin Serial Wire Debug (SWD) programming, test and debugging interface
- Low Power
  - Idle and standby sleep modes
  - SleepWalking peripherals
- Peripherals
  - 6-channel Direct Memory Access Controller (DMAC)
  - 6-channel Event System
  - Two 16-bit Timer/Counters (TC), configurable as either:
    - One 16-bit TC with compare/capture channels
    - One 8-bit TC with compare/capture channels
    - One 32-bit TC with compare/capture channels, by using two TCs
  - 32-bit Real Time Counter (RTC) with clock/calendar function
  - Watchdog Timer (WDT)
  - CRC-32 generator
  - Two Serial Communication Interfaces (SERCOM), each configurable to operate as either:
    - USART with full-duplex and single-wire half-duplex configuration
    - I^2C Bus
    - SMBUS/PMBUS
    - SPI
    - LIN slave
  - 12-bit, 350ksps Analog-to-Digital Converter (ADC) with up to 10 channels
    - Differential and single-ended input
    - 1/2x to 16x programmable gain stage
    - Automatic offset and gain error compensation
    - Oversampling and decimation in hardware to support 13-, 14-, 15- or 16-bit resolution
- I/O
  - Up to 22 programmable I/O pins
- Packages
  - 24-pin QFN
  - 14-pin SOIC
- Operating Voltage
  - 2.4V – 3.63V

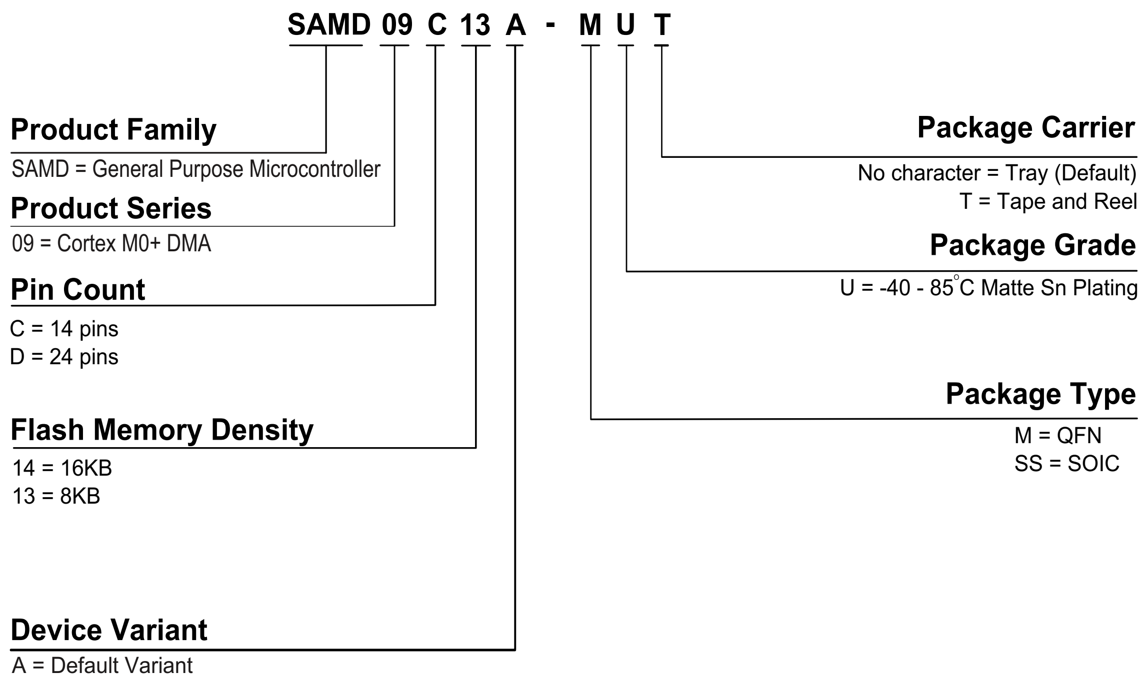
1. Configuration Summary

Table 1-1. Configuration Summary

	SAM D09D – 24-pin QFN	SAM D09C – 14-pin SOIC
Pins	24	14
General Purpose I/O-pins (GPIOs)	22	12
Flash	16KB	8KB
SRAM	4KB	4KB
Timer Counter (TC)	2	2[1]
Waveform output channels for TC	2	2
DMA channels	6	6
Serial Communication Interface (SERCOM)	2	2
Analog-to-Digital Converter (ADC) channels	10	5
Real-Time Counter (RTC)	Yes	Yes
RTC alarms	1	1
RTC compare values	1 32-bit value or 2 16-bit values	1 32-bit value or 2 16-bit values
External Interrupt lines	8	8
Maximum CPU frequency	48MHz	48MHz
Packages	QFN	SOIC
Oscillators	32.768kHz crystal oscillator (XOSC32K) 0.4-32MHz crystal oscillator (XOSC) 32.768kHzinternal oscillator (OSC32K) 32kHz ultra-low-power internal oscillator (OSCULP32K) 8MHz high-accuracy internal oscillator (OSC8M) 48MHz Digital Frequency Locked Loop (DFLL48M) 96MHz Fractional Digital Phased Locked Loop (FDPLL96M)	
Event System channels	6	6
SW Debug Interface	Yes	Yes
Watchdog Timer (WDT)	Yes	Yes

Note: 1. The signals for TC2 are not routed out on the 14-pin package.

2. Ordering Information



2.1 SAM D09C – 14-pin SOIC

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type
ATSAMD09C13A-SSUT	8K	4K	SOIC14	Tape & Reel

2.2 SAM D09D – 24-pin QFN

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type
ATSAMD09D14A-MUT	16K	4K	QFN24	Tape & Reel

2.3 Device Identification

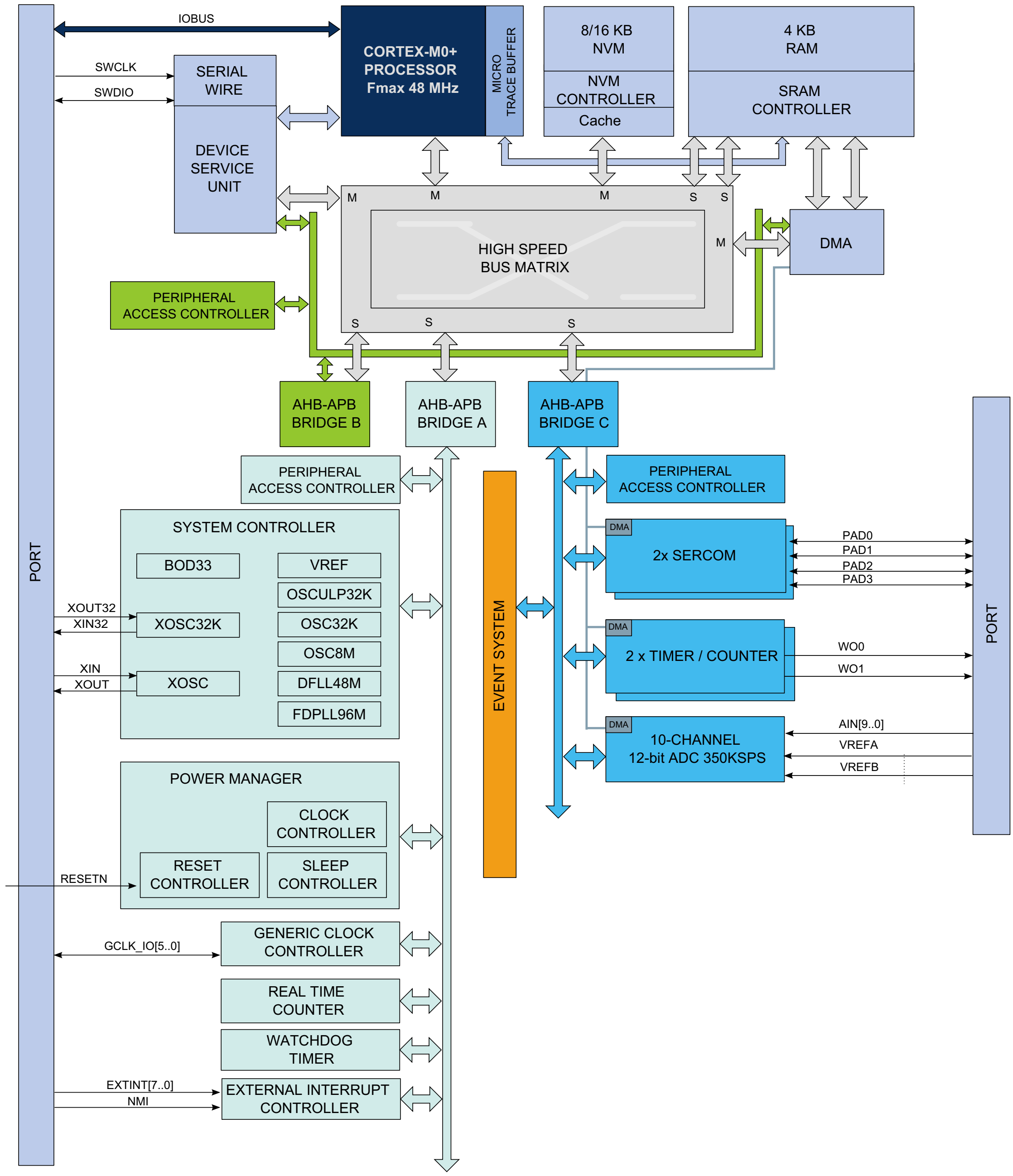
The DSU - Device Service Unit peripheral provides the Device Selection bits in the Device Identification register (DID.DEVSEL) in order to identify the device by software. The device variants have a reset value of DID=0x1001drxx, with the LSB identifying the die number ('d'), the die revision ('r') and the device selection ('xx').

Table 2-1. Device Identification Values

Device Variant	DID.DEVSEL	Device ID (DID)
SAMD09D14AM	0x00	0x10040r00
Reserved	0x01 - 0x06	
SAMD09C13A	0x07	0x10040r07

Note: The device variant (last letter of the ordering number) is independent of the die revision (DSU.DID.REVISION): The device variant denotes functional differences, whereas the die revision marks evolution of the die. The device variant denotes functional differences, whereas the die revision marks evolution of the die

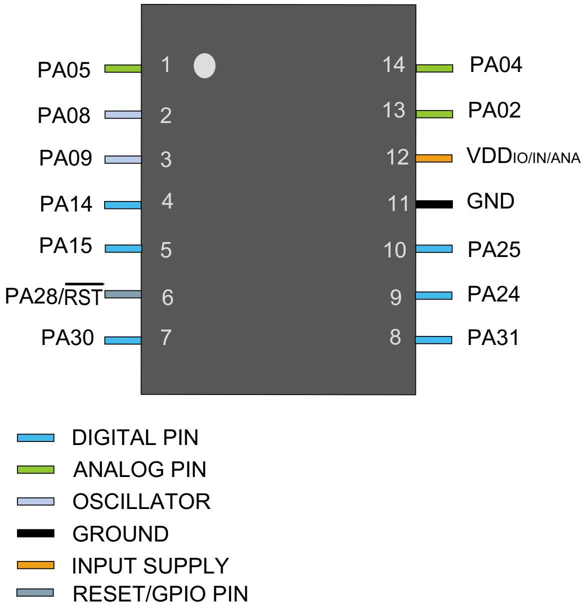
3. Block Diagram



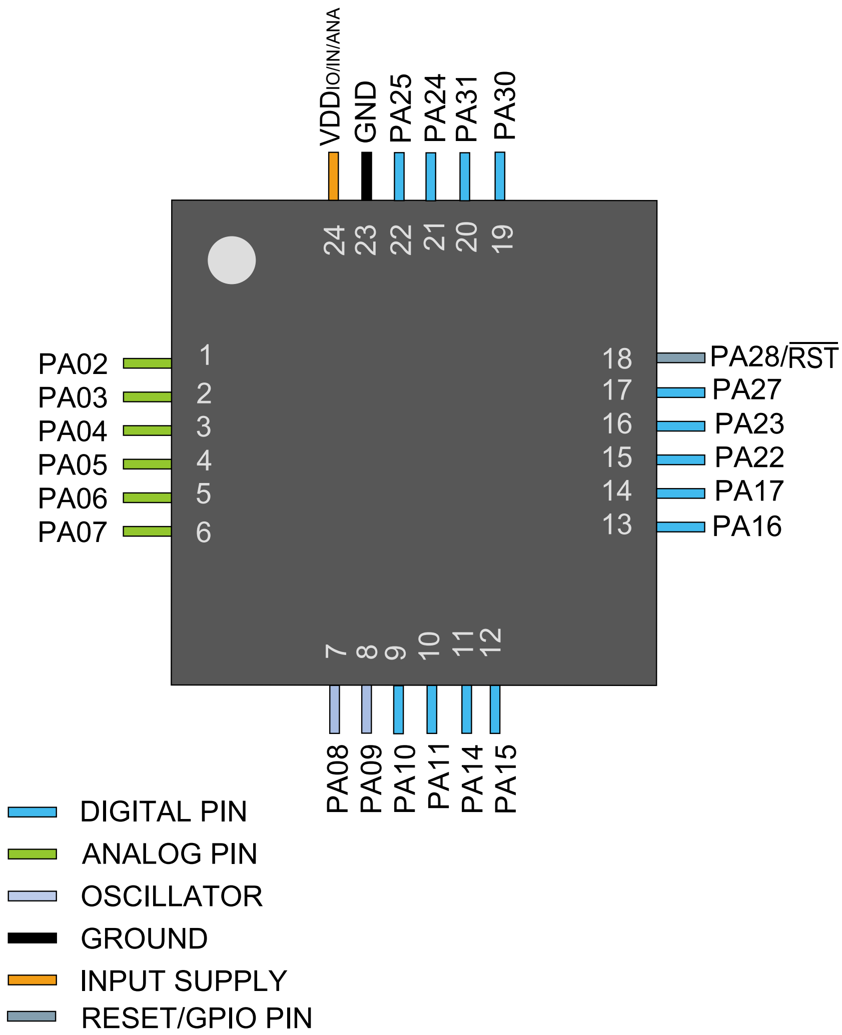
2. Some products have different number of SERCOM instances and ADC signals. Refer to "I/O Multiplexing and Considerations" on page 10 for details.

4. Pinout

4.1 SAM D09C 14-pin SOIC



4.2 SAM D09D 24-pin QFN



5. Signal Descriptions List

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

Table 5-1. Signal Descriptions List

Signal Name	Function	Type	Active Level
Analog Digital Converter - ADC			
AIN[19:0]	ADC Analog Inputs	Analog	
VREFA	ADC Voltage External Reference A	Analog	
VREFB	ADC Voltage External Reference B	Analog	
External Interrupt Controller			
EXTINT[7:0]	External Interrupts	Input	
NMI	External Non-Maskable Interrupt	Input	
Generic Clock Generator - GCLK			
GCLK_IO[5:0]	Generic Clock (source clock or generic clock generator output)	I/O	
Power Manager - PM			
$\overline{\text{RESET}}$	Reset	Input	Low
Serial Communication Interface - SERCOMx			
PAD[3:0]	SERCOM I/O Pads	I/O	
System Control - SYSCTRL			
XIN	Crystal Input	Analog/ Digital	
XIN32	32kHz Crystal Input	Analog/ Digital	
XOUT	Crystal Output	Analog	
XOUT32	32kHz Crystal Output	Analog	
Timer Counter - TCx			
WO[1:0]	Waveform Outputs	Output	
General Purpose I/O - PORT			
PA25 - PA00	Parallel I/O Controller I/O Port A	I/O	
PA28 - PA27	Parallel I/O Controller I/O Port A	I/O	
PA31 - PA30	Parallel I/O Controller I/O Port A	I/O	
PB17 - PB00	Parallel I/O Controller I/O Port B	I/O	
PB23 - PB22	Parallel I/O Controller I/O Port B	I/O	
PB31 - PB30	Parallel I/O Controller I/O Port B	I/O	

6. I/O Multiplexing and Considerations

6.1 Multiplexed Signals

Each pin is by default controlled by the PORT as a general purpose I/O and alternatively it can be assigned to one of the peripheral functions A, B, C, D, E, G or H. To enable a peripheral function on a pin, the Peripheral Multiplexer Enable bit in the Pin Configuration register corresponding to that pin (PINCFGn.PMUXEN, n = 0-31) in the PORT must be written to one. The selection of peripheral function A to H is done by writing to the Peripheral Multiplexing Odd and Even bits in the Peripheral Multiplexing register (PMUXn.PMUXE/O) in the PORT.

Table 5-1 on page 11 describes the peripheral signals multiplexed to the PORT I/O pins.

Table 6-1. PORT Function Multiplexing

Pin					A	B		C	D	E	G	H
SAMD09C 14-pin SOIC	SAMD09D 24-pin QFN	I/O Pin	Supply	Type	EIC	REF	ADC	SERCOM(1)	SERCOM-ALT	TC	COM	GCLK
13	1	PA02	VDD		EXTINT[2]		AIN[0]					
	2	PA03	VDD		EXTINT[3]	ADC/VREFA	AIN[1]					
14	3	PA04	VDD		EXTINT[4]	ADC/VREFB	AIN[2]	SERCOM0/PAD[2]	SERCOM0/PAD[0]	TC1/WO[0]		
1	4	PA05	VDD		EXTINT[5]		AIN[3]	SERCOM0/PAD[3]	SERCOM0/PAD[1]	TC1/WO[1]		
	5	PA06	VDD		EXTINT[6]		AIN[4]	SERCOM0/PAD[0]	SERCOM0/PAD[2]	TC2/WO[0]		
	6	PA07	VDD		EXTINT[7]		AIN[5]	SERCOM0/PAD[1]	SERCOM0/PAD[3]	TC2/WO[1]		
2	7	PA08	VDD		EXTINT[6]			SERCOM1/PAD[2]	SERCOM0/PAD[2]			GCLK_IO[0]
3	8	PA09	VDD		EXTINT[7]			SERCOM1/PAD[3]	SERCOM0/PAD[3]			GCLK_IO[1]
	9	PA10	VDD		EXTINT[2]		AIN[8]	SERCOM0/PAD[2]		TC2/WO[0]		GCLK_IO[4]
	10	PA11	VDD		EXTINT[3]		AIN[9]	SERCOM0/PAD[3]		TC2/WO[1]		GCLK_IO[5]
4	11	PA14	VDD	I^2C	NMI		AIN[6]	SERCOM0/PAD[0]		TC1/WO[0]		GCLK_IO[4]
5	12	PA15	VDD	I^2C	EXTINT[1]		AIN[7]	SERCOM0/PAD[1]		TC1/WO[1]		GCLK_IO[5]
	13	PA16	VDD		EXTINT[0]			SERCOM1/PAD[2]		TC1/WO[0]		GCLK_IO[2]
	14	PA17	VDD		EXTINT[1]			SERCOM1/PAD[3]		TC1/WO[1]		GCLK_IO[3]
	15	PA22	VDD	I^2C	EXTINT[6]			SERCOM1/PAD[0]		TC1/WO[0]		GCLK_IO[1]
	16	PA23	VDD	I^2C	EXTINT[7]			SERCOM1/PAD[1]		TC1/WO[1]		GCLK_IO[2]
	17	PA27	VDD		EXTINT[7]							GCLK_IO[0]
6	18	PA28	VDD									
7	19	PA30	VDD		EXTINT[2]			SERCOM1/PAD[0]	SERCOM1/PAD[2]	TC2/WO[0]	CORTEX_M0P/SWCLK	GCLK_IO[0]
8	20	PA31	VDD		EXTINT[3]			SERCOM1/PAD[1]	SERCOM1/PAD[3]	TC2/WO[1]	SWDIO(2)	GCLK_IO[0]
9	21	PA24	VDD		EXTINT[4]			SERCOM1/PAD[2]				GCLK_IO[0]
10	22	PA25	VDD		EXTINT[5]			SERCOM1/PAD[3]				GCLK_IO[0]

Notes:
1. Refer to "Electrical Characteristics" on page 648 for details on the I^2C pin characteristics.Only some pins can be used in SERCOM I^2C mode. See the Type column for using a SERCOM pin in I^2C mode
2. This function is only activated in the presence of a debugger.
3. If the PA24 and PA25 pins are not connected, it is recommended to enable a pull-up on PA24 and PA25 through input GPIO mode. The aim is to avoid an eventually extract power consumption (<1mA) due to a not stable level on pad.

6.2 Other Functions

6.2.1 Oscillator Pinout

The oscillators are not mapped to the normal PORT functions and their multiplexing are controlled by registers in the System Controller (SYSCTRL).

Oscillator	Supply	Signal	I/O Pin
XOSC	$VDD_{IO/IN/ANA}$	XIN	PA08
		XOUT	PA09
XOSC32K	$VDD_{IO/IN/ANA}$	XIN32	PA08
		XOUT32	PA09

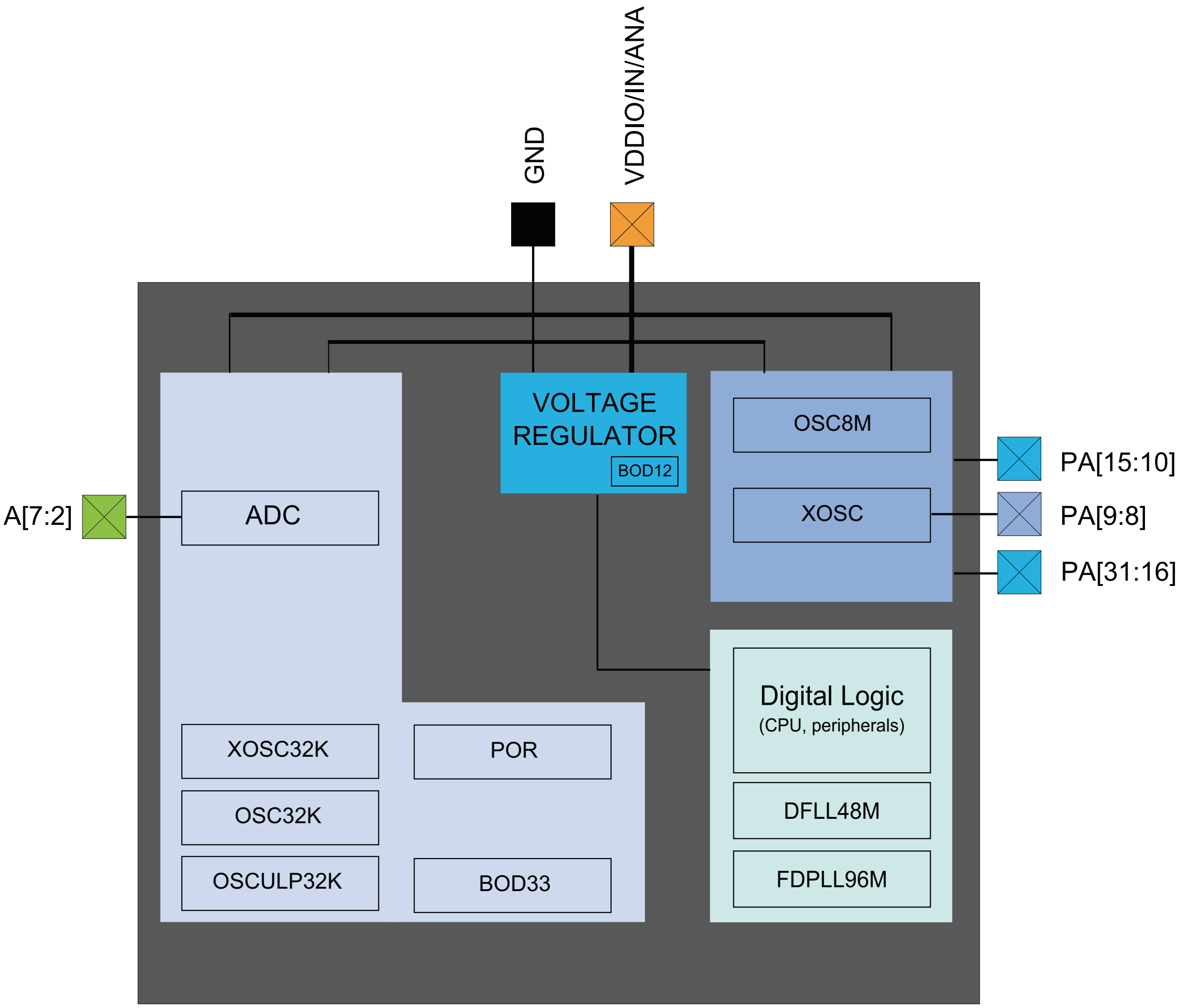
6.2.2 Serial Wire Debug Interface Pinout

Only the SWCLK pin is mapped to the normal PORT functions. A debugger cold-plugging or hot-plugging detection will automatically switch the SWDIO port to the SWDIO function.

Signal	Supply	I/O Pin
SWCLK	VDDIO	PA30
SWDIO	VDDIO	PA31

7. Power Supply and Start-Up Considerations

7.1 Power Domain Overview



7.2 Power Supply Considerations

7.2.1 Power Supplies

The Atmel® SAMD09 has single power supply pins

- VDDIO/IN/ANA: Powers I/O lines, OSC8M and XOSC, the internal regulator, ADC, OSCULP32K, OSC32K, XOSC32K. Voltage is 2.4V to 3.63V.
- Internal regulated voltage output. Powers the core, memories, peripherals, DFLL48M and FDPLL96M. Voltage is 1.2V.

The ground pin is GND.

7.2.2 Voltage Regulator

The voltage regulator has two different modes:

- Normal mode: To be used when the CPU and peripherals are running

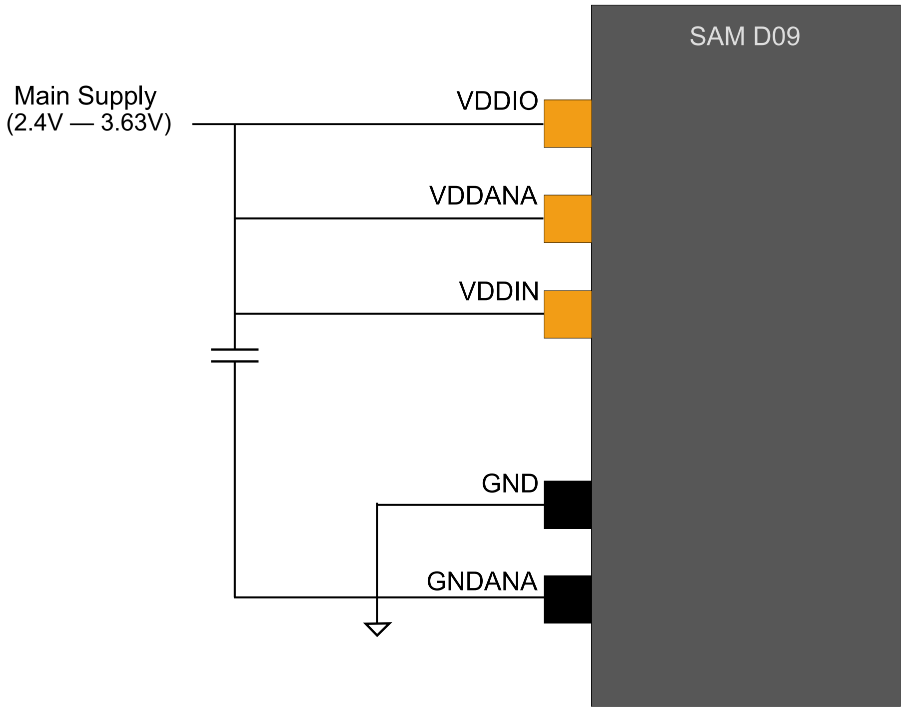
- Low Power (LP) mode: To be used when the regulator draws small static current. It can be used in standby mode

7.2.3 Typical Powering Schematics

The SAM D09 uses a single supply from 2.4V to 3.63V.

The following figure shows the recommended power supply connection.

Figure 7-1. Power Supply Connection



7.2.4 Power-Up Sequence

7.2.4.1 Minimum Rise Rate

The integrated power-on reset (POR) circuitry monitoring the VDDIO/IN/ANA power supply requires a minimum rise rate. Refer to the "Electrical Characteristics" on page 648 for details.

7.2.4.2 Maximum Rise Rate

The rise rate of the power supply must not exceed the values described in Electrical Characteristics. Refer to the "Electrical Characteristics" on page 648 for details.

7.3 Power-Up

This section summarizes the power-up sequence of the SAM D09. The behavior after power-up is controlled by the Power Manager. Refer to "PM – Power Manager" on page 107 for details.

7.3.1 Starting of Clocks

After power-up, the device is set to its initial state and kept in reset, until the power has stabilized throughout the device. Once the power has stabilized, the device will use a 1MHz clock. This clock is derived from the 8MHz Internal Oscillator (OSC8M), which is divided by eight and used as a clock source for generic clock generator 0. Generic clock generator 0 is the main clock for the Power Manager (PM).

Some synchronous system clocks are active, allowing software execution.

Refer to the “Clock Mask Register” section in “PM – Power Manager” on page 107 for the list of default peripheral clocks running. Synchronous system clocks that are running are by default not divided and receive a 1MHz clock through generic clock generator 0. Other generic clocks are disabled except GCLK_WDT, which is used by the Watchdog Timer (WDT).

7.3.2 I/O Pins

After power-up, the I/O pins are tri-stated.

7.3.3 Fetching of Initial Instructions

After reset has been released, the CPU starts fetching PC and SP values from the reset address, which is 0x00000000. This address points to the first executable address in the internal flash. The code read from the internal flash is free to configure the clock system and clock sources. Refer to “PM – Power Manager” on page 107, “GCLK – Generic Clock Controller” on page 85 and “SYSCTRL – System Controller” on page 140 for details. Refer to the ARM Architecture Reference Manual for more information on CPU startup (http://www.arm.com).

7.4 Power-On Reset and Brown-Out Detector

The SAM D09 embeds three features to monitor, warn and/or reset the device:

- POR: Power-on reset on VDDIO/IN/ANA
- BOD33: Brown-out detector
- BOD12: Voltage Regulator Internal Brown-out detector on VDDCORE. The Voltage Regulator Internal BOD is calibrated in production and its calibration configuration is stored in the NVM User Row. This configuration should not be changed if the user row is written to assure the correct behavior of the BOD12.

7.4.1 Power-On Reset on VDDIO/IN/ANA

POR monitors VDDIO/IN/ANA. It is always activated and monitors voltage at startup and also during all the sleep modes. If VDDIO/IN/ANA goes below the threshold voltage, the entire chip is reset.

7.4.2 Brown-Out Detector

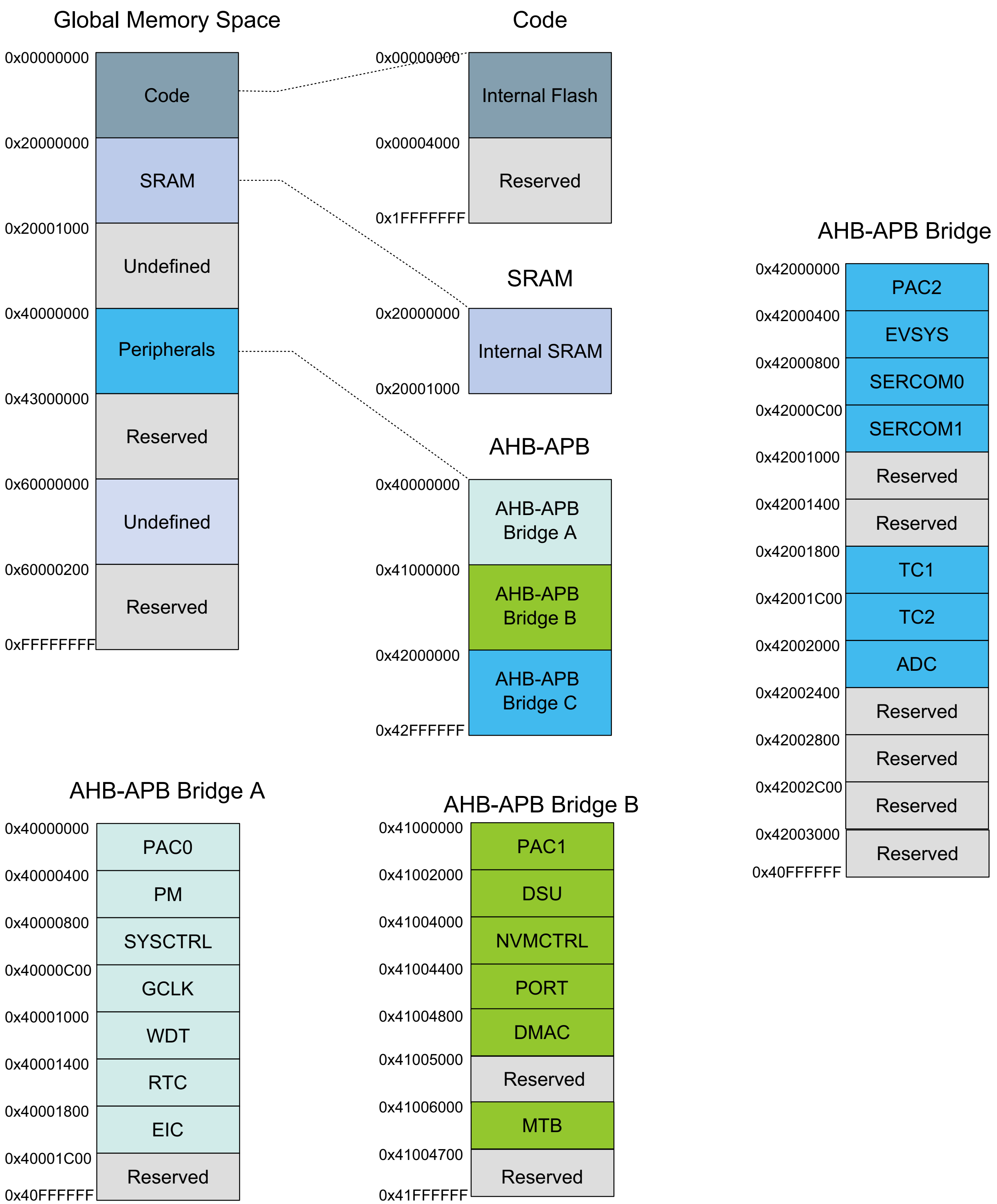
BOD33 monitors 3.3V. Refer to “SYSCTRL – System Controller” on page 140 for details.

7.4.3 Brown-Out Detector on VDDCORE

Once the device has started up, BOD12 monitors the internal VDDCORE.

8. Product Mapping

Figure 8-1. Atmel SAM D09 Product Mapping



9. Memories

9.1 Embedded Memories

- Internal high-speed flash
- Internal high-speed RAM, single-cycle access at full speed
- Dedicated flash area for EEPROM emulation

9.2 Physical Memory Map

The High-Speed bus is implemented as a bus matrix. All High-Speed bus addresses are fixed, and they are never remapped in any way, even during boot. The 32-bit physical address space is mapped as follow:

Table 9-1. SAM D09 physical memory map

Memory	Start address	Size	
		SAMD09x14	SAMD09x13
Embedded Flash	0x00000000	16Kbytes	8Kbytes
Embedded SRAM	0x20000000	4Kbytes	4Kbytes
Peripheral Bridge A	0x40000000	64Kbytes	64Kbytes
Peripheral Bridge B	0x41000000	64Kbytes	64Kbytes
Peripheral Bridge C	0x42000000	64Kbytes	64Kbytes

Table 9-2. Flash memory parameters

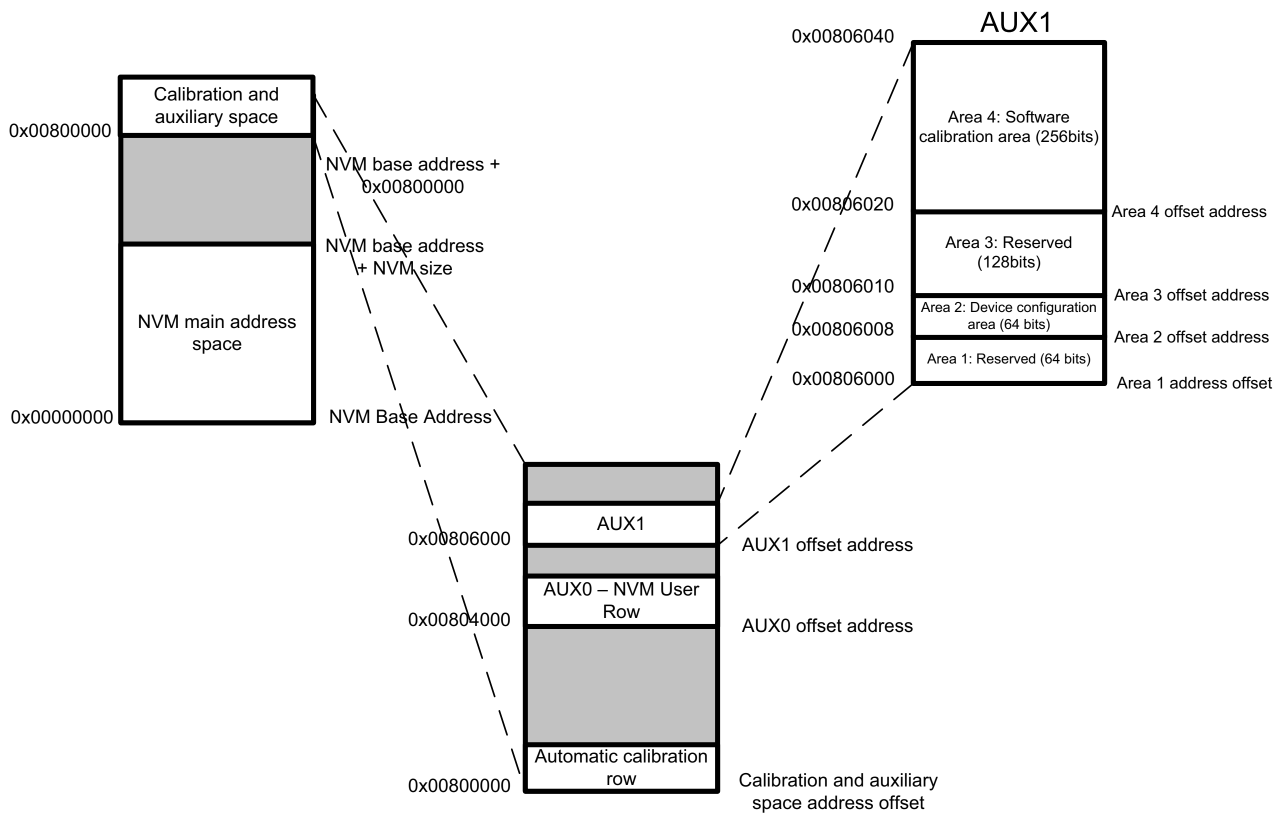
Device	Flash size (FLASH_PM)	Number of pages (FLASH_P)	Page size (FLASH_W)
SAMD09x14	16Kbytes	256	64 bytes
SAMD09x13	8Kbytes	128	64 bytes

Note: 1. x = C, D

9.3 NVM Calibration and Auxiliary Space

The device calibration data are stored in different sections of the NVM calibration and auxiliary space presented in the following figure.

Figure 9-1. Calibration and Auxiliary space



The values from the automatic calibration row is loaded into their respective registers on startup.

9.4 NVM User Row Mapping

The NVM User Row contains calibration data that are automatically read at device power on.

The NVM User Row can be read at address 0x804000.

To write the NVM User Row refer to "NVMCTRL – Non-Volatile Memory Controller" on page 352.

Note that when writing to the user row the values do not get loaded by the other modules on the device until a device reset occurs.

Table 9-3. NVM User Row Mapping

Bit Position	Name	Usage	Production setting
2:0	BOOTPROT	Used to select one of eight different bootloader sizes. Refer to "NVMCTRL – Non-Volatile Memory Controller" on page 352.	7
3	Reserved		1
6:4	EEPROM	Used to select one of eight different EEPROM sizes. Refer to "NVMCTRL – Non-Volatile Memory Controller" on page 352.	7
7	Reserved		1
13:8	BOD33 Level	BOD33 Threshold Level at power on. Refer to SYSCTRL.BOD33 register.	7
14	BOD33 Enable	BOD33 Enable at power on. Refer to SYSCTRL.BOD33 register.	1
16:15	BOD33 Action	BOD33 Action at power on. Refer to SYSCTRL.BOD33 register.	1
24:17	Reserved	Voltage Regulator Internal BOD(BOD12) configuration. These bits are written in production and must not be changed. Default value = 0x70.	0x70
25	WDT Enable	WDT Enable at power on. Refer to WDT.CTRL register.	0
26	WDT Always-On	WDT Always-On at power on. Refer to WDT.CTRL register.	0
30:27	WDT Period	WDT Period at power on. Refer to WDT.CONFIG register.	0xB
34:31	WDT Window	WDT Window mode time-out at power on. Refer to WDT.CONFIG register. Default value, WINDOW_1 = 0x5	0xB
38:35	WDT EWOFFSET	WDT Early Warning Interrupt Time Offset at power on. Refer to WDT.EWCTRL register.	0xB
39	WDT WEN	WDT Timer Window Mode Enable at power on. Refer to WDT.CTRL register.	0
40	BOD33 Hysteresis	BOD33 Hysteresis configuration at power on. Refer to SYSCTRL.BOD33 register.	0
41	Reserved	Voltage Regulator Internal BOD(BOD12) configuration. This bit is written in production and must not be changed. Default value = 0.	0
47:42	Reserved		0x3F
63:48	LOCK	NVM Region Lock Bits. Refer to "NVMCTRL – Non-Volatile Memory Controller" on page 352.	0xFFFF

9.5 NVM Software Calibration Row Mapping

The NVM Software Calibration Row contains calibration data that are measured and written during production test. These calibration values should be read by the application software and written back to the corresponding register.

The NVM Software Calibration Row can be read at address 0x806020.

The NVM Software Calibration Row can not be written.

Table 9-4. NVM Software Calibration Row Mapping

Bit Position	Name	Description
2:0	Reserved	
14:3	Reserved	
26:15	Reserved	
34:27	ADC LINEARITY	ADC Linearity Calibration. Should be written to CALIB register.
37:35	ADC BIASCAL	ADC Bias Calibration. Should be written to CALIB register.
44:38	OSC32K CAL	OSC32KCalibration. Should be written to OSC32K register.
63:58	DFLL48M COARSE CAL	DFLL48M Coarse calibration value. Should be written to the SYSCTRL DFLLVAL register.
73:64	Reserved	
127:74	Reserved	

9.6 Serial Number

Each device has a unique 128-bit serial number which is a concatenation of four 32-bit words contained at the following addresses:

Word 0: 0x0080A00C

Word 1: 0x0080A040

Word 2: 0x0080A044

Word 3: 0x0080A048

The uniqueness of the serial number is guaranteed only when using all 128 bits.

10. Processor And Architecture

10.1 Cortex M0+ Processor

The Atmel SAM D09 implements the ARM® Cortex™-M0+ processor, based on the ARMv6 Architecture and Thumb®-2 ISA. The Cortex M0+ is 100% instruction set compatible with its predecessor, the Cortex-M0 core, and upward compatible to Cortex-M3 and M4 cores. The ARM Cortex-M0+ implemented is revision r0p1. For more information refer to www.arm.com.

10.1.1 Cortex M0+ Configuration

Table 10-1. Cortex M0+ Configuration

Features	Configurable option	Atmel SAM D09 configuration
Interrupts	External interrupts 0-32	32
Data endianness	Little-endian or big-endian	Little-endian
SysTick timer	Present or absent	Present
Number of watchpoint comparators	0, 1, 2	2
Number of breakpoint comparators	0, 1, 2, 3, 4	4
Halting debug support	Present or absent	Present
Multiplier	Fast or small	Fast (single cycle)
Single-cycle I/O port	Present or absent	Present
Wake-up interrupt controller	Supported or not supported	Not supported
Vector Table Offset Register	Present or absent	Present
Unprivileged/Privileged support	Present or absent	Absent(1)
Memory Protection Unit	Not present or 8-region	Not present
Reset all registers	Present or absent	Absent
Instruction fetch width	16-bit only or mostly 32-bit	32-bit

Note: 1. All software run in privileged mode only.

The ARM Cortex-M0+ core has two bus interfaces:

- Single 32-bit AMBA-3 AHB-Lite system interface that provides connections to peripherals and all system memory, which includes flash and RAM
- Single 32-bit I/O port bus interfacing to the PORT with 1-cycle loads and stores

10.1.2 Cortex-M0+ Peripherals

- System Control Space (SCS)
  - The processor provides debug through registers in the SCS. Refer to the Cortex-M0+ Technical Reference Manual for details (www.arm.com).
- System Timer (SysTick)
  - The System Timer is a 24-bit timer that extends the functionality of both the processor and the NVIC. Refer to the Cortex-M0+ Technical Reference Manual for details (www.arm.com).

- Nested Vectored Interrupt Controller (NVIC)
  - External interrupt signals connect to the NVIC, and the NVIC prioritizes the interrupts. Software can set the priority of each interrupt. The NVIC and the Cortex-M0+ processor core are closely coupled, providing low latency interrupt processing and efficient processing of late arriving interrupts. Refer to "Nested Vector Interrupt Controller" on page 23 and the Cortex-M0+ Technical Reference Manual for details (www.arm.com.).
- System Control Block (SCB)
  - The System Control Block provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions. Refer to the Cortex-M0+ Devices Generic User Guide for details (www.arm.com.).

10.1.3 Cortex-M0+ Address Map

Table 10-2. Cortex-M0+ Address Map

Address	Peripheral
0xE000E000	System Control Space (SCS)
0xE000E010	System Timer (SysTick)
0xE000E100	Nested Vectored Interrupt Controller (NVIC)
0xE000ED00	System Control Block (SCB)

10.1.4 I/O Interface

10.1.4.1 Overview

Because accesses to the AMBA® AHB-Lite™ and the single cycle I/O interface can be made concurrently, the Cortex-M0+ processor can fetch the next instructions while accessing the I/Os. This enables single cycle I/O accesses to be sustained for as long as needed.

10.1.4.2 Description

Direct access to PORT registers.

10.2 Nested Vector Interrupt Controller

10.2.1 Overview

The ARMv6-M Nested Vectored Interrupt Controller (NVIC) in Atmel SAM D09 supports 32 external interrupts with 4 different priority levels. For more details refer to the Cortex-M0 Technical Reference Manual.

10.2.2 Interrupt Line Mapping

Table 10-3. Interrupt Line Mapping

Peripheral Source	NVIC Line
EIC NMI – External Interrupt Controller	NMI
PM – Power Manager	0
SYSCTRL – System Control	1
WDT – Watchdog Timer	2
RTC – Real Time Clock	3
EIC – External Interrupt Controller	4
NVMCTRL – Non-Volatile Memory Controller	5
DMAC - Direct Memory Access Controller	6
Reserved	7
EVSYS – Event System	8
SERCOM0 – Serial Communication Controller 0	9
SERCOM1 – Serial Communication Controller 1	10
Reserved	11
Reserved	12
TC1 – Timer Counter 1	13
TC2 – Timer Counter 2	14
ADC – Analog-to-Digital Converter	15
Reserved	16
Reserved	17
Reserved	18

10.3 High-Speed Bus System

10.3.1 Features

High-Speed Bus Matrix has the following features:

- Symmetric crossbar bus switch implementation
- Allows concurrent accesses from different masters to different slaves
- 32-bit data bus
- Operation at a 1-to-1 clock frequency with the bus masters

10.3.2 Configuration

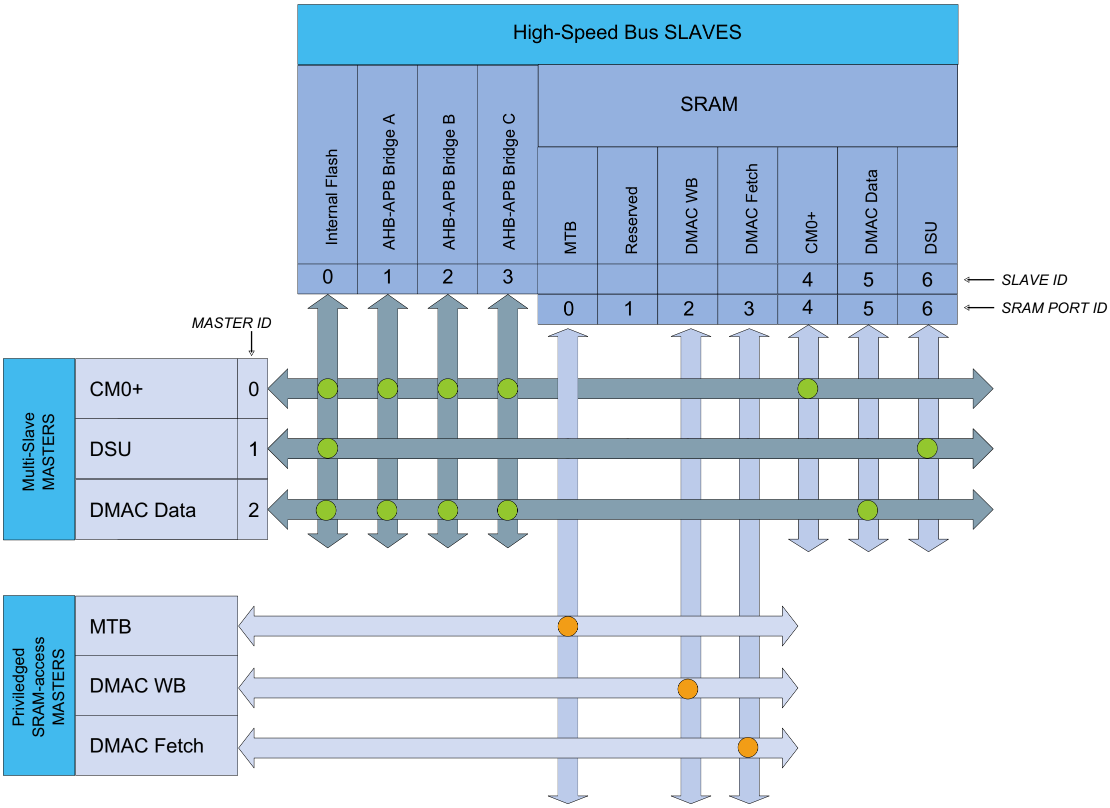


Table 10-4. Bus Matrix Masters

Bus Matrix Masters	Master ID
CM0+ - Cortex M0+ Processor	0
DSU - Device Service Unit	1
DMAC - Direct Memory Access Controller / Data Access	2

Table 10-5. Bus Matrix Slaves

Bus Matrix Slaves	Slave ID
Internal Flash Memory	0
AHB-APB Bridge A	1
AHB-APB Bridge B	2
AHB-APB Bridge C	3
SRAM Port 4 - CM0+ Access	4
SRAM Port 5 - DMAC Data Access	5
SRAM Port 6 - DSU Access	6

Table 10-6. SRAM Port Connection

SRAM Port Connection	Port ID	Connection Type
MTB - Memory Trace Buffer	0	Direct
Reserved	1	Direct
DMAC - Direct Memory Access Controller - Write-Back Access	2	Direct
DMAC - Direct Memory Access Controller - Fetch Access	3	Direct
CM0+ - Cortex M0+ Processor	4	Bus Matrix
DMAC - Direct Memory Access Controller - Data Access	5	Bus Matrix
DSU - Device Service Unit	6	Bus Matrix

10.3.3 SRAM Quality of Service

To ensure that masters with latency requirements get sufficient priority when accessing RAM, the different masters can be configured to have a given priority for different type of access.

The Quality of Service (QoS) level is independently selected for each master accessing the RAM. For any access to the RAM the RAM also receives the QoS level. The QoS levels and their corresponding bit values for the QoS level configuration is shown in Table 10-7.

Table 10-7. Quality of Service

Value	Name	Description
00	DISABLE	Background (no sensitive operations)
01	LOW	Sensitive Bandwidth
10	MEDIUM	Sensitive Latency
11	HIGH	Critical Latency

If a master is configured with QoS level 0x00 or 0x01 there will be minimum one cycle latency for the RAM access.

The priority order for concurrent accesses are decided by two factors. First the QoS level for the master and then a static priority given by table nn-mm (table: SRAM port connection) where the lowest port ID has the highest static priority.

The MTB has fixed QoS level 3 and the DSU has fixed QoS level 1.

The CPU QoS level can be written/read at address 0x41007110, bits [1:0]. Its reset value is 0x0.

Refer to different master QOSCTRL registers for configuring QoS for the other master (DMAC).

10.4 AHB-APB Bridge

The AHB-APB bridge is an AHB slave, providing an interface between the high-speed AHB domain and the low-power APB domain. It is used to provide access to the programmable control registers of peripherals (see "Product Mapping" on page 17).

AHB-APB bridge is based on AMBA APB Protocol Specification V2.0 (ref. as APB4) including:

- Wait state support
- Error reporting
- Transaction protection
- Sparse data transfer (byte, half-word and word)

Additional enhancements:

- Address and data cycles merged into a single cycle
- Sparse data transfer also apply to read access

to operate the AHB-APB bridge, the clock (CLK_HPBx_AHB) must be enabled. See "PM – Power Manager" on page 107 for details.

10.5 PAC – Peripheral Access Controller

10.5.1 Overview

There is one PAC associated with each AHB-APB bridge. The PAC can provide write protection for registers of each peripheral connected on the same bridge.

The PAC peripheral bus clock (CLK_PACx_APB) is enabled by default, and can be enabled and disabled in the Power Manager. Refer to "PM – Power Manager" on page 107 for details. CLK_PAC0_APB and CLK_PAC1_APB are enabled at reset, while CLK_PAC2_APB is disabled at reset. The PAC will continue to operate in any sleep mode where the selected clock source is running.

Write-protection does not apply for debugger access. When the debugger makes an access to a peripheral, write-protection is ignored so that the debugger can update the register.

Write-protect registers allow the user to disable a selected peripheral's write-protection without doing a read-modify-write operation. These registers are mapped into two I/O memory locations, one for clearing and one for setting the register bits. Writing a one to a bit in the Write Protect Clear register (WPCLR) will clear the corresponding bit in both registers (WPCLR and WPSET) and disable the write-protection for the corresponding peripheral, while writing a one to a bit in the Write Protect Set (WPSET) register will set the corresponding bit in both registers (WPCLR and WPSET) and enable the write-protection for the corresponding peripheral. Both registers (WPCLR and WPSET) will return the same value when read.

If a peripheral is write-protected, and if a write access is performed, data will not be written, and the peripheral will return an access error (CPU exception).

The PAC also offers a safety feature for correct program execution, with a CPU exception generated on double write-protection or double unprotection of a peripheral. If a peripheral *n* is write-protected and a write to one in WPSET[n] is detected, the PAC returns an error. This can be used to ensure that the application follows the intended program flow by always following a write-protect with an unprotect, and vice versa. However, in applications where a write-protected peripheral is used in several contexts, e.g., interrupts, care should be taken so that either the interrupt can not happen while the main application or other interrupt levels manipulate the write-protection status, or when the interrupt handler needs to unprotect the peripheral, based on the current protection status, by reading WPSET.

10.6 Register Description

Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, and the 8-bit halves of a 16-bit register can be accessed directly.

Refer to "Product Mapping" on page 17 for PAC locations.

10.6.1 PAC0 Register Description

Write Protect Clear

Name: WPCLR

Offset: 0x00

Reset: 0x00000000

-

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		EIC	RTC	WDT	GCLK	SYSCTRL	PM	
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:7 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 6:1 – EIC, RTC, WDT, GCLK, SYSCTRL, PM: Write Protect Disable**
  0: Write-protection is disabled.
  1: Write-protection is enabled.
  Writing a zero to these bits has no effect.
  Writing a one to these bits will clear the Write Protect bits for the corresponding peripherals.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

Write Protect Set

Name: WPSET

Offset: 0x04

Reset: 0x00000000

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		EIC	**RTC**	**WDT**	**GCLK**	**SYSCTRL**	**PM**	
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:7 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 6:1 – EIC, RTC, WDT, GCLK, SYSCTRL, PM: Write Protect Enable**
  0: Write-protection is disabled.
  1: Write-protection is enabled.
  Writing a zero to these bits has no effect.
  Writing a one to these bits will set the Write Protect bit for the corresponding peripherals.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

10.6.2 PAC1 Register Description

Write Protect Clear

Name: WPCLR

Offset: 0x00

Reset: 0x00000002

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		MTB		DMAC	PORT	NVMCTRL	DSU	
Access	R	R/W	R	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	1	0

- **Bits 31:7,5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 6,4:1 – MTB, DMAC, PORT, NVMCTRL, DSU: Write Protect**
  0: Write-protection is disabled.

  1: Write-protection is enabled.

  Writing a zero to these bits has no effect.

  Writing a one to these bits will clear the Write Protect bit for the corresponding peripherals.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

Write Protect Set

Name: WPSET

Offset: 0x04

Reset: 0x00000002

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		MTB		DMAC	PORT	NVMCTRL	DSU	
Access	R	R/W	R	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	1	0

- **Bits 31:7,5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 6,4:1 – MTB, DMAC, PORT, NVMCTRL, DSU: Write Protect**
  0: Write-protection is disabled.
  1: Write-protection is enabled.
  Writing a zero to these bits has no effect.
  Writing a one to these bits will set the Write Protect bit for the corresponding peripherals.
- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

10.6.3 PAC2 Register Description

Write Protect Clear

Name: WPCLR

Offset: 0x00

Reset: 0x00100000

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
								ADC
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						TC2	**TC1**	
Access	R	R	R	R	R	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					SERCOM1	**SERCOM0**	**EVSYS**	
Access	R	R	R	R	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:17, 15:11, 8:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to reset value when this register is written. These bits will always return reset value when read.

- **Bits 16, 10:9, 3:1 – ADC, TC2, TC1, SERCOM1, SERCOM0, EVSYS: Write Protect**
  0: Write-protection is disabled.

  1: Write-protection is enabled.

  Writing a zero to these bits has no effect.

  Writing a one to these bits will clear the Write Protect bit for the corresponding peripherals.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

Write Protect Set

Name: WPSET

Offset: 0x04

Reset: 0x00100000

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
								ADC
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						TC2	**TC1**	
Access	R	R	R	R	R	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					SERCOM1	**SERCOM0**	**EVSYS**	
Access	R	R	R	R	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:17, 15:11, 8:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to reset value when this register is written. These bits will always return reset value when read.

- **Bits 16, 10:9, 3:1 – ADC, TC2, TC1, SERCOM1, SERCOM0, EVSYS: Write Protect**
  0: Write-protection is disabled.

  1: Write-protection is enabled.

  Writing a zero to these bits has no effect.

  Writing a one to these bits will set the Write Protect bit for the corresponding peripherals.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

11. Peripherals Configuration Summary

Table 11-1. Peripherals Configuration Summary

Peripheral Name	Base Address	IRQ Line	AHB Clock		APB Clock		Generic Clock	PAC		Events		DMA	SleepWalking
			Index	Enabled at Reset	Index	Enabled at Reset	Index	Index	Prot at Reset	User	Generator	Index	
AHB-APB Bridge A	0x40000000		0	Y									
PAC0	0x40000000				0	Y							
PM	0x40000400	0			1	Y		1	N				Y
SYSCTRL	0x40000800	1			2	Y	0: DFLL48M reference 1: FDPLL96M clk source 2: FDPLL96M 32kHz	2	N				Y
GCLK	0x40000C00				3	Y		3	N				Y
WDT	0x40001000	2			4	Y	3	4	N				
RTC	0x40001400	3			5	Y	4	5	N		1: CMP0/ALARM0 2: CMP1 3: OVF 4-11: PER0-7		Y
EIC	0x40001800	NMI, 4			6	Y	5	6	N		12-27: EXTINT0-15		Y
AHB-APB Bridge B	0x41000000		1	Y									
PAC1	0x41000000				0	Y							
DSU	0x41002000		3	Y	1	Y		1	Y				
NVMCTRL	0x41004000	5	4	Y	2	Y	13	2	N				
PORT	0x41004400				3	Y		3	N				
DMAC	0x41004800	6	5	Y	4	Y		4	N	0-3: CH0-3	30-33: CH0-3		
MTB	0x41006000							6	N				
AHB-APB Bridge C	0x42000000		2	Y									
PAC2	0x42000000				0	N							
EVSYS	0x42000400	8			1	N	7-12: one per CHANNEL	1	N				Y
SERCOM0	0x42000800	9			2	N	14: CORE 13: SLOW	2	N			1: RX 2: TX	Y
SERCOM1	0x42000C00	10			3	N	15: CORE 13: SLOW	3	N			3: RX 4: TX	Y

Table 11-1. Peripherals Configuration Summary (Continued)

Peripheral Name	Base Address	IRQ Line	AHB Clock		APB Clock		Generic Clock	PAC		Events		DMA	SleepWalking
			Index	Enabled at Reset	Index	Enabled at Reset	Index	Index	Prot at Reset	User	Generator	Index	
TC1	0x42001800	13			6	N	18	6	N	18: EV	51: OVF 52-53: MC0-1	24: OVF 25-26: MC0-1	Y
TC2	0x42001C00	14			7	N	18	7	N	19: EV	54: OVF 55-56: MCX0-1	27: OVF 28-29: MC0-1	Y
ADC	0x42002000	15			8	Y	19	8	N	23: START 24: SYNC	66: RESRDY 67: WINMON	39: RESRDY	Y

12. DSU – Device Service Unit

12.1 Overview

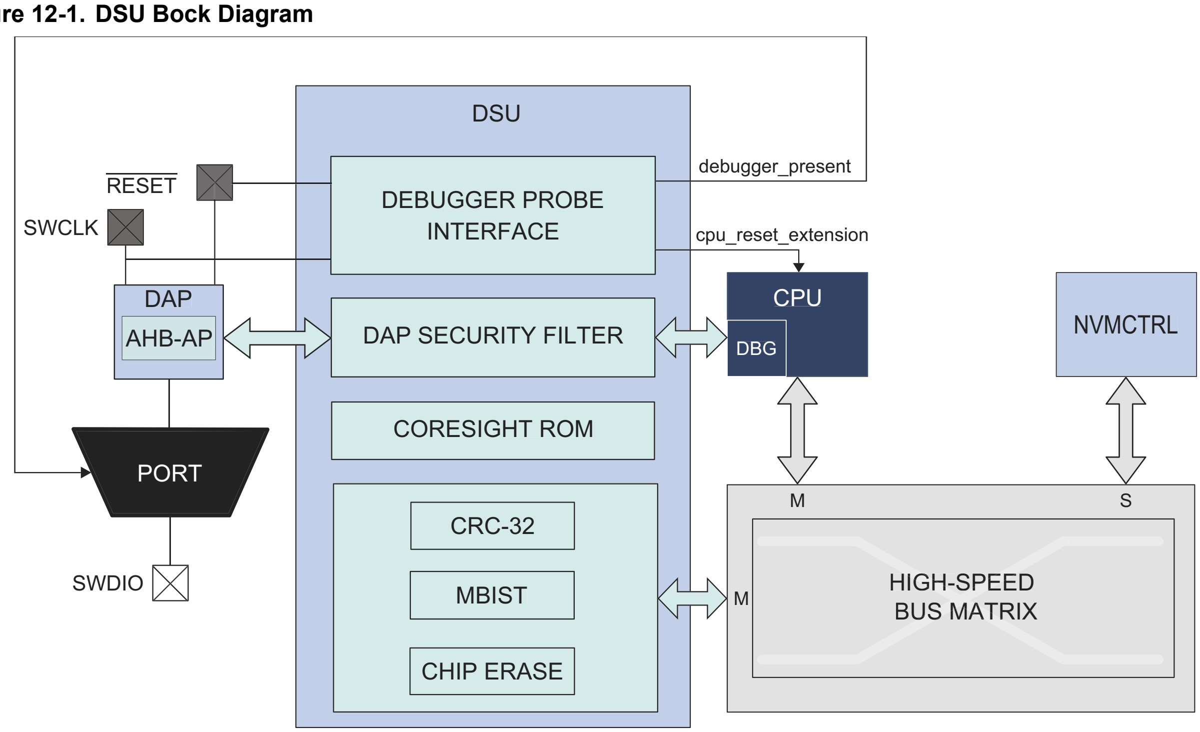
The Device Service Unit (DSU) provides a means to detect debugger probes. This enables the ARM Debug Access Port (DAP) to have control over multiplexed debug pads and CPU reset. The DSU also provides system-level services to debug adapters in an ARM debug system. It implements a CoreSight Debug ROM that provides device identification as well as identification of other debug components in the system. Hence, it complies with the ARM Peripheral Identification specification. The DSU also provides system services to applications that need memory testing, as required for IEC60730 Class B compliance, for example. The DSU can be accessed simultaneously by a debugger and the CPU, as it is connected on the High-Speed Bus Matrix. For security reasons, some of the DSU features will be limited or unavailable when the device is protected by the NVMCTRL security bit (refer to "Security Bit" on page 358).

12.2 Features

- CPU reset extension
- Debugger probe detection (Cold- and Hot-Plugging)
- Chip-Erase command and status
- 32-bit cyclic redundancy check (CRC32) of any memory accessible through the bus matrix
- ARM® CoreSight™ compliant device identification
- Two debug communications channels
- Debug access port security filter
- Onboard memory built-in self-test (MBIST)

12.3 Block Diagram

Figure 12-1. DSU Bock Diagram



12.4 Signal Description

Table 12-1. Signal Description

Signal Name	Type	Description
$\overline{\text{RESET}}$	Digital Input	External reset
SWCLK	Digital Input	SW clock
SWDIO	Digital I/O	SW bidirectional data pin

Refer to "I/O Multiplexing and Considerations" on page 10 for details on the pin mapping for this peripheral.

12.5 Product Dependencies

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

12.5.1 I/O Lines

The SWCLK pin is by default assigned to the DSU module to allow debugger probe detection and the condition to stretch the CPU reset phase. For more information, refer to "Debugger Probe Detection" on page 38. The Hot-Plugging feature depends on the PORT configuration. If the SWCLK pin function is changed in the PORT or if the PORT_MUX is disabled, the Hot-Plugging feature is disabled until a power-reset or an external reset.

12.5.2 Power Management

The DSU will continue to operate in any sleep mode where the selected source clock is running.

Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

12.5.3 Clocks

The DSU bus clocks (CLK_DSU_APB and CLK_DSU_AHB) can be enabled and disabled in the Power Manager. For more information on the CLK_DSU_APB and CLK_DSU_AHB clock masks, refer to "PM – Power Manager" on page 107.

12.5.4 Interrupts

Not applicable.

12.5.5 Events

Not applicable.

12.5.6 Register Access Protection

All registers with write access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Debug Communication Channel 0 register (DCC0)
- Debug Communication Channel 1 register (DCC1)

Write-protection is denoted by the Write-Protection property in the register description.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

12.5.7 Analog Connections

Not applicable.

12.6 Debug Operation

12.6.1 Principle of Operation

The DSU provides basic services to allow on-chip debug using the ARM Debug Access Port and the ARM processor debug resources:

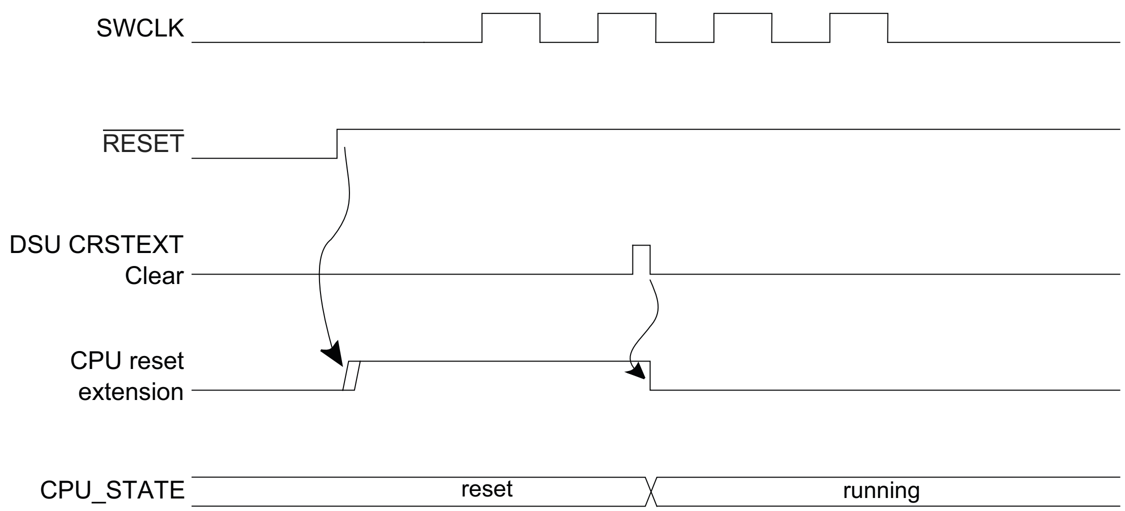
- CPU reset extension
- Debugger probe detection

For more details on the ARM debug components, refer to the ARM Debug Interface v5Architecture Specification.

12.6.2 CPU Reset Extension

"CPU reset extension" refers to the extension of the reset phase of the CPU core after the external reset is released. This ensures that the CPU is not executing code at startup while a debugger connects to the system. It is detected on a RESET release event when SWCLK is low. At startup, SWCLK is internally pulled up to avoid false detection of a debugger if SWCLK is left unconnected. When the CPU is held in the reset extension phase, the CPU Reset Extension bit (CRSTEXT) of the Status A register (STATUSA.CRSTEXT) is set. To release the CPU, write a one to STATUSA.CRSTEXT. STATUSA.CRSTEXT will then be set to zero. Writing a zero to STATUSA.CRSTEXT has no effect. For security reasons, it is not possible to release the CPU reset extension when the device is protected by the NVMCTRL security bit (refer to "Security Bit" on page 358). Trying to do so sets the Protection Error bit (PERR) of the Status A register (STATUSA.PERR).

Figure 12-2. Typical CPU Reset Extension Set and Clear Timing Diagram



12.6.3 Debugger Probe Detection

12.6.3.1 Cold-Plugging

Cold-Plugging is the detection of a debugger when the system is in reset. Cold-Plugging is detected when the CPU reset extension is requested, as described above.

12.6.3.2 Hot-Plugging

Hot-Plugging is the detection of a debugger probe when the system is not in reset. Hot-Plugging is not possible under reset because the detector is reset when POR or RESET are asserted. Hot-Plugging is active when a SWCLK falling edge is detected. The SWCLK pad is multiplexed with other functions and the user must ensure that its default function is assigned to the debug system. If the SWCLK function is changed, the Hot-Plugging feature is disabled until a power-reset or external reset occurs. Availability of the Hot-Plugging feature can be read from the Hot-Plugging Enable bit of the Status B register (STATUSB.HPE).

Figure 12-3. Hot-Plugging Detection Timing Diagram

The presence of a debugger probe is detected when either Hot-Plugging or Cold-Plugging is detected. Once detected, the Debugger Present bit of the Status B register (STATUSB.DBGPRES) is set. For security reasons, Hot-Plugging is not available when the device is protected by the NVMCTRL security bit (refer to "Security Bit" on page 358).

This detection requires that pads are correctly powered. Thus, at cold startup, this detection cannot be done until POR is released. If the device is protected, Cold-Plugging is the only way to detect a debugger probe, and so the external reset timing must be longer than the POR timing. If external reset is deasserted before POR release, the user must retry the procedure above until it gets connected to the device.

12.7 Chip-Erase

Chip-Erase consists of removing all sensitive information stored in the chip and clearing the NVMCTRL security bit (refer to "Security Bit" on page 358). Hence, all volatile memories and the flash array (including the EEPROM emulation area) will be erased. The flash auxiliary rows, including the user row, will not be erased. When the device is protected, the debugger must reset the device in order to be detected. This ensures that internal registers are reset after the protected state is removed. The Chip-Erase operation is triggered by writing a one to the Chip-Erase bit in the Control register (CTRL.CE). This command will be discarded if the DSU is protected by the Peripheral Access Controller (PAC). Once issued, the module clears volatile memories prior to erasing the flash array. To ensure that the Chip-Erase operation is completed, check the Done bit of the Status A register (STATUSA.DONE). The Chip-Erase operation depends on clocks and power management features that can be altered by the CPU. For that reason, it is recommended to issue a Chip-Erase after a Cold-Plugging procedure to ensure that the device is in a known and safe state.

The recommended sequence is as follows:

1. Issue the Cold-Plugging procedure (refer to "Cold-Plugging" on page 38). The device then:
   1. Detects the debugger probe
   2. Holds the CPU in reset
2. Issue the Chip-Erase command by writing a one to CTRL.CE. The device then:
   1. Clears the system volatile memories
   2. Erases the whole flash array (including the EEPROM emulation area, not including auxiliary rows)
   3. Erases the lock row, removing the NVMCTRL security bit protection
3. Check for completion by polling STATUSA.DONE (read as one when completed).
4. Reset the device to let the NVMCTRL update fuses.

12.8 Programming

Programming of the flash or RAM memories is available when the device is not protected by the NVMCTRL security bit (refer to "Security Bit" on page 358).

1. At power up, $\overline{\text{RESET}}$ is driven low by a debugger. The on-chip regulator holds the system in a POR state until the input supply is above the POR threshold (refer to "Power-On Reset (POR) Characteristics" on page 660).

The system continues to be held in this static state until the internally regulated supplies have reached a safe operating state.

2. The PM starts, clocks are switched to the slow clock (Core Clock, System Clock, Flash Clock and any Bus Clocks that do not have clock gate control). Internal resets are maintained due to the external reset.
3. The debugger maintains a low level on SWCLK. Releasing $\overline{\text{RESET}}$ results in a debugger Cold-Plugging procedure.
4. The debugger generates a clock signal on the SWCLK pin, the Debug Access Port (DAP) receives a clock.
5. The CPU remains in reset due to the Cold-Plugging procedure; meanwhile, the rest of the system is released.
6. A Chip-Erase is issued to ensure that the flash is fully erased prior to programming.
7. Programming is available through the AHB-AP.
8. After operation is completed, the chip can be restarted either by asserting $\overline{\text{RESET}}$, toggling power or writing a one to the Status A register CPU Reset Phase Extension bit (STATUSA.CRSTEXT). Make sure that the SWCLK pin is high when releasing $\overline{\text{RESET}}$ to prevent extending the CPU reset.

12.9 Intellectual Property Protection

Intellectual property protection consists of restricting access to internal memories from external tools when the device is protected, and is accomplished by setting the NVMCTRL security bit (refer to "Security Bit" on page 358). This protected state can be removed by issuing a Chip-Erase (refer to "Chip-Erase" on page 39). When the device is protected, read/write accesses using the AHB-AP are limited to the DSU address range and DSU commands are restricted.

The DSU implements a security filter that monitors the AHB transactions generated by the ARM AHB-AP inside the DAP. If the device is protected, then AHB-AP read/write accesses outside the DSU external address range are discarded, causing an error response that sets the ARM AHB-AP sticky error bits (refer to the ARM Debug Interface v5 Architecture Specification on http://www.arm.com).

The DSU is intended to be accessed either:

- Internally from the CPU, without any limitation, even when the device is protected
- Externally from a debug adapter, with some restrictions when the device is protected

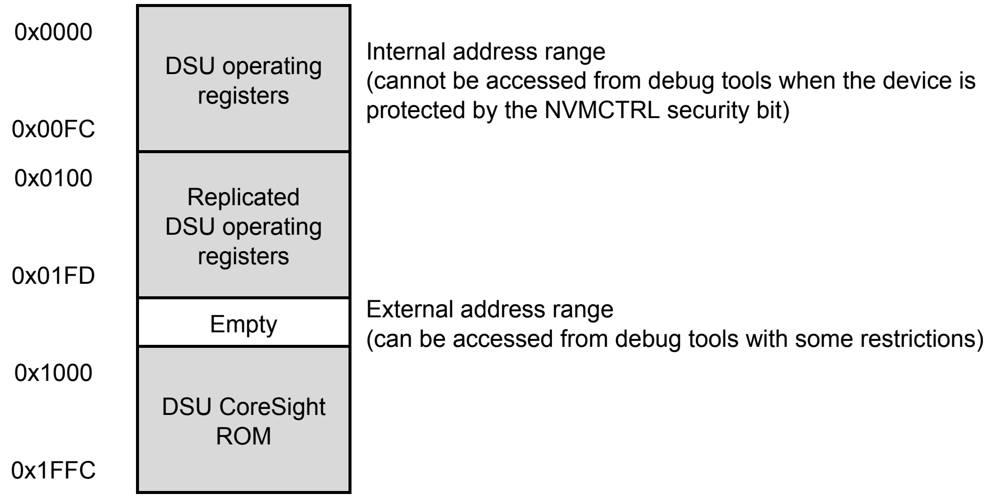
For security reasons, DSU features have limitations when used from a debug adapter. To differentiate external accesses from internal ones, the first 0x100 bytes of the DSU register map have been replicated at offset 0x100:

- The first 0x100 bytes form the internal address range
- The next 0x100 bytes form the external address range

When the device is protected, the DAP can only issue MEM-AP accesses in the DSU address range limited to the 0x100-0x2000 offset range.

The DSU operating registers are located in the 0x00-0xFF area and remapped in 0x100-0x1FF to differentiate accesses coming from a debugger and the CPU. If the device is protected and an access is issued in the region 0x100-0x1FF, it is subject to security restrictions. For more information, refer to Table 12-2.

Figure 12-4. APB Memory Mapping



Some features not activated by APB transactions are not available when the device is protected:

Table 12-2. Feature Availability Under Protection

Features	Availability When the Device is Protected
CPU reset extension	Yes
Debugger Cold-Plugging	Yes
Debugger Hot-Plugging	No

12.10 Device Identification

Device identification relies on the ARM CoreSight component identification scheme, which allows the chip to be identified as an ATMEL device implementing a DSU. The DSU contains identification registers to differentiate the device.

12.10.1 CoreSight Identification

A system-level ARM CoreSight ROM table is present in the device to identify the vendor and the chip identification method. Its address is provided in the MEM-AP BASE register inside the ARM Debug Access Port. The CoreSight ROM implements a 64-bit conceptual ID composed as follows from the PID0 to PID7 CoreSight ROM Table registers:

Figure 12-5. Conceptual 64-Bit Peripheral ID

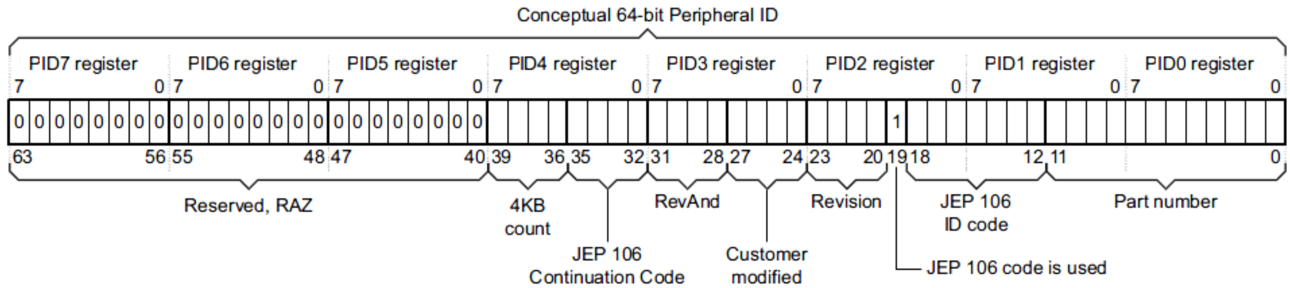


Table 12-3. Conceptual 64-Bit Peripheral ID Bit Descriptions

Field	Size	Description	Location
JEP-106 CC code	4	Atmel continuation code: 0x0	PID4
JEP-106 ID code	7	Atmel device ID: 0x1F	PID1+PID2
4KB count	4	Indicates that the CoreSight component is a ROM: 0x0	PID4
RevAnd	4	Not used; read as 0	PID3
CUSMOD	4	Not used; read as 0	PID3
PARTNUM	12	Contains 0xCD0 to indicate that DSU is present	PID0+PID1
REVISION	4	DSU revision (starts at 0x0 and increments by 1 at both major and minor revisions). Identifies DSU identification method variants. If 0x0, this indicates that device identification can be completed by reading the Device Identification register (DID)	PID3

For more information, refer to the ARM Debug Interface Version 5 Architecture Specification.

12.10.2 DSU Chip Identification Method:

The DSU DID register identifies the device by implementing the following information:

- Processor identification
- Family identification
- Subfamily identification
- Device select

12.11 Functional Description

12.11.1 Principle of Operation

The DSU provides memory services such as CRC32 or MBIST that require almost the same interface. Hence, the Address, Length and Data registers are shared. They must be configured first; then a command can be issued by writing the Control register. When a command is ongoing, other commands are discarded until the current operation is completed. Hence, the user must wait for the STATUSA.DONE bit to be set prior to issuing another one.

12.11.2 Basic Operation

12.11.2.1 Initialization

The module is enabled by enabling its clocks. For more details, refer to "Clocks" on page 37. The DSU registers can be write-protected. Refer to "PAC – Peripheral Access Controller" on page 27.

12.11.2.2 Operation from a debug adapter

Debug adapters should access the DSU registers in the external address range 0x100 – 0x2000. If the device is protected by the NVMCTRL security bit (refer to "Security Bit" on page 358), accessing the first 0x100 bytes causes the system to return an error (refer to "Intellectual Property Protection" on page 40).

12.11.2.3 Operation from the CPU

There are no restrictions when accessing DSU registers from the CPU. However, the user should access DSU registers in the internal address range (0x0 – 0x100) to avoid external security restrictions (refer to "Intellectual Property Protection" on page 40).

12.11.3 32-bit Cyclic Redundancy Check (CRC32)

The DSU unit provides support for calculating a cyclic redundancy check (CRC32) value for a memory area (including flash and AHB RAM).

When the CRC32 command is issued from:

- The internal range, the CRC32 can be operated at any memory location
- The external range, the CRC32 operation is restricted; DATA, ADDR and LENGTH values are forced (see below)

Table 12-4. AMOD Bit Descriptions when Operating CRC32

AMOD[1:0]	Short Name	External Range Restrictions
0	ARRAY	CRC32 is restricted to the full flash array area (EEPROM emulation area not included) DATA forced to 0xFFFFFFFF before calculation (no seed)
1	EEPROM	CRC32 of the whole EEPROM emulation area DATA forced to 0xFFFFFFFF before calculation (no seed)
2-3	Reserved	

The algorithm employed is the industry standard CRC32 algorithm using the generator polynomial 0xEDB88320 (reversed representation).

12.11.3.1 Starting CRC32 Calculation

CRC32 calculation for a memory range is started after writing the start address into the Address register (ADDR) and the size of the memory range into the Length register (LENGTH). Both must be word-aligned.

The initial value used for the CRC32 calculation must be written to the Data register. This value will usually be 0xFFFFFFFF, but can be, for example, the result of a previous CRC32 calculation if generating a common CRC32 of separate memory blocks.

Once completed, the calculated CRC32 value can be read out of the Data register. The read value must be complemented to match standard CRC32 implementations or kept non-inverted if used as starting point for subsequent CRC32 calculations.

If the device is in protected state by the NVMCTRL security bit (refer to "Security Bit" on page 358), it is only possible to calculate the CRC32 of the whole flash array when operated from the external address space. In most cases, this area will be the entire onboard non-volatile memory. The Address, Length and Data registers will be forced to predefined values once the CRC32 operation is started, and values written by the user are ignored. This allows the user to verify the contents of a protected device.

The actual test is started by writing a one in the 32-bit Cyclic Redundancy Check bit of the Control register (CTRL.CRC). A running CRC32 operation can be canceled by resetting the module (writing a one to CTRL.SWRST).

12.11.3.2 Interpreting the Results

The user should monitor the Status A register. When the operation is completed, STATUSA.DONE is set. Then the Bus Error bit of the Status A register (STATUSA.BERR) must be read to ensure that no bus error occurred.

12.11.4 Debug Communication Channels

The Debug Communication Channels (DCCO and DCC1) consist of a pair of registers with associated handshake logic, accessible by both CPU and debugger even if the device is protected by the NVMCTRL security bit (refer to "Security Bit" on page 358). The registers can be used to exchange data between the CPU and the debugger, during run time as well as in debug mode. This enables the user to build a custom debug protocol using only these registers. The DCC0 and DCC1 registers are accessible when the protected state is active. When the device is protected, however, it is not possible to connect a debugger while the CPU is running (STATUSA.CRSTEXT is not writable and the CPU is held under reset). Dirty bits in the status registers indicate whether a new value has been written in DCC0 or DCC1. These bits,DCC0D and DCC1D, are located in the STATUSB registers. They are automatically set on write and cleared on read. The DCC0 and DCC1 registers are shared with the onboard memory testing logic (MBIST). Accordingly, DCC0 and DCC1 must not be used while performing MBIST operations.

12.11.5 Testing of Onboard Memories (MBIST)

The DSU implements a feature for automatic testing of memory also known as MBIST. This is primarily intended for production test of onboard memories. MBIST cannot be operated from the external address range when the device is protected by the NVMCTRL security bit (refer to "Security Bit" on page 358). If a MBIST command is issued when the device is protected, a protection error is reported in the Protection Error bit in the Status A register (STATUSA.PERR).

1. Algorithm

   The algorithm used for testing is a type of March algorithm called "March LR". This algorithm is able to detect a wide range of memory defects, while still keeping a linear run time. The algorithm is:

   1. Write entire memory to 0, in any order.
   2. Bit for bit read 0, write 1, in descending order.
   3. Bit for bit read 1, write 0, read 0, write 1, in ascending order.
   4. Bit for bit read 1, write 0, in ascending order.
   5. Bit for bit read 0, write 1, read 1, write 0, in ascending order.
   6. Read 0 from entire memory, in ascending order.

   The specific implementation used has a run time that depends on CPU clock frequency and on the number of bytes tested in RAM. The detected faults are:

   - Address decoder faults
   - Stuck-at faults
   - Transition faults
   - Coupling faults
   - Linked Coupling faults
   - Stuck-open faults

2. Starting MBIST

   To test a memory, you need to write the start address of the memory to the ADDR.ADDR bit group, and the size of the memory into the Length register. See "Physical Memory Map" on page 18 to know which memories are available, and which address they are at.

   For best test coverage, an entire physical memory block should be tested at once. It is possible to test only a subset of a memory, but the test coverage will then be somewhat lower.

   The actual test is started by writing a one to CTRL.MBIST. A running MBIST operation can be canceled by writing a one to CTRL.SWRST.

3. Interpreting the Results

   The tester should monitor the STATUSA register. When the operation is completed, STATUSA.DONE is set. There are two different modes:

   - ADDR.AMOD=0: exit-on-error (default)

   In this mode, the algorithm terminates either when a fault is detected or on successful completion. In both cases, STATUSA.DONE is set. If an error was detected, STATUSA.FAIL will be set. User then can read the DATA and ADDR registers to locate the fault. Refer to "Locating Errors" on page 44.

   - ADDR.AMOD=1: pause-on-error

   In this mode, the MBIST algorithm is paused when an error is detected. In such a situation, only STATUSA.FAIL is asserted. The state machine waits for user to clear STATUSA.FAIL by writing a one in STATUSA.FAIL to resume. Prior to resuming, user can read the DATA and ADDR registers to locate the fault. Refer to "Locating Errors" on page 44.

4. Locating Errors

If the test stops with STATUSA.FAIL set, one or more bits failed the test. The test stops at the first detected error. The position of the failing bit can be found by reading the following registers:

- ADDR: Address of the word containing the failing bit.
- DATA: contains data to identify which bit failed, and during which phase of the test it failed. The DATA register will in this case contains the following bit groups:

Table 12-5. DATA bits Description When MBIST Operation Returns An Error

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

- bit_index: contains the bit number of the failing bit
- phase: indicates which phase of the test failed and the cause of the error. See Table 12-6 on page 45.

Table 12-6. MBIST Operation Phases

Phase	Test Actions
0	Write all bits to zero. This phase cannot fail.
1	Read 0, write 1, increment address
2	Read 1, write 0
3	Read 0, write 1, decrement address
4	Read 1, write 0, decrement address
5	Read 0, write 1
6	Read 1, write 0, decrement address
7	Read all zeros. bit_index is not used

12.11.6 System Services Availability When Accessed Externally

External access: Access performed in the DSU address offset 0x200-0x1FFF range.

Internal access: Access performed in the DSU address offset 0x0-0x100 range.

Table 12-7. Available Features When Operated From The External Address Range

Features	Availability From The External Address Range
Chip-Erase command and status	Yes
CRC32	Yes, only full array or full EEPROM
CoreSight Compliant Device identification	Yes
Debug communication channels	Yes
Testing of onboard memories (MBIST)	No
STATUSA.CRSTEXT clearing	No (STATUSA.PERR is set when attempting to do so)

12.12 Register Summary

Table 12-8. Register Summary

Offset	Name	Bit Pos.								
0x0000	CTRL	7:0	SMSA	ARR		CE	MBIST	CRC		SWRST
0x0001	STATUSA	7:0				PERR	FAIL	BERR	CRSTEXT	DONE
0x0002	STATUSB	7:0				HPE	DCCD1	DCCD0	DBGPRES	PROT
0x0003	Reserved									
0x0004	ADDR	7:0	ADDR[5:0]						AMOD[1:0]	
0x0005		15:8	ADDR[13:6]							
0x0006		23:16	ADDR[21:14]							
0x0007		31:24	ADDR[29:22]							
0x0008	LENGTH	7:0	LENGTH[5:0]							
0x0009		15:8	LENGTH[13:6]							
0x000A		23:16	LENGTH[21:14]							
0x000B		31:24	LENGTH[29:22]							
0x000C	DATA	7:0	DATA[7:0]							
0x000D		15:8	DATA[15:8]							
0x000E		23:16	DATA[23:16]							
0x000F		31:24	DATA[31:24]							
0x0010	DCC0	7:0	DATA[7:0]							
0x0011		15:8	DATA[15:8]							
0x0012		23:16	DATA[23:16]							
0x0013		31:24	DATA[31:24]							
0x0014	DCC1	7:0	DATA[7:0]							
0x0015		15:8	DATA[15:8]							
0x0016		23:16	DATA[23:16]							
0x0017		31:24	DATA[31:24]							
0x0018	DID	7:0	DEVSEL[7:0]							
0x0019		15:8	DIE[3:0]				REVISION[3:0]			
0x001A		23:16	FAMILY		SERIES[5:0]					
0x001B		31:24	PROCESSOR[3:0]				FAMILY[4:1]			
0x001C ... 0x00EF	Reserved									
0x00F0	DCFG0	7:0	DCFG[7:0]							
0x00F1		15:8	DCFG[15:8]							
0x00F2		23:16	DCFG[23:16]							
0x00F3		31:24	DCFG[31:24]							
0x00F4	DCFG1	7:0	DCFG[7:0]							
0x00F5		15:8	DCFG[15:8]							
0x00F6		23:16	DCFG[23:16]							
0x00F7		31:24	DCFG[31:24]							
0x00F8 ... 0x0FFF	Reserved									

Offset	Name	Bit Pos.								
0x1000	ENTRY0	7:0							FMT	EPRES
0x1001		15:8	ADDOFF[3:0]							
0x1002		23:16	ADDOFF[11:4]							
0x1003		31:24	ADDOFF[19:12]							
0x1004	ENTRY1	7:0							FMT	EPRES
0x1005		15:8	ADDOFF[3:0]							
0x1006		23:16	ADDOFF[11:4]							
0x1007		31:24	ADDOFF[19:12]							
0x1008	END	7:0	END[7:0]							
0x1009		15:8	END[15:8]							
0x100A		23:16	END[23:16]							
0x100B		31:24	END[31:24]							
0x100C ... 0x1FCB	Reserved									
0x1FCC	MEMTYPE	7:0								SMEMP
0x1FCD		15:8								
0x1FCE		23:16								
0x1FCF		31:24								
0x1FD0	PID4	7:0	FKBC[3:0]				JEPCC[3:0]			
0x1FD1		15:8								
0x1FD2		23:16								
0x1FD3		31:24								
0x1FD4	PID5	7:0								
0x1FD5		15:8								
0x1FD6		23:16								
0x1FD7		31:24								
0x1FD8	PID6	7:0								
0x1FD9		15:8								
0x1FDA		23:16								
0x1FDB		31:24								
0x1FDC	PID7	7:0								
0x1FDD		15:8								
0x1FDE		23:16								
0x1FDF		31:24								
0x1FE0	PID0	7:0	PARTNBL[7:0]							
0x1FE1		15:8								
0x1FE2		23:16								
0x1FE3		31:24								
0x1FE4	PID1	7:0	JEPIDCL[3:0]				PARTNBH[3:0]			
0x1FE5		15:8								
0x1FE6		23:16								
0x1FE7		31:24								

Offset	Name	Bit Pos.								
0x1FE8	PID2	7:0	REVISION[3:0]				JEPU	JEPIDCH[2:0]		
0x1FE9		15:8								
0x1FEA		23:16								
0x1FEB		31:24								
0x1FEC	PID3	7:0	REVAND[3:0]				CUSMOD[3:0]			
0x1FED		15:8								
0x1FEE		23:16								
0x1FEF		31:24								
0x1FF0	CID0	7:0	PREAMBLEB0[7:0]							
0x1FF1		15:8								
0x1FF2		23:16								
0x1FF3		31:24								
0x1FF4	CID1	7:0	CCLASS[3:0]				PREAMBLE[3:0]			
0x1FF5		15:8								
0x1FF6		23:16								
0x1FF7		31:24								
0x1FF8	CID2	7:0	PREAMBLEB2[7:0]							
0x1FF9		15:8								
0x1FFA		23:16								
0x1FFB		31:24								
0x1FFC	CID3	7:0	PREAMBLEB3[7:0]							
0x1FFD		15:8								
0x1FFE		23:16								
0x1FFF		31:24								

12.13 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write protection is denoted by the Write-Protected property in each individual register description. Refer to "Register Access Protection" on page 37 for details.

12.13.1 Control

Name: CTRL

Offset: 0x0000

Reset: 0x00

Access: Write-Only

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	SMSA	ARR		CE	MBIST	CRC		SWRST
Access	W	W	R	W	W	W	R	W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – SMSA: Start Memory Stream Access**

- **Bit 6 – ARR: Auxiliary Row Read**

- **Bit 5 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 4 – CE: Chip-Erase**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts the Chip-Erase operation.

- **Bit 3 – MBIST: Memory built-in self-test**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts the memory BIST algorithm.

- **Bit 2 – CRC: 32-bit Cyclic Redundancy Code**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts the cyclic redundancy check algorithm.

- **Bit 1 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 0 – SWRST: Software Reset**
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets the module.

12.13.2 Status A

Name: STATUSA

Offset: 0x0001

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
				PERR	FAIL	BERR	CRSTEXT	DONE
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 4 – PERR: Protection Error**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Protection Error bit.
  This bit is set when a command that is not allowed in protected state is issued.

- **Bit 3 – FAIL: Failure**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Failure bit.
  This bit is set when a DSU operation failure is detected.

- **Bit 2 – BERR: Bus Error**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Bus Error bit.
  This bit is set when a bus error is detected.

- **Bit 1 – CRSTEXT: CPU Reset Phase Extension**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the CPU Reset Phase Extension bit.
  This bit is set when a debug adapter Cold-Plugging is detected, which extends the CPU reset phase.

- **Bit 0 – DONE: Done**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Done bit.
  This bit is set when a DSU operation is completed.

12.13.3 Status B

Name: STATUSB

Offset: 0x0002

Reset: 0X000000XX

Access: Read-Only

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
				HPE	DCCD1	DCCD0	DBGPRES	PROT
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	X	X

- **Bits 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 4 – HPE: Hot-Plugging Enable**
  Writing a zero to this bit has no effect.

  Writing a one to this bit has no effect.

  This bit is set when Hot-Plugging is enabled.

  This bit is cleared when Hot-Plugging is disabled. This is the case when the SWCLK function is changed. Only a power-reset or a external reset can set it again.

- **Bits 3:2 – DCCDx [x=1..0]: Debug Communication Channel x Dirty**
  Writing a zero to this bit has no effect.

  Writing a one to this bit has no effect.

  This bit is set when DCCx is written.

  This bit is cleared when DCCx is read.

- **Bit 1 – DBGPRES: Debugger Present**
  Writing a zero to this bit has no effect.

  Writing a one to this bit has no effect.

  This bit is set when a debugger probe is detected.

  This bit is never cleared.

- **Bit 0 – PROT: Protected**
  Writing a zero to this bit has no effect.

  Writing a one to this bit has no effect.

  This bit is set at powerup when the device is protected.

  This bit is never cleared.

12.13.4 Address

Name: ADDR

Offset: 0x0004

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	ADDR[29:22]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ADDR[21:14]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[13:6]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[5:0]						AMOD[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:2 – ADDR[29:0]: Address**
  Initial word start address needed for memory operations.
- **Bits 1:0 – AMOD[1:0]: Access Mode**

12.13.5 Length

Name: LENGTH

Offset: 0x0008

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	LENGTH[29:22]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LENGTH[21:14]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LENGTH[13:6]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LENGTH[5:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:2 – LENGTH[29:0]: Length**
  Length in words needed for memory operations.

- **Bits 1:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

12.13.6 Data

Name: DATA
Offset: 0x000C
Reset: 0x00000000
Access: Read-Write
Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – DATA[31:0]: Data**
  Memory operation initial value or result value.

12.13.7 Debug Communication Channel n

Name: DCCn

Offset: 0x0010+n*0x4 [n=0..1]

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – DATA[31:0]: Data**
  Data register.

12.13.8 Device Identification

Name: DID

Offset: 0x0018

Reset: -

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
	PROCESSOR[3:0]				FAMILY[4:1]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FAMILY		SERIES[5:0]					
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIE[3:0]				REVISION[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DEVSEL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

The information in this register is related to the ordering code. Refer to the "Ordering Information" on page 4 for details.

- **Bits 31:28 – PROCESSOR[3:0]: Processor**
  The value of this field defines the processor used on the device. For this device, the value of this field is 0x1, corresponding to the ARM Cortex-M0+ processor.

Table 12-9. Processor

PROCESSOR[3:0]	Description
0x0	Cortex-M0
0x1	Cortex-M0+
0x2	Cortex-M3
0x3	Cortex-M4
0x4-0xF	Reserved

- **Bits 27:23 – FAMILY[4:0]: Family**
  The value of this field corresponds to the Product Family part of the ordering code. For this device, the value of this field is 0x0, corresponding to the SAM D family of base line microcontrollers.

Table 12-10. Family

FAMILY[4:0]	Description
0x0	General purpose microcontroller
0x1	PicoPower
0x2-0x1F	Reserved

- **Bit 22 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bits 21:16 – SERIES[5:0]: Series**
  The value of this field corresponds to the Product Series part of the ordering code. For this device, the value of this field is 0x00, corresponding to a product with the Cortex-M0+ processor and a basic feature set.

Table 12-11. Series

SERIES[5:0]	Description
0x0	Cortex-M0+ processor, basic feature set
0x1-0x3F	Reserved

- **Bits 15:12 – DIE[3:0]: Die Number**
  Identifies the die in the family.
- **Bits 11:8 – REVISION[3:0]: Revision Number**
  Identifies the die revision number.
- **Bits 7:0 – DEVSEL[7:0]: Device Select**
  DEVSEL is used to identify a device within a product family and product series. The value corresponds to the Flash memory density, pin count and device variant parts of the ordering code. Refer to "Ordering Information" on page 4 for details.
  DEVSEL is used to identify a device within a product family and product series. The value corresponds to the Flash memory density, pin count and device variant parts of the ordering code. Refer to "Ordering Information" on page 4 for details.

Table 12-12. Device Selection

DEVSEL	Device	Device ID	Flash	RAM	Pincount	DAC	SERCOM2	USB
0x2	Reserved							
0x5	Reserved							
0x8-0xFF	Reserved							

12.13.9 Device Configuration

Name: DCFGn

Offset: 0x00F0+n*0x4 [n=0..1]

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
	DCFG[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCFG[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DCFG[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCFG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – DCFG[31:0]: Device Configuration**

12.13.10 Coresight ROM Table Entry n

Name: ENTRYn

Offset: 0x1000+n*0x4 [n=0..1]

Reset: -

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
	ADDOFF[19:12]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ADDOFF[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDOFF[3:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
							FMT	EPRES
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:12 – ADDOFF[19:0]: Address Offset**
  The base address of the component, relative to the base address of this ROM table.

- **Bits 11:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – FMT: Format**
  Always read as one, indicates a 32-bit ROM table.

- **Bit 0 – EPRES: Entry Present**
  This bit indicates whether an entry is present at this location in the ROM table.
  This bit is set at powerup if the device is not protected indicating that the entry is not present.
  This bit is cleared at powerup if the device is not protected indicating that the entry is present.

12.13.11 Coresight ROM Table End

Name: END

Offset: 0x1008

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
	END[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	END[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	END[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	END[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – END[31:0]: End Marker**
  Indicates the end of the CoreSight ROM table entries.

12.13.12 Coresight ROM Table Memory Type

Name: MEMTYPE

Offset: 0x1FCC

Reset: 0X0000000000000000000000000000000X

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
								SMEMP
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	X

- **Bits 31:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – SMEMP: System Memory Present**
  This bit indicates whether system memory is present on the bus that connects to the ROM table.
  This bit is set at powerup if the device is not protected indicating that the system memory is accessible from a debug adapter.
  This bit is cleared at powerup if the device is protected indicating that the system memory is not accessible from a debug adapter.

12.13.13 Peripheral Identification 4

Name: PID4

Offset: 0x1FD0

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FKBC[3:0]				JEPCC[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:4 – FKBC[3:0]: 4KB count**
  These bits will always return zero when read, indicating that this debug component occupies one 4KB block.

- **Bits 3:0 – JEPCC[3:0]: JEP-106 Continuation Code**
  These bits will always return zero when read, indicating a Atmel device.

12.13.14 Peripheral Identification 5

Name: PID5

Offset: 0x1FD4

Reset: -

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

12.13.15 Peripheral Identification 6

Name: PID6

Offset: 0x1FD8

Reset: -

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

12.13.16 Peripheral Identification 7

Name: PID7

Offset: 0x1FDC

Reset: -

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

12.13.17 Peripheral Identification 0

Name: PID0

Offset: 0x1FE0

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PARTNBL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – PARTNBL[7:0]: Part Number Low**
  These bits will always return 0xD0 when read, indicating that this device implements a DSU module instance.

12.13.18 Peripheral Identification 1

Name: PID1

Offset: 0x1FE4

Reset: 0x000000FC

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	JEPIDCL[3:0]				PARTNBH[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:4 – JEPIDCL[3:0]: Low part of the JEP-106 Identity Code**
  These bits will always return 0xF when read, indicating a Atmel device (Atmel JEP-106 identity code is 0x1F).

- **Bits 3:0 – PARTNBH[3:0]: Part Number High**
  These bits will always return 0xC when read, indicating that this device implements a DSU module instance.

12.13.19 Peripheral Identification 2

Name: PID2

Offset: 0x1FE8

Reset: 0x00000009

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	REVISION[3:0]				JEPU	JEPIDCH[2:0]		
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	1

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:4 – REVISION[3:0]: Revision Number**
  Revision of the peripheral. Starts at 0x0 and increments by one at both major and minor revisions.

- **Bit 3 – JEPU: JEP-106 Identity Code is used**
  This bit will always return one when read, indicating that JEP-106 code is used.

- **Bits 2:0 – JEPIDCH[2:0]: JEP-106 Identity Code High**
  These bits will always return 0x1 when read, indicating an Atmel device (Atmel JEP-106 identity code is 0x1F).

12.13.20 Peripheral Identification 3

Name: PID3
Offset: 0x1FEC
Reset: 0x00000000
Access: Read-Only
Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	REVAND[3:0]				CUSMOD[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:4 – REVAND[3:0]: Revision Number**
  These bits will always return 0x0 when read.

- **Bits 3:0 – CUSMOD[3:0]: ARM CUSMOD**
  These bits will always return 0x0 when read.

12.13.21Component Identification 0

Name: CID0

Offset: 0x1FF0

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PREAMBLEB0[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – PREAMBLEB0[7:0]: Preamble Byte 0**
  These bits will always return 0xD when read.

12.13.22 Component Identification 1

Name: CID1

Offset: 0x1FF4

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CCLASS[3:0]				PREAMBLE[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:4 – CCLASS[3:0]: Component Class**
  These bits will always return 0x1 when read indicating that this ARM CoreSight component is ROM table (refer to the ARM Debug Interface v5 Architecture Specification at http://www.arm.com).

- **Bits 3:0 – PREAMBLE[3:0]: Preamble**
  These bits will always return 0x0 when read.

12.13.23 Component Identification 2

Name: CID2

Offset: 0x1FF8

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PREAMBLEB2[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – PREAMBLEB2[7:0]: Preamble Byte 2**
  These bits will always return 0x05 when read.

12.13.24 Component Identification 3

Name: CID3

Offset: 0x1FFC

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PREAMBLEB3[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

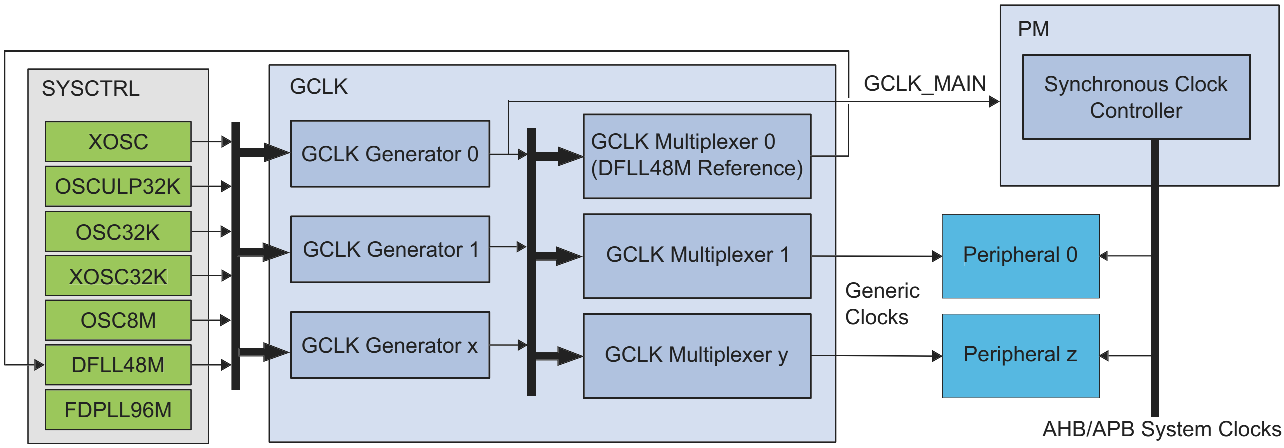
- **Bits 7:0 – PREAMBLEB3[7:0]: Preamble Byte 3**
  These bits will always return 0xB1 when read.

13. Clock System

This chapter only aims to summarize the clock distribution and terminology in the SAM D09 device. It will not explain every detail of its configuration. For in-depth documentation, see the referenced module chapters.

13.1 Clock Distribution

Figure 13-1. Clock distribution

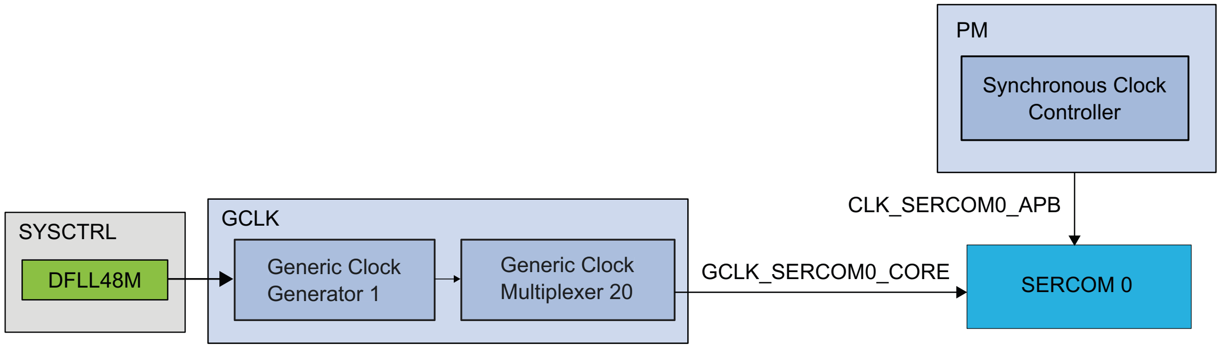


The clock system on the SAM D09 consists of:

- ***Clock sources***, controlled by SYSCTRL
  - A Clock source is the base clock signal used in the system. Example clock sources are the internal 8MHz oscillator (OSC8M), External crystal oscillator (XOSC) and the Digital frequency locked loop (DFLL48M).
- ***Generic Clock Controller (GCLK)*** which controls the clock distribution system, made up of:
  - ***Generic Clock generators****:* A programmable prescaler, that can use any of the system clock sources as its source clock. The Generic Clock Generator 0, also called GCLK_MAIN, is the clock feeding the Power Manager used to generate synchronous clocks.
  - ***Generic Clocks***: Typically the clock input of a peripheral on the system. The generic clocks, through the Generic Clock Multiplexer, can use any of the Generic Clock generators as its clock source. Multiple instances of a peripheral will typically have a separate generic clock for each instance. The DFLL48M clock input (when multiplying another clock source) is generic clock 0.
- **Power Manager (PM)**
  - The PM controls synchronous clocks on the system. This includes the CPU, bus clocks (APB, AHB) as well as the synchronous (to the CPU) user interfaces of the peripherals. It contains clock masks that can turn on/off the user interface of a peripheral as well as prescalers for the CPU and bus clocks.

Figure 13-2 shows an example where SERCOM0 is clocked by the DFLL48M in open loop mode. The DFLL48M is enabled, the Generic Clock Generator 1 uses the DFLL48M as its clock source, and the generic clock 20, also called GCLK_SERCOM0_CORE, that is connected to SERCOM0 uses generator 1 as its source. The SERCOM0 interface, clocked by CLK_SERCOM0_APB, has been unmasked in the APBC Mask register in the PM.

Figure 13-2. Example of SERCOM clock



13.2 Synchronous and Asynchronous Clocks

As the CPU and the peripherals can be clocked from different clock sources, possibly with widely different clock speeds, some peripheral accesses by the CPU needs to be synchronized between the different clock domains. In these cases the peripheral includes a SYNCBUSY status flag that can be used to check if a sync operation is in progress. As the nature of the synchronization might vary between different peripherals, detailed description for each peripheral can be found in the sub-chapter "synchronization" for each peripheral where this is necessary.

In the datasheet references to synchronous clocks are referring to the CPU and bus clocks, while asynchronous clocks are clock generated by generic clocks.

13.3 Register Synchronization

There are two different register synchronization schemes implemented on this device: some modules use a common synchronizer register synchronization scheme, while other modules use a distributed synchronizer register synchronization scheme.

The modules using a common synchronizer register synchronization scheme are: GCLK, WDT, RTC, EIC, TC, ADC.

The modules using a distributed synchronizer register synchronization scheme are: SERCOM USART, SERCOM SPI, SERCOM I2C, I2S.

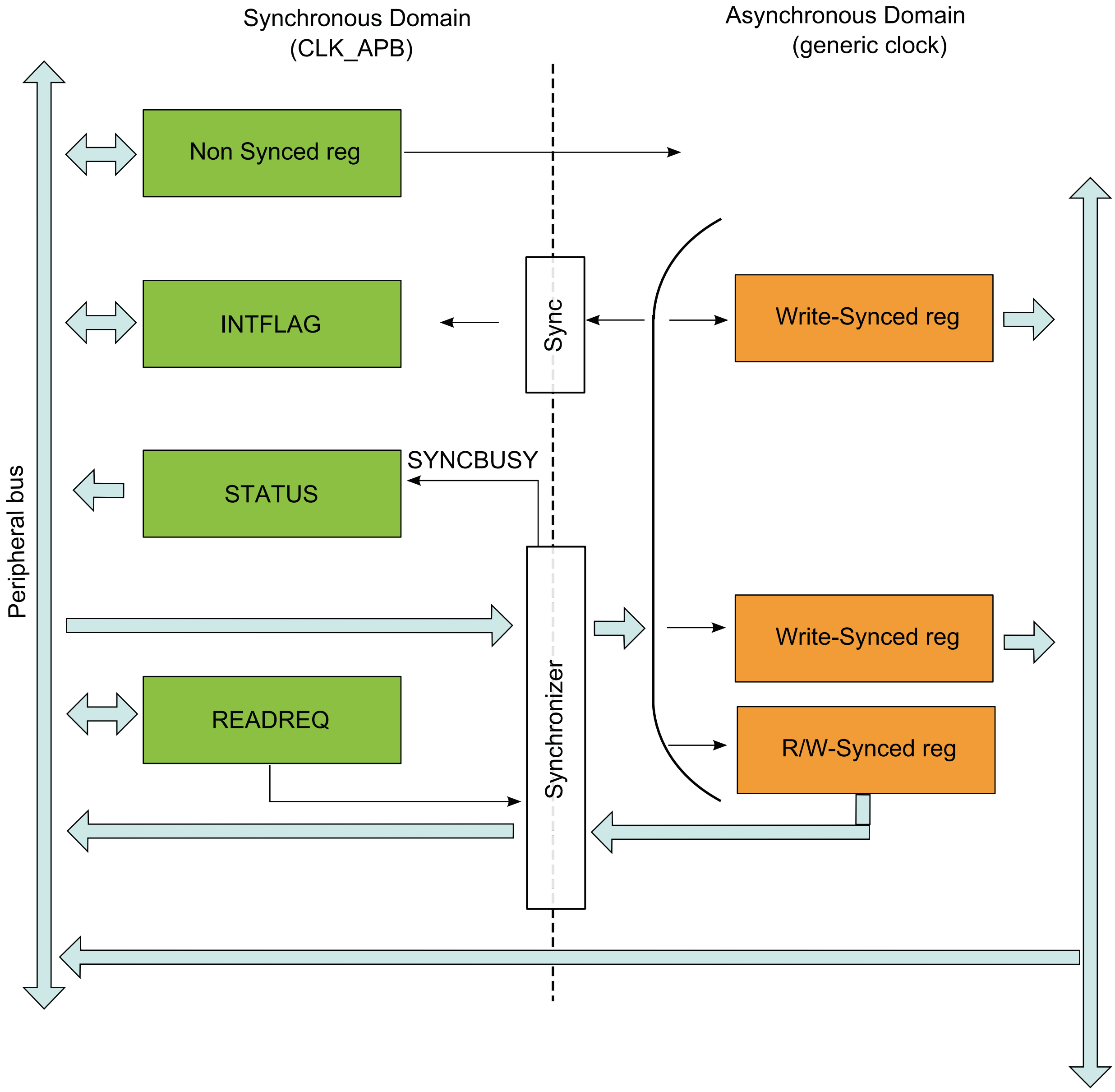
13.3.1 Common Synchronizer Register Synchronization

13.3.1.1 Overview

All peripherals are composed of one digital bus interface, which is connected to the APB or AHB bus and clocked using a corresponding synchronous clock, and one core clock, which is clocked using a generic clock. Access between these clock domains must be synchronized. As this mechanism is implemented in hardware the synchronization process takes place even if the different clocks domains are clocked from the same source and on the same frequency. All registers in the bus interface are accessible without synchronization. All core registers in the generic clock domain must be synchronized when written. Some core registers must be synchronized when read. Registers that need synchronization has this denoted in each individual register description. Two properties are used: write-synchronization and read-synchronization.

A common synchronizer is used for all registers in one peripheral, as shown in Figure 13-3. Therefore, only one register per peripheral can be synchronized at a time.

Figure 13-3. Synchronization



13.3.1.2 Write-Synchronization

The write-synchronization is triggered by a write to any generic clock core register. The Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set when the write-synchronization starts and cleared when the write-synchronization is complete. Refer to "Synchronization Delay" on page 81 for details on the synchronization delay.

When the write-synchronization is ongoing (STATUS.SYNCBUSY is one), any of the following actions will cause the peripheral bus to stall until the synchronization is complete:

- Writing a generic clock core register
- Reading a read-synchronized core register
- Reading the register that is being written (and thus triggered the synchronization)

Core registers without read-synchronization will remain static once they have been written and synchronized, and can be read while the synchronization is ongoing without causing the peripheral bus to stall. APB registers can also be read while the synchronization is ongoing without causing the peripheral bus to stall.

13.3.1.3 Read-Synchronization

Reading a read-synchronized core register will cause the peripheral bus to stall immediately until the read-synchronization is complete. STATUS.SYNCBUSY will not be set. Refer to "Synchronization Delay" on page 81 for details on the synchronization delay. Note that reading a read-synchronized core register while STATUS.SYNCBUSY is one will cause the peripheral bus to stall twice; first because of the ongoing synchronization, and then again because reading a read-synchronized core register will cause the peripheral bus to stall immediately.

13.3.1.4 Completion of synchronization

The user can either poll STATUS.SYNCBUSY or use the Synchronisation Ready interrupt (if available) to check when the synchronization is complete. It is also possible to perform the next read/write operation and wait, as this next operation will be started once the previous write/read operation is synchronized and/or complete.

13.3.1.5 Read Request

The read request functionality is only available to peripherals that have the Read Request register (READREQ) implemented. Refer to the register description of individual peripheral chapters for details.

To avoid forcing the peripheral bus to stall when reading read-synchronized core registers, the read request mechanism can be used.

Basic Read Request

Writing a one to the Read Request bit in the Read Request register (READREQ.RREQ) will request read-synchronization of the register specified in the Address bits in READREQ (READREQ.ADDR) and set STATUS.SYNCBUSY. When read-synchronization is complete, STATUS.SYNCBUSY is cleared. The read-synchronized value is then available for reading without delay until READREQ.RREQ is written to one again.

The address to use is the offset to the peripheral's base address of the register that should be synchronized.

Continuous Read Request

Writing a one to the Read Continuously bit in READREQ (READREQ.RCONT) will force continuous read-synchronization of the register specified in READREQ.ADDR. The latest value is always available for reading without stalling the bus, as the synchronization mechanism is continuously synchronizing the given value.

SYNCBUSY is set for the first synchronization, but not for the subsequent synchronizations. If another synchronization is attempted, i.e. by executing a write-operation of a write-synchronized register, the read request will be stopped, and will have to be manually restarted.

Note that continuous read-synchronization is paused in sleep modes where the generic clock is not running. This means that a new read request is required if the value is needed immediately after exiting sleep.

13.3.1.6 Enable Write-Synchronization

Writing to the Enable bit in the Control register (CTRL.ENABLE) will also trigger write-synchronization and set STATUS.SYNCBUSY. CTRL.ENABLE will read its new value immediately after being written. The Synchronisation Ready interrupt (if available) cannot be used for Enable write-synchronization.

When the enable write-synchronization is ongoing (STATUS.SYNCBUSY is one), attempt to do any of the following will cause the peripheral bus to stall until the enable synchronization is complete:

- Writing a core register
- Writing an APB register
- Reading a read-synchronized core register

APB registers can be read while the enable write-synchronization is ongoing without causing the peripheral bus to stall.

13.3.1.7 Software Reset Write-Synchronization

Writing a one to the Software Reset bit in CTRL (CTRL.SWRST) will also trigger write-synchronization and set STATUS.SYNCBUSY. When writing a one to the CTRL.SWRST bit it will immediately read as one. CTRL.SWRST and STATUS.SYNCBUSY will be cleared by hardware when the peripheral has been reset. Writing a zero to the

CTRL.SWRST bit has no effect. The Synchronisation Ready interrupt (if available) cannot be used for Software Reset write-synchronization.

When the software reset is in progress (STATUS.SYNCBUSY and CTRL.SWRST are one), attempt to do any of the following will cause the peripheral bus to stall until the Software Reset synchronization and the reset is complete:

- Writing a core register
- Writing an APB register
- Reading a read-synchronized register

APB registers can be read while the software reset is being write-synchronized without causing the peripheral bus to stall.

13.3.1.8 Synchronization Delay

The synchronization will delay the write or read access duration by a delay D, given by the equation:

$$5 \cdot P_{GCLK} + 2 \cdot P_{APB} < D < 6 \cdot P_{GCLK} + 3 \cdot P_{APB}$$

Where P_{GCLK} is the period of the generic clock and P_{APB} is the period of the peripheral bus clock. A normal peripheral bus register access duration is $2 \cdot P_{APB}$.

13.3.2 Distributed Synchronizer Register Synchronization

13.3.2.1 Overview

All peripherals are composed of one digital bus interface, which is connected to the APB or AHB bus and clocked using a corresponding synchronous clock, and one core clock, which is clocked using a generic clock. Access between these clock domains must be synchronized. As this mechanism is implemented in hardware the synchronization process takes place even if the different clocks domains are clocked from the same source and on the same frequency. All registers in the bus interface are accessible without synchronization. All core registers in the generic clock domain must be synchronized when written. Some core registers must be synchronized when read. Registers that need synchronization has this denoted in each individual register description.

13.3.2.2 General Write synchronization

Inside the same module, each core register, denoted by the Write-Synchronized property, use its own synchronization mechanism so that writing to different core registers can be done without waiting for the end of synchronization of previous core register access.

To write again to the same core register in the same module, user must wait for the end of synchronization or the write will be discarded.

For each core register, that can be written, a synchronization status bit is associated

Example:

REGA, REGB are 8-bit core registers. REGC is 16-bit core register.

Offset	Register
0x00	REGA
0x01	REGB
0x02	REGC
0x03	

Since synchronization is per register, user can write REGA (8-bit access) then immediately write REGB (8-bit access) without error.

User can write REGC (16-bit access) without affecting REGA or REGB. But if user writes REGC in two consecutive 8-bit accesses, second write will be discarded and generate an error.

When user makes a 32-bit access to offset 0x00, all registers are written but REGA, REGB, REGC can be updated at a different time because of independent write synchronization

13.3.2.3 General read synchronization

Before any read of a core register, the user must check that the related bit in SYNCBUSY register is cleared.

Read access to core register is always immediate but the return value is reliable only if a synchonization of this core register is not going.

13.3.2.4 Completion of synchronization

The user can either poll SYNCBUSY register or use the Synchronisation Ready interrupt (if available) to check when the synchronization is complete.

13.3.2.5 Enable Write-Synchronization

Writing to the Enable bit in the Control register (CTRL.ENABLE) will also trigger write-synchronization and set SYNCBUSY.ENABLE. CTRL.ENABLE will read its new value immediately after being written. The Synchronisation Ready interrupt (if available) cannot be used for Enable write-synchronization.

13.3.2.6 Software Reset Write-Synchronization

Writing a one to the Software Reset bit in CTRL (CTRL.SWRST) will also trigger write-synchronization and set SYNCBUSY.SWRST. When writing a one to the CTRL.SWRST bit it will immediately read as one. CTRL.SWRST and SYNCBUSY.SWRST will be cleared by hardware when the peripheral has been reset. Writing a zero to the CTRL.SWRST bit has no effect. The Synchronisation Ready interrupt (if available) cannot be used for Software Reset write-synchronization.

13.3.2.7 Synchronization Delay

The synchronization will delay the write or read access duration by a delay D, given by the equation:

$$5 \cdot P_{GCLK} + 2 \cdot P_{APB} < D < 6 \cdot P_{GCLK} + 3 \cdot P_{APB}$$

Where P_{GCLK} is the period of the generic clock and P_{APB} is the period of the peripheral bus clock. A normal peripheral bus register access duration is $2 \cdot P_{APB}$.

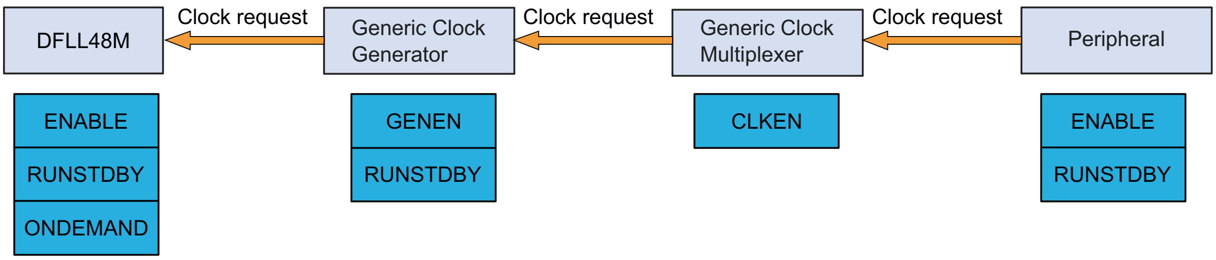
13.4 Enabling a Peripheral

To enable a peripheral clocked by a generic clock, the following parts of the system needs to be configured:

- A running clock source.
- A clock from the Generic Clock Generator must be configured to use one of the running clock sources, and the generator must be enabled.
- The generic clock, through the Generic Clock Multiplexer, that connects to the peripheral needs to be configured with a running clock from the Generic Clock Generator, and the generic clock must be enabled.
- The user interface of the peripheral needs to be unmasked in the PM. If this is not done the peripheral registers will read as all 0’s and any writes to the peripheral will be discarded.

13.5 On-demand, Clock Requests

Figure 13-4. Clock request routing



All the clock sources in the system can be run in an on-demand mode, where the clock source is in a stopped state when no peripherals are requesting the clock source. Clock requests propagate from the peripheral, via the GCLK, to the clock source. If one or more peripheral is using a clock source, the clock source will be started/kept running. As soon as the clock source is no longer needed and no peripheral have an active request the clock source will be stopped until requested again. For the clock request to reach the clock source, the peripheral, the generic clock and the clock from the Generic Clock Generator in-between must be enabled. The time taken from a clock request being asserted to the clock source being ready is dependent on the clock source startup time, clock source frequency as well as the divider used in the Generic Clock Generator. The total startup time from a clock request to the clock is available for the peripheral is:

Delay_start_max = Clock source startup time + 2 * clock source periods + 2 * divided clock source periods

Delay_start_min = Clock source startup time + 1 * clock source period + 1 * divided clock source period

The delay for shutting down the clock source when there is no longer an active request is:

Delay_stop_min = 1 * divided clock source period + 1 * clock source period

Delay_stop_max = 2 * divided clock source periods + 2 * clock source periods

The On-Demand principle can be disabled individually for each clock source by clearing the ONDEMAND bit located in each clock source controller. The clock is always running whatever is the clock request. This has the effect to remove the clock source startup time at the cost of the power consumption.

In standby mode, the clock request mechanism is still working if the modules are configured to run in standby mode (RUNSTDBY bit).

13.6 Power Consumption vs Speed

Due to the nature of the asynchronous clocking of the peripherals there are some considerations that needs to be taken if either targeting a low-power or a fast-acting system. If clocking a peripheral with a very low clock, the active power consumption of the peripheral will be lower. At the same time the synchronization to the synchronous (CPU) clock domain is dependent on the peripheral clock speed, and will be longer with a slower peripheral clock; giving lower response time and more time waiting for the synchronization to complete.

13.7 Clocks after Reset

On any reset the synchronous clocks start to their initial state:

- OSC8M is enabled and divided by 8
- GCLK_MAIN uses OSC8M as source
- CPU and BUS clocks are undivided

On a power reset the GCLK starts to their initial state:

- All generic clock generators disabled except:
  - the generator 0 (GCLK_MAIN) using OSC8M as source, with no division

- the generator 2 using OSCULP32K as source, with no division
- All generic clocks disabled except:
  - the WDT generic clock using the generator 2 as source

On a user reset the GCLK starts to their initial state, except for:

- generic clocks that are write-locked (WRTLOCK is written to one prior to reset or the WDT generic clock if the WDT Always-On at power on bit set in the NVM User Row)
- The generic clock dedicated to the RTC if the RTC generic clock is enabled

On any reset the clock sources are reset to their initial state except the 32KHz clock sources which are reset only by a power reset.

14. GCLK – Generic Clock Controller

14.1 Overview

Several peripherals may require specific clock frequencies to operate correctly. The Generic Clock Controller consists of number of generic clock generators and generic clock multiplexers that can provide a wide range of clock frequencies. The generic clock generators can be set to use different external and internal clock sources. The selected clock can be divided down in the generic clock generator. The outputs from the generic clock generators are used as clock sources for the generic clock multiplexers, which select one of the sources to generate a generic clock (GCLK_PERIPHERAL), as shown in Figure 14-2. The number of generic clocks, m, depends on how many peripherals the device has.

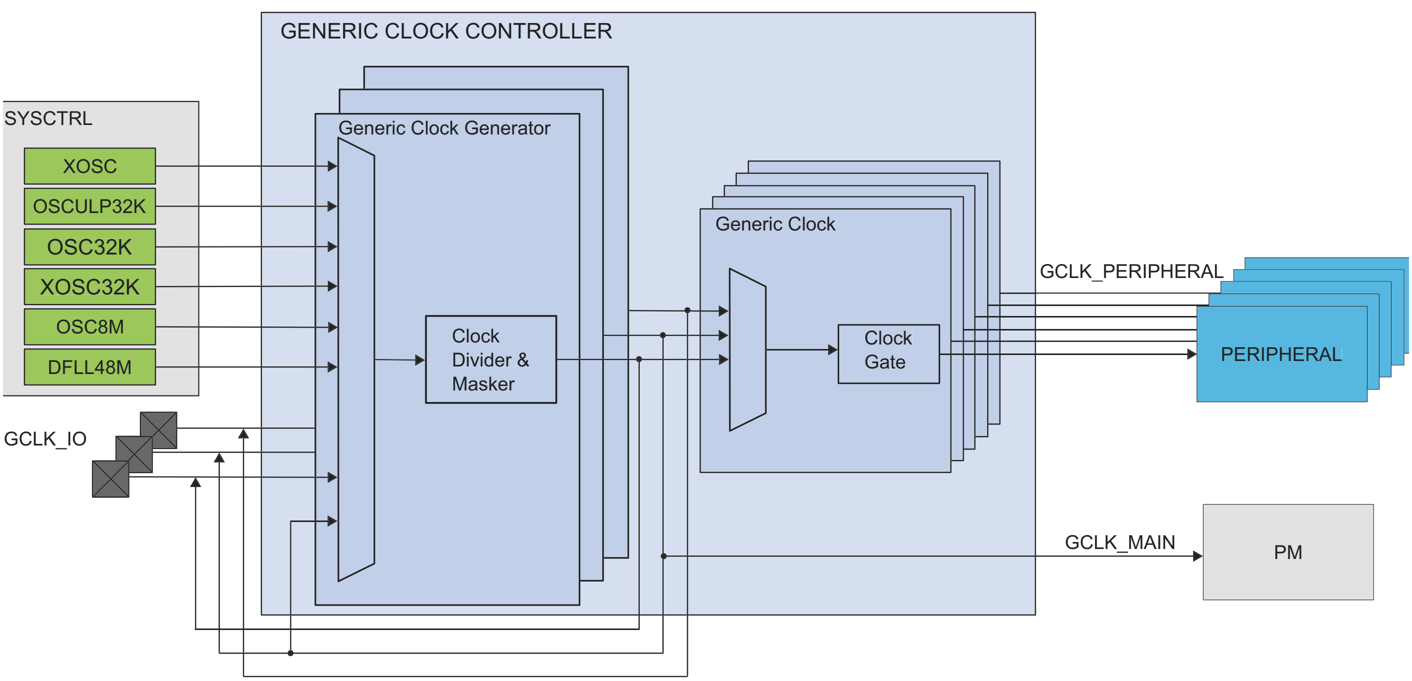
14.2 Features

- Provides generic clocks
- Wide frequency range
- Clock source for the generator can be changed on the fly

14.3 Block Diagram

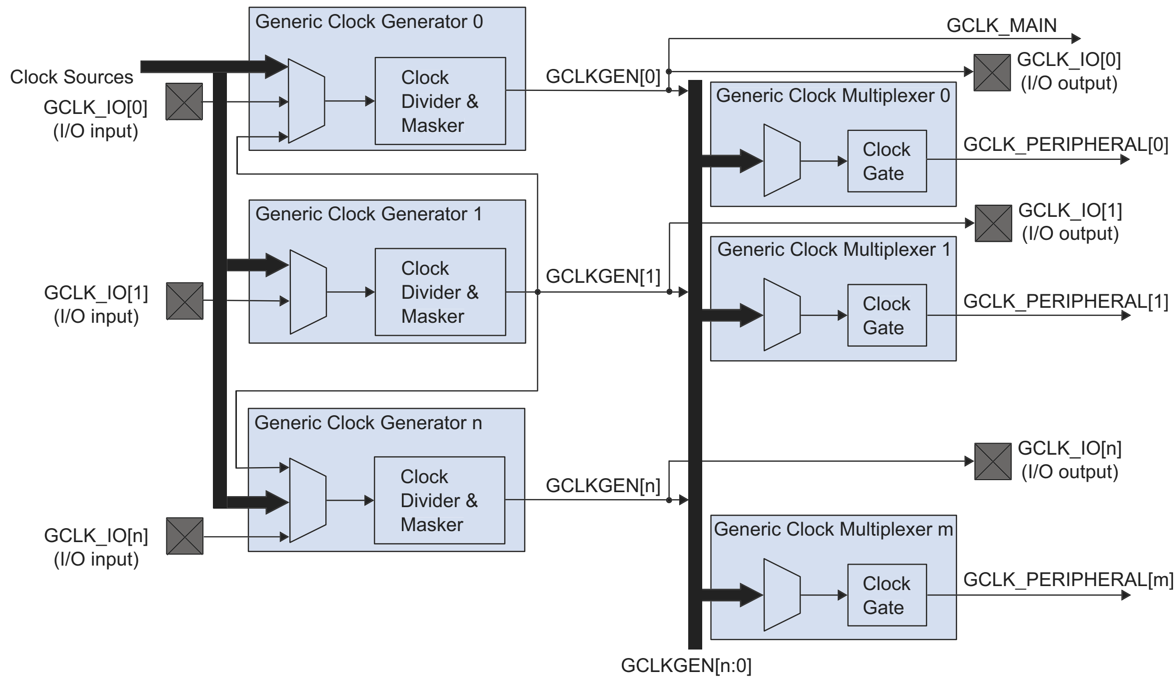
The Generic Clock Controller can be seen in the clocking diagram, which is shown in Figure 14-1 .

Figure 14-1. Device Clocking Diagram



The Generic Clock Controller block diagram is shown in Figure 14-2.

Figure 14-2. Generic Clock Controller Block Diagram[1]



Note: 1. If the GENCTRL.SRC=GCLKIN the GCLK_IO is set as an input.

14.4 Signal Description

Signal Name	Type	Description
GCLK_IO[5:0]	Digital I/O	Source clock when input Generic clock when output

Refer to "I/O Multiplexing and Considerations" on page 10 for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

14.5 Product Dependencies

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

14.5.1 I/O Lines

Using the Generic Clock Controller's I/O lines requires the I/O pins to be configured. Refer to "PORT" on page 375 for details.

14.5.2 Power Management

The Generic Clock Controller can operate in all sleep modes, if required. Refer to Table 15-4 for details on the different sleep modes.

14.5.3 Clocks

The Generic Clock Controller bus clock (CLK_GCLK_APB) can be enabled and disabled in the Power Manager, and the default state of CLK_GCLK_APB can be found in the Peripheral Clock Masking section in APBAMASK.

14.5.4 Interrupts

Not applicable.

14.5.5 Events

Not applicable.

14.5.6 Debug Operation

Not applicable.

14.5.7 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC).

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode or the CPU reset is extended, all write-protection is automatically disabled.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

14.5.8 Analog Connections

Not applicable.

14.6 Functional Description

14.6.1 Principle of Operation

The GCLK module is comprised of eight generic clock generators sourcing m generic clock multiplexers.

A clock source selected as input to one of the generic clock generators can be used directly, or it can be prescaled in the generic clock generator before the generator output is used as input to one or more of the generic clock multiplexers.

A generic clock multiplexer provides a generic clock to a peripheral (GCLK_PERIPHERAL). A generic clock can act as the clock to one or several of peripherals.

14.6.2 Basic Operation

14.6.2.1 Initialization

Before a generic clock is enabled, the clock source of its generic clock generator should be enabled. The generic clock must be configured as outlined by the following steps:

1. The generic clock generator division factor must be set by performing a single 32-bit write to the Generic Clock Generator Division register (GENDIV):
   - The generic clock generator that will be selected as the source of the generic clock must be written to the ID bit group (GENDIV.ID).
   - The division factor must be written to the DIV bit group (GENDIV.DIV)

   Refer to GENDIV register for details.
2. The generic clock generator must be enabled by performing a single 32-bit write to the Generic Clock Generator Control register (GENCTRL):
   - The generic clock generator that will be selected as the source of the generic clock must be written to the ID bit group (GENCTRL.ID)
   - The generic clock generator must be enabled by writing a one to the GENEN bit (GENCTRL.GENEN)

   Refer to GENCTRL register for details.

3. The generic clock must be configured by performing a single 16-bit write to the Generic Clock Control register (CLKCTRL):
   - The generic clock that will be configured must be written to the ID bit group (CLKCTRL.ID)
   - The generic clock generator used as the source of the generic clock must be written to the GEN bit group (CLKCTRL.GEN)

   Refer to CLKCTRL register for details.

14.6.2.2 Enabling, Disabling and Resetting

The GCLK module has no enable/disable bit to enable or disable the whole module.

The GCLK is reset by writing a one to the Software Reset bit in the Control register (CTRL.SWRST). All registers in the GCLK will be reset to their initial state except for generic clocks and associated generators that have their Write Lock bit written to one. Refer to "Configuration Lock" on page 90 for details.

14.6.2.3 Generic Clock Generator

Each generic clock generator (GCLKGEN) can be set to run from one of eight different clock sources except GCLKGEN[1] which can be set to run from one of seven sources. GCLKGEN[1] can act as source to the other generic clock generators but can not act as source to itself.

Each generic clock generator GCLKGEN[x] can be connected to one specific GCLK_IO[x] pin. The GCLK_IO[x] can be set to act as source to GCLKGEN[x] or GCLK_IO[x] can be set up to output the clock generated by GCLKGEN[x].

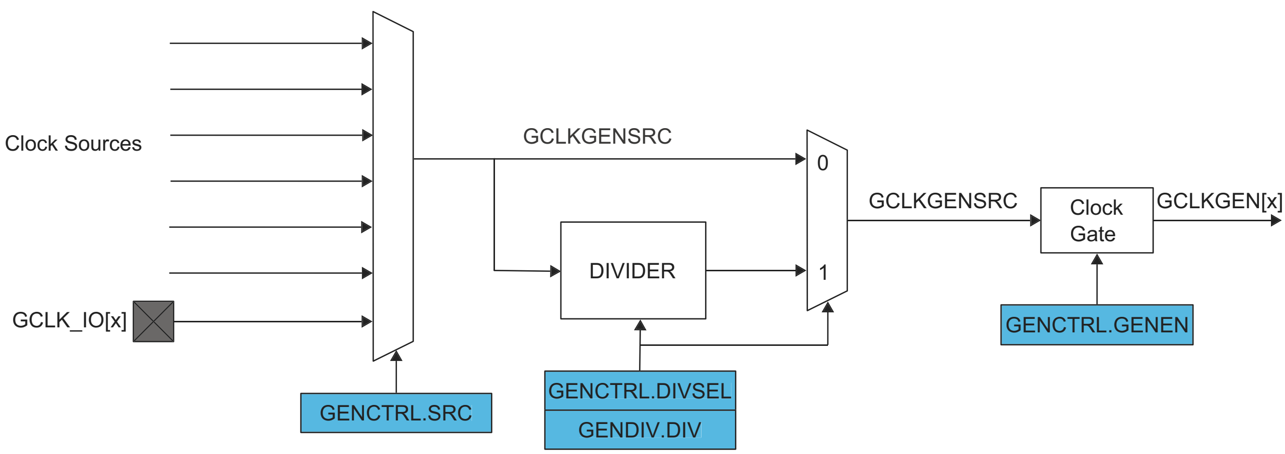
The selected source (GCLKGENSRC see Figure 14-3) can optionally be divided. Each generic clock generator can be independently enabled and disabled.

Each GCLKGEN clock can then be used as a clock source for the generic clock multiplexers. Each generic clock is allocated to one or several peripherals.

GCLKGEN[0], is used as GCLK_MAIN for the synchronous clock controller inside the Power Manager.

Refer to "PM – Power Manager" on page 107 for details on the synchronous clock generation.

Figure 14-3. Generic Clock Generator



14.6.2.4 Enabling a Generic Clock Generator

A generic clock generator is enabled by writing a one to the Generic Clock Generator Enable bit in the Generic Clock Generator Control register (GENCTRL.GENEN).

14.6.2.5 Disabling a Generic Clock Generator

A generic clock generator is disabled by writing a zero to GENCTRL.GENEN. When GENCTRL.GENEN is read as zero, the GCLKGEN clock is disabled and clock gated.

14.6.2.6 Selecting a Clock Source for the Generic Clock Generator

Each generic clock generator can individually select a clock source by writing to the Source Select bit group in GENCTRL (GENCTRL.SRC). Changing from one clock source, A, to another clock source, B, can be done on the fly. If clock source B is not ready, the generic clock generator will continue running with clock source A. As soon as clock source B is ready, however, the generic clock generator will switch to it. During the switching, the generic clock generator holds clock requests to clock sources A and B and then releases the clock source A request when the switch is done.

The available clock sources are device dependent (usually the crystal oscillators, RC oscillators, PLL and DFLL clocks). GCLKGEN[1] can be used as a common source for all the generic clock generators except generic clock generator 1.

14.6.2.7 Changing Clock Frequency

The selected generic clock generator source, GENCLKSRC can optionally be divided by writing a division factor in the Division Factor bit group in the Generic Clock Generator Division register (GENDIV.DIV).

Depending on the value of the Divide Selection bit in GENCTRL (GENCTRL.DIVSEL), it can be interpreted in two ways by the integer divider.

Note that the number of DIV bits for each generic clock generator is device dependent.

Refer to Table 14-10 for details.

14.6.2.8 Duty Cycle

When dividing a clock with an odd division factor, the duty-cycle will not be 50/50. Writing a one to the Improve Duty Cycle bit in GENCTRL (GENCTRL.IDC) will result in a 50/50 duty cycle.

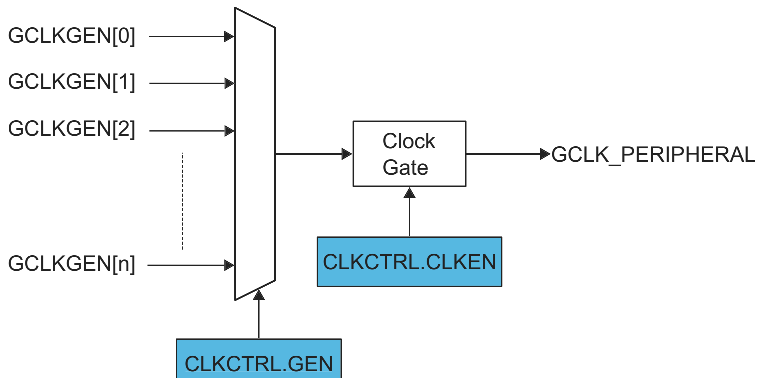
14.6.2.9 Generic Clock Output on I/O Pins

Each Generic Clock Generator's output can be directed to a GCLK_IO pin. If the Output Enable bit in GENCTRL (GENCTRL.OE) is one and the generic clock generator is enabled (GENCTRL.GENEN is one), the generic clock generator requests its clock source and the GCLKGEN clock is output to a GCLK_IO pin. If GENCTRL.OE is zero, GCLK_IO is set according to the Output Off Value bit. If the Output Off Value bit in GENCTRL (GENCTRL.OOV) is zero, the output clock will be low when generic clock generator is turned off. If GENCTRL.OOV is one, the output clock will be high when generic clock generator is turned off.

In standby mode, if the clock is output (GENCTRL.OE is one), the clock on the GCLK_IO pin is frozen to the OOV value if the Run In Standby bit in GENCTRL (GENCTRL.RUNSTDBY) is zero. If GENCTRL.RUNSTDBY is one, the GCLKGEN clock is kept running and output to GCLK_IO.

14.6.3 Generic Clock

Figure 14-4. Generic Clock Multiplexer



14.6.3.1 Enabling a Generic Clock

Before a generic clock is enabled, one of the generic clock generators must be selected as the source for the generic clock by writing to CLKCTRL.GEN. The clock source selection is individually set for each generic clock.

When a generic clock generator has been selected, the generic clock is enabled by writing a one to the Clock Enable bit in CLKCTRL (CLKCTRL.CLKEN). The CLKCTRL.CLKEN bit must be synchronized to the generic clock domain. CLKCTRL.CLKEN will continue to read as its previous state until the synchronization is complete.

14.6.3.2 Disabling a Generic Clock

A generic clock is disabled by writing a zero to CLKCTRL.CLKEN. The SYNCBUSY bit will be cleared when this write-synchronization is complete. CLKCTRL.CLKEN will continue to read as its previous state until the synchronization is complete. When the generic clock is disabled, the generic clock is clock gated.

14.6.3.3 Selecting a Clock Source for the Generic Clock

When changing a generic clock source by writing to CLKCTRL.GEN, the generic clock must be disabled before being re-enabled with the new clock source setting. This prevents glitches during the transition:

a. Write a zero to CLKCTRL.CLKEN
b. Wait until CLKCTRL.CLKEN reads as zero
c. Change the source of the generic clock by writing CLKCTRL.GEN
d. Re-enable the generic clock by writing a one to CLKCTRL.CLKEN

14.6.3.4 Configuration Lock

The generic clock configuration is locked for further write accesses by writing the Write Lock bit (WRTLOCK) in the CLKCTRL register. All writes to the CLKCTRL register will be ignored. It can only be unlocked by a power reset.

The generic clock generator sources of a locked generic clock are also locked. The corresponding GENCTRL and GENDIV are locked, and can be unlocked only by a power reset.

There is one exception concerning the GCLKGEN[0]. As it is used as GCLK_MAIN, it can not be locked. It is reset by any reset to startup with a known configuration.

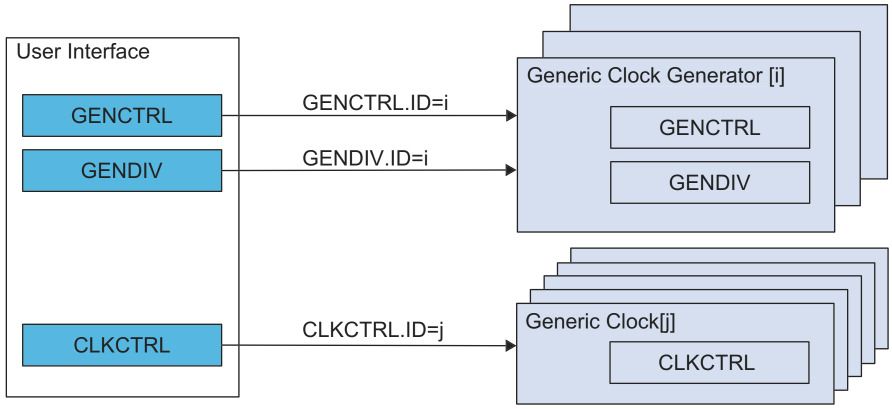
The SWRST can not unlock the registers.

14.6.4 Additional Features

14.6.4.1 Indirect Access

The Generic Clock Generator Control and Division registers (GENCTRL and GENDIV) and the Generic Clock Control register (CLKCTRL) are indirectly addressed as shown in Figure 14-5.

Figure 14-5. GCLK Indirect Access



Writing these registers is done by setting the corresponding ID bit group.

To read a register, the user must write the ID of the channel, i, in the corresponding register. The value of the register for the corresponding ID is available in the user interface by a read access.

For example, the sequence to read the GENCTRL register of generic clock generator i is:

a. Do an 8-bit write of the i value to GENCTRL.ID
b. Read GENCTRL

14.6.4.2 Generic Clock Enable after Reset

The Generic Clock Controller must be able to provide a generic clock to some specific peripherals after a reset. That means that the configuration of the generic clock generators and generic clocks after reset is device-dependent.

Refer to Table 14-8 and Table 14-9 for details on GENCTRL reset.

Refer to Table 14-12 and Table 14-13 for details on GENDIV reset.

Refer to Table 14-4 and Table 14-5 for details on CLKCTRL reset.

14.6.5 Sleep Mode Operation

14.6.5.1 SleepWalking

The GCLK module supports the SleepWalking feature. During a sleep mode where the generic clocks are stopped, a peripheral that needs its generic clock to execute a process must request it from the Generic Clock Controller.

The Generic Clock Controller will receive this request and then determine which generic clock generator is involved and which clock source needs to be awakened. It then wakes up the clock source, enables the generic clock generator and generic clock stages successively and delivers the generic clock to the peripheral.

14.6.5.2 Run in Standby Mode

In standby mode, the GCLK can continuously output the generic clock generator output to GCLK_IO.

Refer to "Generic Clock Output on I/O Pins" on page 89 for details.

14.6.6 Synchronization

Due to the asynchronicity between CLK_GCLK_APB and GCLKGENSRC some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set immediately, and cleared when synchronization is complete.

If an operation that requires synchronization is executed while STATUS.SYNCBUSY is one, the bus will be stalled. All operations will complete successfully, but the CPU will be stalled and interrupts will be pending as long as the bus is stalled.

The following registers need synchronization when written:

- Generic Clock Generator Control register (GENCTRL)
- Generic Clock Generator Division register (GENDIV)
- Control register (CTRL)

Write-synchronization is denoted by the Write-Synchronization property in the register description.

Refer to "Register Synchronization" on page 78 for further details.

14.7 Register Summary

Table 14-1. Register Summary

Offset	Name	Bit Pos.								
0x0	CTRL	7:0								SWRST
0x1	STATUS	7:0	SYNCBUSY							
0x2	CLKCTRL	7:0			ID[5:0]					
0x3		15:8	WRTLOCK	CLKEN			GEN[3:0]			
0x4	GENCTRL	7:0					ID[3:0]			
0x5		15:8				SRC[4:0]				
0x6		23:16			RUNSTDBY	DIVSEL	OE	OOV	IDC	GENEN
0x7		31:24								
0x8	GENDIV	7:0					ID[3:0]			
0x9		15:8	DIV[7:0]							
0xA		23:16	DIV[15:8]							
0xB		31:24								

14.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-protected property in each individual register description. Refer to "Register Access Protection" on page 87 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Write-Synchronized or the Read-Synchronized property in each individual register description. Refer to "Synchronization" on page 91 for details.

14.8.1 Control

Name: CTRL

Offset: 0x0

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
								SWRST
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – SWRST: Software Reset**
  0: There is no reset operation ongoing.
  1: There is a reset operation ongoing.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all registers in the GCLK to their initial state after a power reset, except for generic clocks and associated generators that have their WRTLOCK bit in CLKCTRL read as one.
  Refer to Table 14-8 for details on GENCTRL reset.
  Refer to Table 14-12 for details on GENDIV reset.
  Refer to Table 14-4 for details on CLKCTRL reset.
  Due to synchronization, there is a delay from writing CTRL.SWRST until the reset is complete. CTRL.SWRST and STATUS.SYNCBUSY will both be cleared when the reset is complete.

14.8.2 Status

Name: STATUS

Offset: 0x1

Reset: 0x00

Access: Read-Only

Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCBUSY							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – SYNCBUSY: Synchronization Busy Status**
  This bit is cleared when the synchronization of registers between the clock domains is complete.
  This bit is set when the synchronization of registers between clock domains is started.

- **Bits 6:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

14.8.3 Generic Clock Control

Name: CLKCTRL

Offset: 0x2

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
	WRTLOCK	CLKEN			GEN[3:0]			
Access	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			ID[5:0]					
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to configure one of the generic clocks, as specified in the CLKCTRL.ID bit group. To write to the CLKCTRL register, do a 16-bit write with all configurations and the ID.

To read the CLKCTRL register, first do an 8-bit write to the CLKCTRL.ID bit group with the ID of the generic clock whose configuration is to be read, and then read the CLKCTRL register.

- **Bit 15 – WRTLOCK: Write Lock**
  When this bit is written, it will lock from further writes the generic clock pointed to by CLKCTRL.ID, the generic clock generator pointed to in CLKCTRL.GEN and the division factor used in the generic clock generator. It can only be unlocked by a power reset.

  One exception to this is generic clock generator 0, which cannot be locked.

  0: The generic clock and the associated generic clock generator and division factor are not locked.

  1: The generic clock and the associated generic clock generator and division factor are locked.

- **Bit 14 – CLKEN: Clock Enable**
  This bit is used to enable and disable a generic clock.

  0: The generic clock is disabled.

  1: The generic clock is enabled.

- **Bits 13:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:8 – GEN[3:0]: Generic Clock Generator**

Table 14-2. Generic Clock Generator

GEN[3:0]	Name	Description
0x0	GCLK0	Generic clock generator 0
0x1	GCLK1	Generic clock generator 1
0x2	GCLK2	Generic clock generator 2

Table 14-2. Generic Clock Generator (Continued)

GEN[3:0]	Name	Description
0x3	GCLK3	Generic clock generator 3
0x4	GCLK4	Generic clock generator 4
0x5	GCLK5	Generic clock generator 5
0x6 - 0xF		Reserved

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – ID[5:0]: Generic Clock Selection ID**
  These bits select the Generic Clock that needs to be configured, as shown in Table 14-3.

Table 14-3. Generic Clock Selection ID

ID[5:0]	Name	Module Instance
0x0	GCLK_DFLL48M_REF	DFLL48MReference
0x1	GCLK_DPLL	FDPLL96M input clock source for reference
0x2	GCLK_DPLL_32K	FDPLL96M 32kHz clock for FDPLL96M internal lock timer
0x3	GCLK_WDT	WDT
0x4	GCLK_RTC	RTC
0x5	GCLK_EIC	EIC
0x6		Reserved
0x07	GCLK_EVSYS_CHANNEL_0	EVSYS_CHANNEL_0
0x08	GCLK_EVSYS_CHANNEL_1	EVSYS_CHANNEL_1
0x09	GCLK_EVSYS_CHANNEL_2	EVSYS_CHANNEL_2
0x0A	GCLK_EVSYS_CHANNEL_3	EVSYS_CHANNEL_3
0x0B	GCLK_EVSYS_CHANNEL_4	EVSYS_CHANNEL_4
0x0C	GCLK_EVSYS_CHANNEL_5	EVSYS_CHANNEL_5
0x0D	GCLK_SERCOMx_SLOW	SERCOMx_SLOW
0x0E	GCLK_SERCOM0_CORE	SERCOM0_CORE
0X0F	GCLK_SERCOM1_CORE	SERCOM1_CORE
0x10		Reserved
0x11		Reserved
0x12	GCLK_TC2	TC2
0x13	GCLK_ADC	ADC
0x14-0x3F		Reserved

A power reset will reset the CLKCTRL register for all IDs, including the RTC. If the WRTLOCK bit of the corresponding ID is zero and the ID is not the RTC, a user reset will reset the CLKCTRL register for this ID.

After a power reset, the reset value of the CLKCTRL register versus module instance is as shown in Table 14-4.

Table 14-4. CLKCTRL Reset Value after a Power Reset

Module Instance	Reset Value after a Power Reset		
	CLKCTRL.GEN	CLKCTRL.CLKEN	CLKCTRL.WRTLOCK
RTC	0x00	0x00	0x00
WDT	0x02	0x01 if WDT Enable bit in NVM User Row written to one 0x00 if WDT Enable bit in NVM User Row written to zero	0x01 if WDT Always-On bit in NVM User Row written to one 0x00 if WDT Always-On bit in NVM User Row written to zero
Others	0x00	0x00	0x00

After a user reset, the reset value of the CLKCTRL register versus module instance is as shown in Table 14-5.

Table 14-5. CLKCTRL Reset Value after a User Reset

Module Instance	Reset Value after a User Reset		
	CLKCTRL.GEN	CLKCTRL.CLKEN	CLKCTRL.WRTLOCK
RTC	0x00 if WRTLOCK=0 and CLKEN=0 No change if WRTLOCK=1 or CLKEN=1	0x00 if WRTLOCK=0 and CLKEN=0 No change if WRTLOCK=1 or CLKEN=1	No change
WDT	0x02 if WRTLOCK=0 No change if WRTLOCK=1	If WRTLOCK=0 0x01 if WDT Enable bit in NVM User Row written to one 0x00 if WDT Enable bit in NVM User Row written to zero If WRTLOCK=1 no change	No change
Others	0x00 if WRTLOCK=0 No change if WRTLOCK=1	0x00 if WRTLOCK=0 No change if WRTLOCK=1	No change

14.8.4 Generic Clock Generator Control

Name: GENCTRL

Offset: 0x4

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			RUNSTDBY	DIVSEL	OE	OOV	IDC	GENEN
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				SRC[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					ID[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to configure one of the generic clock generators, as specified in the GENCTRL.ID bit group. To write to the GENCTRL register, do a 32-bit write with all configurations and the ID.

To read the GENCTRL register, first do an 8-bit write to the GENCTRL.ID bit group with the ID of the generic clock generator those configuration is to be read, and then read the GENCTRL register.

- **Bits 31:22 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 21 – RUNSTDBY: Run in Standby**
  This bit is used to keep the generic clock generator running when it is configured to be output to its dedicated GCLK_IO pin. If GENCTRL.OE is zero, this bit has no effect and the generic clock generator will only be running if a peripheral requires the clock.

  0: The generic clock generator is stopped in standby and the GCLK_IO pin state (one or zero) will be dependent on the setting in GENCTRL.OOV.

  1: The generic clock generator is kept running and output to its dedicated GCLK_IO pin during standby mode.

- **Bit 20 – DIVSEL: Divide Selection**
  This bit is used to decide how the clock source used by the generic clock generator will be divided. If the clock source should not be divided, the DIVSEL bit must be zero and the GENDIV.DIV value for the corresponding generic clock generator must be zero or one.
  0: The generic clock generator equals the clock source divided by GENDIV.DIV.
  1: The generic clock generator equals the clock source divided by 2^(GENDIV.DIV+1).

- **Bit 19 – OE: Output Enable**
  This bit is used to enable output of the generated clock to GCLK_IO when GCLK_IO is not selected as a source in the GENCLK.SRC bit group.
  0: The generic clock generator is not output.
  1: The generic clock generator is output to the corresponding GCLK_IO, unless the corresponding GCLK_IO is selected as a source in the GENCLK.SRC bit group.

- **Bit 18 – OOV: Output Off Value**
  This bit is used to control the value of GCLK_IO when GCLK_IO is not selected as a source in the GENCLK.SRC bit group.
  0: The GCLK_IO will be zero when the generic clock generator is turned off or when the OE bit is zero.
  1: The GCLK_IO will be one when the generic clock generator is turned off or when the OE bit is zero.

- **Bit 17 – IDC: Improve Duty Cycle**
  This bit is used to improve the duty cycle of the generic clock generator when odd division factors are used.
  0: The generic clock generator duty cycle is not 50/50 for odd division factors.
  1: The generic clock generator duty cycle is 50/50.

- **Bit 16 – GENEN: Generic Clock Generator Enable**
  This bit is used to enable and disable the generic clock generator.
  0: The generic clock generator is disabled.
  1: The generic clock generator is enabled.

- **Bits 15:13 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 12:8 – SRC[4:0]: Source Select**
  hese bits define the clock source to be used as the source for the generic clock generator, as shown in Table 14-6.

Table 14-6. Source Select

Value	Name	Description
0x00	XOSC	XOSC oscillator output
0x01	GCLKIN	Generator input pad
0x02	GCLKGEN1	Generic clock generator 1 output
0x03	OSCULP32K	OSCULP32K oscillator output
0x04	OSC32K	OSC32K oscillator output
0x05	XOSC32K	XOSC32K oscillator output

Table 14-6. Source Select (Continued)

Value	Name	Description
0x06	OSC8M	OSC8M oscillator output
0x07	DFLL48M	DFLL48M output
0x08	FDPLL96M	FDPLL96M output
0x09-0x1F	Reserved	Reserved for future use

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 3:0 – ID[3:0]: Generic Clock Generator Selection**
  These bits select the generic clock generator that will be configured or read. The value of the ID bit group versus which generic clock generator is configured is shown in Table 14-7.

Table 14-7. Generic Clock Generator Selection

Value	Name	Description
0x0	GCLKGEN0	Generic clock generator 0
0x1	GCLKGEN1	Generic clock generator 1
0x2	GCLKGEN2	Generic clock generator 2
0x3	GCLKGEN3	Generic clock generator 3
0x4	GCLKGEN4	Generic clock generator 4
0x5	GCLKGEN5	Generic clock generator 5
0x6 - 0xF		Reserved

A power reset will reset the GENCTRL register for all IDs, including the generic clock generator used by the RTC. If a generic clock generator ID other than generic clock generator 0 is not a source of a “locked” generic clock or a source of the RTC generic clock, a user reset will reset the GENCTRL for this ID.

After a power reset, the reset value of the GENCTRL register is as shown in Table 14-8.

Table 14-8. GENCTRL Reset Value after a Power Reset

GCLK Generator ID	Reset Value after a Power Reset
0x00	0x00010600
0x01	0x00000001
0x02	0x00010302
0x03	0x00000003
0x04	0x00000004
0x05	0x00000005

After a user reset, the reset value of the GENCTRL register is as shown in Table 14-9.

Table 14-9. GENCTRL Reset Value after a User Reset

GCLK Generator ID	Reset Value after a User Reset
0x00	0x00010600
0x01	0x00000001 if the generator is not used by the RTC No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one
0x02	0x00010302 if the generator is not used by the RTC No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one
0x03	0x00000003 if the generator is not used by the RTC No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one
0x04	0x00000004 if the generator is not used by the RTC No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one
0x05	0x00000005 if the generator is not used by the RTC No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one

14.8.5 Generic Clock Generator Division

Name: GENDIV

Offset: 0x8

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIV[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					ID[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to configure one of the generic clock generators, as specified in the GENDIV.ID bit group. To write to the GENDIV register, do a 32-bit write with all configurations and the ID.

To read the GENDIV register, first do an 8-bit write to the GENDIV.ID bit group with the ID of the generic clock generator whose configuration is to be read, and then read the GENDIV register.

- **Bits 31:24 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 23:8 – DIV[15:0]: Division Factor**
  These bits apply a division on each selected generic clock generator. The number of DIV bits each generator has can be seen in Table 14-10. Writes to bits above the specified number will be ignored.

Table 14-10. Division Factor

Generator	Division Factor Bits
Generic clock generator 0	8 division factor bits - DIV[7:0]
Generic clock generator 1	16 division factor bits - DIV[15:0]
Generic clock generators 2	5 division factor bits - DIV[4:0]
Generic clock generators 3 - 8	8 division factor bits - DIV[7:0]

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 3:0 – ID[3:0]: Generic Clock Generator Selection**
  These bits select the generic clock generator on which the division factor will be applied, as shown in Table 14-11.

Table 14-11. Generic Clock Generator Selection

ID[3:0]	Name	Description
0x0	GCLKGEN0	Generic clock generator 0
0x1	GCLKGEN1	Generic clock generator 1
0x2	GCLKGEN2	Generic clock generator 2
0x3	GCLKGEN3	Generic clock generator 3
0x4	GCLKGEN4	Generic clock generator 4
0x5	GCLKGEN5	Generic clock generator 5
0x6 - 0xF		Reserved

A power reset will reset the GENDIV register for all IDs, including the generic clock generator used by the RTC. If a generic clock generator ID other than generic clock generator 0 is not a source of a ,“locked” generic clock or a source of the RTC generic clock, a user reset will reset the GENDIV for this ID.

After a power reset, the reset value of the GENDIV register is as shown in Table 14-12.

Table 14-12. GENDIV Reset Value after a Power Reset

GCLK Generator ID	Reset Value after a Power Reset
0x00	0x00000000
0x01	0x00000001
0x02	0x00000002
0x03	0x00000003
0x04	0x00000004
0x05	0x00000005

After a user reset, the reset value of the GENDIV register is as shown in Table 14-13.

Table 14-13. GENDIV Reset Value after a User Reset

GCLK Generator ID	Reset Value after a User Reset
0x00	0x00000000
0x01	0x00000001 if the generator is not used by the RTC and not a source of a 'locked' generic clock No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one
0x02	0x00000002 if the generator is not used by the RTC and not a source of a 'locked' generic clock No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one
0x03	0x00000003 if the generator is not used by the RTC and not a source of a 'locked' generic clock No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one
0x04	0x00000004 if the generator is not used by the RTC and not a source of a 'locked' generic clock No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one
0x05	0x00000005 if the generator is not used by the RTC and not a source of a 'locked' generic clock No change if the generator is used by the RTC or used by a GCLK with a WRTLOCK as one

15. PM – Power Manager

15.1 Overview

The Power Manager (PM) controls the reset, clock generation and sleep modes of the microcontroller.

Utilizing a main clock chosen from a large number of clock sources from the GCLK, the clock controller provides synchronous system clocks to the CPU and the modules connected to the AHB and the APBx bus. The synchronous system clocks are divided into a number of clock domains; one for the CPU and AHB and one for each APBx. Any synchronous system clock can be changed at run-time during normal operation. The clock domains can run at different speeds, enabling the user to save power by running peripherals at a relatively low clock frequency, while maintaining high CPU performance. In addition, the clock can be masked for individual modules, enabling the user to minimize power consumption. If for some reason the main clock stops oscillating, the clock failure detector allows switching the main clock to the safe OSC8M clock.

Before entering the STANDBY sleep mode the user must make sure that a significant amount of clocks and peripherals are disabled, so that the voltage regulator is not overloaded. This is because during STANDBY sleep mode the internal voltage regulator will be in low power mode.

Various sleep modes and clock gating are provided in order to fit power consumption requirements. This enables the microcontroller to stop unused modules to save power. In ACTIVE mode, the CPU is executing application code. When the device enters a sleep mode, program execution is stopped and some modules and clock domains are automatically switched off by the PM according to the sleep mode. The application code decides which sleep mode to enter and when. Interrupts from enabled peripherals and all enabled reset sources can restore the microcontroller from a sleep mode to ACTIVE mode.

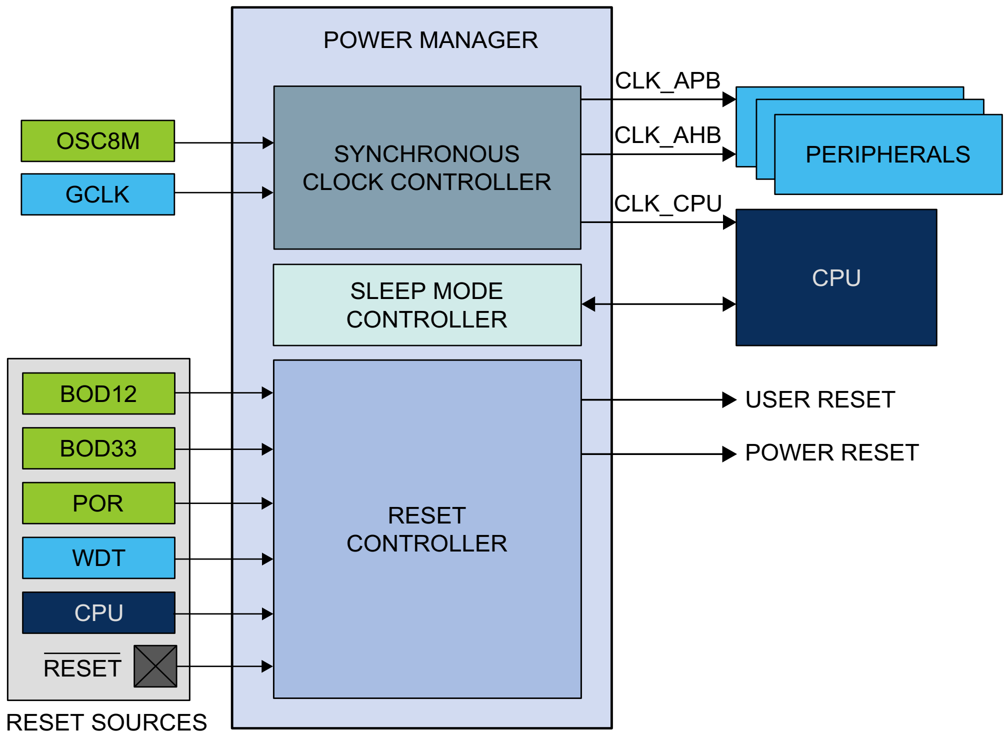
The PM also contains a reset controller, which collects all possible reset sources. It issues a microcontroller reset and sets the device to its initial state, and allows the reset source to be identified by software.

15.2 Features

- Reset control
  - Reset the microcontroller and set it to an initial state according to the reset source
  - Multiple reset sources
    - Power reset sources: POR, BOD12, BOD33
    - User reset sources: External reset ($\overline{\text{RESET}}$), Watchdog Timer reset, software reset
  - Reset status register for reading the reset source from the application code
- Clock control
  - Controls CPU, AHB and APB system clocks
    - Multiple clock sources and division factor from GCLK
    - Clock prescaler with 1x to 128x division
  - Safe run-time clock switching from GCLK
  - Module-level clock gating through maskable peripheral clocks
  - Clock failure detector
- Power management control
  - Sleep modes: IDLE, STANDBY
  - SleepWalking support on GCLK clocks

15.3 Block Diagram

Figure 15-1. PM Block Diagram



15.4 Signal Description

Signal Name	Type	Description
RESET	Digital input	External reset

Refer to "I/O Multiplexing and Considerations" on page 10 for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

15.5 Product Dependencies

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

15.5.1 I/O Lines

Not applicable.

15.5.2 Power Management

Not applicable.

15.5.3 Clocks

The PM bus clock (CLK_PM_APB) can be enabled and disabled in the power manager, and the default state of CLK_PM_APB can be found in Table 15-1. If this clock is disabled in the Power Manager, it can only be re-enabled by a reset.

A generic clock (GCLK_MAIN) is required to generate the main clock. The clock source for GCLK_MAIN is configured by default in the Generic Clock Controller, and can be re-configured by the user if needed. Refer to "GCLK – Generic Clock Controller" on page 85 for details.

15.5.3.1 Main Clock

The main clock (CLK_MAIN) is the common source for the synchronous clocks. This is fed into the common 8-bit prescaler that is used to generate synchronous clocks to the CPU, AHB and APBx modules.

15.5.3.2 CPU Clock

The CPU clock (CLK_CPU) is routed to the CPU. Halting the CPU clock inhibits the CPU from executing instructions.

15.5.3.3 AHB Clock

The AHB clock (CLK_AHB) is the root clock source used by peripherals requiring an AHB clock. The AHB clock is always synchronous to the CPU clock and has the same frequency, but may run even when the CPU clock is turned off. A clock gate is inserted from the common AHB clock to any AHB clock of a peripheral.

15.5.3.4 APBx Clocks

The APBx clock (CLK_APBX) is the root clock source used by modules requiring a clock on the APBx bus. The APBx clock is always synchronous to the CPU clock, but can be divided by a prescaler, and will run even when the CPU clock is turned off. A clock gater is inserted from the common APB clock to any APBx clock of a module on APBx bus.

15.5.4 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the PM interrupt requires the Interrupt Controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

15.5.5 Events

Not applicable.

15.5.6 Debug Operation

When the CPU is halted in debug mode, the PM continues normal operation. In sleep mode, the clocks generated from the PM are kept running to allow the debugger accessing any modules. As a consequence, power measurements are not possible in debug mode.

15.5.7 Register Access Protection

All registers with write access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Interrupt Flag register (INTFLAG). Refer to INTFLAG for details
- Reset Cause register (RCAUSE). Refer to RCAUSE for details

Write-protection is denoted by the Write-Protection property in the register description.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

15.5.8 Analog Connections

Not applicable.

15.6 Functional Description

15.6.1 Principle of Operation

15.6.1.1 Synchronous Clocks

The GCLK_MAIN clock from GCLK module provides the source for the main clock, which is the common root for the synchronous clocks for the CPU and APBx modules. The main clock is divided by an 8-bit prescaler, and each of the derived clocks can run from any tapping off this prescaler or the undivided main clock, as long as $f_{CPU} \geq f_{APBx}$. The synchronous clock source can be changed on the fly to respond to varying load in the application. The clocks for each module in each synchronous clock domain can be individually masked to avoid power consumption in inactive modules. Depending on the sleep mode, some clock domains can be turned off (see Table 15-4 on page 116).

15.6.1.2 Reset Controller

The Reset Controller collects the various reset sources and generates reset for the device. 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.

15.6.1.3 Sleep Mode Controller

In ACTIVE mode, all clock domains are active, allowing software execution and peripheral operation. The PM Sleep Mode Controller allows the user to choose between different sleep modes depending on application requirements, to save power (see Table 15-4 on page 116).

15.6.2 Basic Operation

15.6.2.1 Initialization

After a power-on reset, the PM is enabled and the Reset Cause (RCAUSE - refer to RCAUSE for details) register indicates the POR source. The default clock source of the GCLK_MAIN clock is started and calibrated before the CPU starts running. The GCLK_MAIN clock is selected as the main clock without any division on the prescaler. The device is in the ACTIVE mode.

By default, only the necessary clocks are enabled (see Table 15-1).

15.6.2.2 Enabling, Disabling and Resetting

The PM module is always enabled and can not be reset.

15.6.2.3 Selecting the Main Clock Source

Refer to "GCLK – Generic Clock Controller" on page 85 for details on how to configure the main clock source.

15.6.2.4 Selecting the 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 the CPU Prescaler Selection bits in the CPU Select register (CPUSEL.CPUDIV), resulting in a CPU clock frequency determined by this equation:

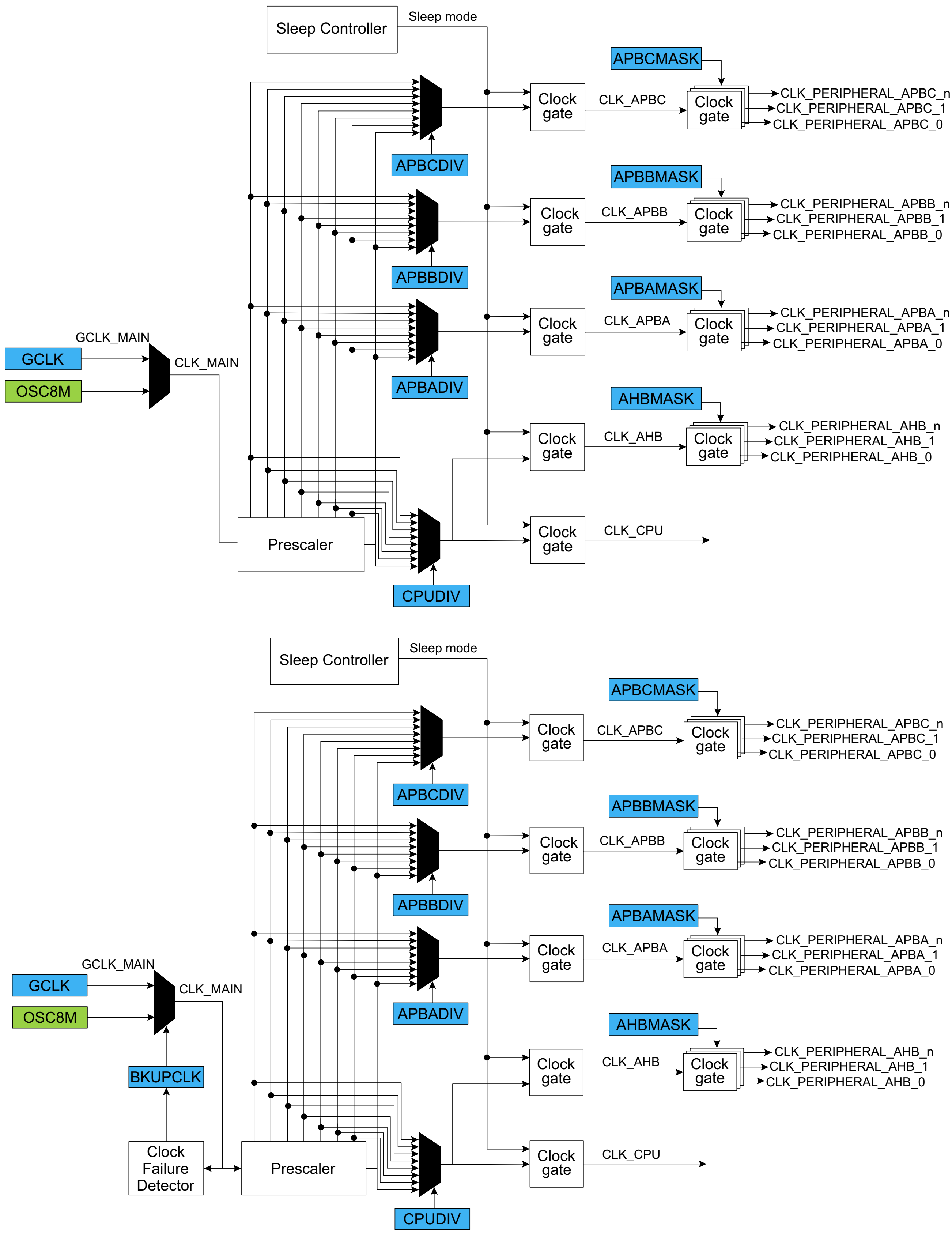
$$f_{CPU} = \frac{f_{main}}{2^{CPUDIV}}$$

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

Note that the AHB clock is always equal to the CPU clock.

CPUSEL and APBxSEL can be written without halting or disabling peripheral modules. Writing CPUSEL and APBxSEL allows a new clock setting to be written to all synchronous clocks at the same time. It is possible to keep one or more clocks unchanged. This way, it is possible to, for example, scale the CPU speed according to the required performance, while keeping the APBx frequency constant.

Figure 15-2. Synchronous Clock Selection and Prescaler



15.6.2.5 Clock Ready Flag

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

15.6.2.6 Peripheral Clock Masking

It is possible to disable or enable the clock for a peripheral in the AHB or APBx clock domain by writing the corresponding bit in the Clock Mask register (APBxMASK - refer to APBAMASK for details) to zero or one. Refer to Table 15-1 for the default state of each of the peripheral clocks.

Table 15-1. Peripheral Clock Default State

Peripheral Clock	Default State
CLK_PAC0_APB	Enabled
CLK_PM_APB	Enabled
CLK_SYSCTRL_APB	Enabled
CLK_GCLK_APB	Enabled
CLK_WDT_APB	Enabled
CLK_RTC_APB	Enabled
CLK_EIC_APB	Enabled
CLK_PAC1_APB	Enabled
CLK_DSU_APB	Enabled
CLK_NVMCTRL_APB	Enabled
CLK_PORT_APB	Enabled
CLK_HMATRIX_APB	Enabled
CLK_PAC2_APB	Disabled
CLK_SERCOMx_APB	Disabled
CLK_TCx_APB	Disabled
CLK_ADC_APB	Enabled
CLK_DMAC_APB	Enabled

When the APB clock for a module is not provided its registers cannot be read or written. The module can be re-enabled later by writing the corresponding mask bit to one.

A module may be connected to several clock domains (for instance, AHB and APB), in which case it will have several mask bits.

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 NVM Controller (NVMCTRL) will cause a problem if the CPU needs to read from the flash memory. Switching off the clock to the Power Manager (PM), which contains the mask registers, or the corresponding APBx bridge, will make it impossible to write the mask registers again. In this case, they can only be re-enabled by a system reset.

15.6.2.7 Clock Failure Detector

This mechanism allows the main clock to be switched automatically to the safe OSC8M clock when the main clock source is considered off. This may happen for instance when an external crystal oscillator is selected as the clock source for the main clock and the crystal fails. The mechanism is to designed to detect, during a OSCULP32K clock period, at least one rising edge of the main clock. If no rising edge is seen, the clock is considered failed.

The clock failure detector is enabled by writing a one to the Clock Failure Detector Enable bit in CTRL (CFDEN_CTRL). Refer to CTRL for detailed information.

As soon as the Clock Failure Detector Enable bit (CTRL.CFDEN) is one, the clock failure detector (CFD) will monitor the undivided main clock. When a clock failure is detected, the main clock automatically switches to the OSC8M clock and the Clock Failure Detector flag in the interrupt Flag Status and Clear register (INTFLAG.CFD) is set and the corresponding interrupt request will be generated if enabled. The BKUPCLK bit in the CTRL register is set by hardware to indicate that the main clock comes from OSC8M. The GCLK_MAIN clock source can be selected again by writing a zero to the CTRL.BKUPCLK bit. Writing the bit does not fix the failure, however.

Note 1: The detector does not monitor while the main clock is temporarily unavailable (startup time after a wake-up, etc.) or in sleep mode. The Clock Failure Detector must be disabled before entering standby mode.

Note 2: The clock failure detector must not be enabled if the source of the main clock is not significantly faster than the OSCULP32K clock. For instance, if GCLK_MAIN is the internal 32kHz RC, then the clock failure detector must be disabled.

Note 3: The OSC8M internal oscillator should be enabled to allow the main clock switching to the OSC8M clock.

15.6.2.8 Reset Controller

The latest reset cause is available in RCAUSE, and can be read during the application boot sequence in order to determine proper action.

There are two groups of reset sources:

- Power Reset: Resets caused by an electrical issue.
- User Reset: Resets caused by the application.

The table below lists the parts of the device that are reset, depending on the reset type.

Table 15-2. Effects of the Different Reset Events

	Power Reset	User Reset	
	POR, BOD12, BOD33	External Reset	WDT Reset, SysResetReq
RTC All the 32kHz sources WDT with ALWAYSON feature Generic Clock with WRTLOCK feature	Y	N	N
Debug logic	Y	Y	N
Others	Y	Y	Y

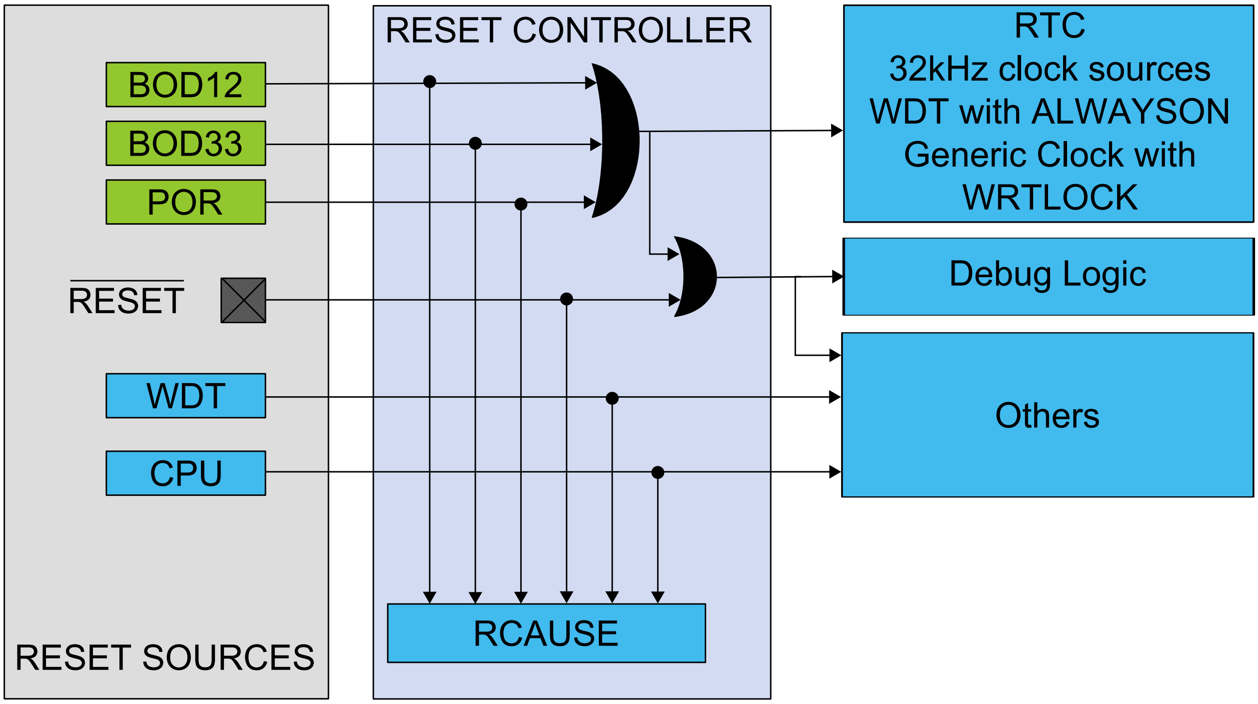
The external reset is generated when pulling the $\overline{\text{RESET}}$ pin low. This pin has an internal pull-up, and does not need to be driven externally during normal operation.

The POR, BOD12 and BOD33 reset sources are generated by their corresponding module in the System Controller Interface (SYSCTRL).

The WDT reset is generated by the Watchdog Timer.

The System Reset Request (SysResetReq) is a software reset generated by the CPU when asserting the SYSRESETREQ bit located in the Reset Control register of the CPU (See the ARM® Cortex® Technical Reference Manual on http://www.arm.com).

Figure 15-3. Reset Controller



15.6.2.9 Sleep Mode Controller

Sleep mode is activated by the Wait For Interrupt instruction (WFI). The Idle bits in the Sleep Mode register (SLEEP.IDLE) and the SLEEPDEEP bit of the System Control register of the CPU should be used as argument to select the level of the sleep mode.

There are two main types of sleep mode:

- IDLE mode: The CPU is stopped. Optionally, some synchronous clock domains are stopped, depending on the IDLE argument. Regulator operates in normal mode.
- STANDBY mode: All clock sources are stopped, except those where the RUNSTDBY bit is set. Regulator operates in low-power mode. Before entering standby mode the user must make sure that a significant amount of clocks and peripherals are disabled, so that the voltage regulator is not overloaded.

Table 15-3. Sleep Mode Entry and Exit Table

Mode	Level	Mode Entry	Wake-Up Sources
IDLE	0	SCR.SLEEPDEEP = 0 SLEEP.IDLE=Level WFI	Synchronous[2] (APB, AHB), asynchronous[1]
	1		Synchronous (APB), asynchronous
	2		Asynchronous
STANDBY		SCR.SLEEPDEEP = 1 WFI	Asynchronous

Notes: 1. Asynchronous: interrupt generated on generic clock or external clock or external event.
2. Synchronous: interrupt generated on the APB clock.

Table 15-4. Sleep Mode Overview

Sleep Mode	CPU Clock	AHB Clock	APB Clock	Oscillators				Main Clock	Regulator Mode	RAM Mode
				ONDEMAND = 0		ONDEMAND = 1				
				RUNSTDBY=0	RUNSTDBY=1	RUNSTDBY=0	RUNSTDBY=1			
Idle 0	Stop	Run	Run	Run	Run	Run if requested	Run if requested	Run	Normal	Normal
Idle 1	Stop	Stop	Run	Run	Run	Run if requested	Run if requested	Run	Normal	Normal
Idle 2	Stop	Stop	Stop	Run	Run	Run if requested	Run if requested	Run	Normal	Normal
Standby	Stop	Stop	Stop	Stop	Run	Stop	Run if requested	Stop	Low power	Low power

IDLE Mode

The IDLE modes allow power optimization with the fastest wake-up time.

The CPU is stopped. To further reduce power consumption, the user can disable the clocking of modules and clock sources by configuring the SLEEP.IDLE bit group. The module will be halted regardless of the bit settings of the mask registers in the Power Manager (PM.AHBMASK, PM.APBxMASK).

Regulator operates in normal mode.

- Entering IDLE mode: The IDLE mode is entered by executing the WFI instruction. Additionally, if the SLEEPONEXIT bit in the ARM Cortex System Control register (SCR) is set, the IDLE mode will also be entered when the CPU exits the lowest priority ISR. This mechanism can be useful for applications that only require the processor to run when an interrupt occurs. Before entering the IDLE mode, the user must configure the IDLE mode configuration bit group and must write a zero to the SCR.SLEEPDEEP bit.
- Exiting IDLE mode: The processor wakes the system up when it detects the occurrence of any interrupt that is not masked in the NVIC Controller with sufficient priority to cause exception entry. The system goes back to the ACTIVE mode. The CPU and affected modules are restarted.

STANDBY Mode

The STANDBY mode allows achieving very low power consumption.

In this mode, all clocks are stopped except those which are kept running if requested by a running module or have the ONDEMAND bit set to zero. For example, the RTC can operate in STANDBY mode. In this case, its Generic Clock clock source will also be enabled.

The regulator and the RAM operate in low-power mode.

A SLEEPONEXIT feature is also available.

- Entering STANDBY mode: This mode is entered by executing the WFI instruction with the SCR.SLEEPDEEP bit of the CPU is written to 1.
- Exiting STANDBY mode: Any peripheral able to generate an asynchronous interrupt can wake up the system. For example, a module running on a Generic clock can trigger an interrupt. When the enabled asynchronous wake-up event occurs and the system is woken up, the device will either execute the interrupt service routine or

continue the normal program execution according to the Priority Mask Register (PRIMASK) configuration of the CPU.

15.6.3 SleepWalking

SleepWalking is the capability for a device to temporarily wakeup clocks for peripheral to perform a task without waking-up the CPU in STANDBY sleep mode. At the end of the sleepwalking task, the device can either be waken-up by an interrupt (from a peripheral involved in SleepWalking) or enter again into STANDBY sleep mode.

In Atmel SAM D09 devices, SleepWalking is supported only on GCLK clocks by using the on-demand clock principle of the clock sources. Refer to "On-demand, Clock Requests" on page 83 for more details.

15.6.4 Interrupts

The peripheral has the following interrupt sources:

- Clock Ready flag
- Clock failure detector

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAG) register is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register. An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the peripheral is reset. An interrupt flag is cleared by writing a one to the corresponding bit in the INTFLAG register. Each peripheral can have one interrupt request line per interrupt source or one common interrupt request line for all the interrupt sources. Refer to "Nested Vector Interrupt Controller" on page 23 for details. If the peripheral has one common interrupt request line for all the interrupt sources, the user must read the INTFLAG register to determine which interrupt condition is present.

15.6.5 Events

Not applicable.

15.6.6 Sleep Mode Operation

In all IDLE sleep modes, the power manager is still running on the selected main clock.

In STANDDBY sleep mode, the power manager is frozen and is able to go back to ACTIVE mode upon any asynchronous interrupt.

15.7 Register Summary

Table 15-5. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0				BKUPCLK		CFDEN		
0x01	SLEEP	7:0							IDLE[1:0]	
0x02	EXTCTRL	7:0								SETDIS
0x03 ... 0x07	Reserved									
0x08	CPUSEL	7:0						CPUDIV[2:0]		
0x09	APBASEL	7:0						APBADIV[2:0]		
0x0A	APBBSEL	7:0						APBBDIV[2:0]		
0x0B	APBCSEL	7:0						APBCDIV[2:0]		
0x0C ... 0x13	Reserved									
0x14	AHBMASK	7:0			DMAC	NVMCTRL	DSU	HPB2	HPB1	HPB0
0x15		15:8								
0x16		23:16								
0x17		31:24								
0x18	APBAMASK	7:0		EIC	RTC	WDT	GCLK	SYSCTRL	PM	PAC0
0x19		15:8								
0x1A		23:16								
0x1B		31:24								
0x1C	APBBMASK	7:0				DMAC	PORT	NVMCTRL	DSU	PAC1
0x1D		15:8								
0x1E		23:16								
0x1F		31:24								
0x20	APBCMASK	7:0	TC2	TC1			SERCOM1	SERCOM0	EVSYS	PAC2
0x21		15:8								ADC
0x22		23:16								
0x23		31:24								
0x24 ... 0x33	Reserved									
0x34	INTENCLR	7:0							CFD	CKRDY
0x35	INTENSET	7:0							CFD	CKRDY
0x36	INTFLAG	7:0							CFD	CKRDY
0x37	Reserved									
0x38	RCAUSE	7:0		SYST	WDT	EXT		BOD33	BOD12	POR

15.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, and the 8-bit halves of a 16-bit register can be accessed directly.

Exception for APBASEL, APBBSEL and APBCSEL: These registers must only be accessed with 8-bit access.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to "Register Access Protection" on page 109 for details.

15.8.1 Control

Name: CTRL

Offset: 0x00

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
				BKUPCLK		CFDEN		
Access	R	R	R	R/W	R	R/W	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 4 – BKUPCLK: Backup Clock Select**
  This bit is set by hardware when a clock failure is detected.
  0: The GCLK_MAIN clock is selected for the main clock.
  1: The OSC8M backup clock is selected for the main clock.

- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 2 – CFDEN: Clock Failure Detector Enable**
  0: The clock failure detector is disabled.
  1: The clock failure detector is enabled.

- **Bits 1:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

15.8.2 Sleep Mode

Name: SLEEP

Offset: 0x01

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
							IDLE[1:0]	
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 1:0 – IDLE[1:0]: Idle Mode Configuration**
  These bits select the Idle mode configuration after a WFI instruction.

Table 15-6. Idle Mode Configuration

IDLE[1:0]	Name	Description
0x0	CPU	The CPU clock domain is stopped
0x1	AHB	The CPU and AHB clock domains are stopped
0x2	APB	The CPU, AHB and APB clock domains are stopped
0x3		Reserved

15.8.3 External Reset Controller

Name: EXTCTRL

Offset: 0x02

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
								SETDIS
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 0 – SETDIS: External Reset Disable**

15.8.4 CPU Clock Select

Name: CPUSEL

Offset: 0x08

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
						CPUDIV[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – CPUDIV[2:0]: CPU Prescaler Selection**
  These bits define the division ratio of the main clock prescaler (2^n).

Table 15-7. CPU Prescaler Selection

CPUDIV[2:0]	Name	Description
0x0	DIV1	Divide by 1
0x1	DIV2	Divide by 2
0x2	DIV4	Divide by 4
0x3	DIV8	Divide by 8
0x4	DIV16	Divide by 16
0x5	DIV32	Divide by 32
0x6	DIV64	Divide by 64
0x7	DIV128	Divide by 128

15.8.5 APBA Clock Select

Name: APBASEL

Offset: 0x09

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
						APBADIV[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – APBADIV[2:0]: APBA Prescaler Selection**
  These bits define the division ratio of the APBA clock prescaler (2^n).

Table 15-8. APBA Prescaler Selection

APBADIV[2:0]	Name	Description
0x0	DIV1	Divide by 1
0x1	DIV2	Divide by 2
0x2	DIV4	Divide by 4
0x3	DIV8	Divide by 8
0x4	DIV16	Divide by 16
0x5	DIV32	Divide by 32
0x6	DIV64	Divide by 64
0x7	DIV128	Divide by 128

15.8.6 APBB Clock Select

Name: APBBSEL

Offset: 0x0A

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
						APBBDIV[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 2:0 – APBBDIV[2:0]: APBB Prescaler Selection**
  These bits define the division ratio of the APBB clock prescaler (2^n).

Table 15-9. APBB Prescaler Selection

APBBDIV[2:0]	Name	Description
0x0	DIV1	Divide by 1
0x1	DIV2	Divide by 2
0x2	DIV4	Divide by 4
0x3	DIV8	Divide by 8
0x4	DIV16	Divide by 16
0x5	DIV32	Divide by 32
0x6	DIV64	Divide by 64
0x7	DIV128	Divide by 128

15.8.7 APBC Clock Select

Name: APBCSEL

Offset: 0x0B

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
						APBCDIV[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – APBCDIV[2:0]: APBC Prescaler Selection**
  These bits define the division ratio of the APBC clock prescaler (2^n).

Table 15-10. APBC Prescaler Selection

APBCDIV[2:0]	Name	Description
0x0	DIV1	Divide by 1
0x1	DIV2	Divide by 2
0x2	DIV4	Divide by 4
0x3	DIV8	Divide by 8
0x4	DIV16	Divide by 16
0x5	DIV32	Divide by 32
0x6	DIV64	Divide by 64
0x7	DIV128	Divide by 128

15.8.8 AHB Mask

Name: AHBMASK

Offset: 0x14

Reset: 0x0000007F

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	1	1	1	1	1	1

- **Bits 31:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 5 – DMAC: DMAC AHB Clock Mask**
  0: The AHB clock for the DMAC is stopped.
  1: The AHB clock for the DMAC is enabled.

- **Bit 4 – NVMCTRL: NVMCTRL AHB Clock Mask**
  0: The AHB clock for the NVMCTRL is stopped.
  1: The AHB clock for the NVMCTRL is enabled.

- **Bit 3 – DSU: DSU AHB Clock Mask**
  0: The AHB clock for the DSU is stopped.
  1: The AHB clock for the DSU is enabled.

- **Bit 2 – HPB2: HPB2 AHB Clock Mask**
  0: The AHB clock for the HPB2 is stopped.

1: The AHB clock for the HPB2 is enabled.

- **Bit 1 – HPB1: HPB1 AHB Clock Mask**
  0: The AHB clock for the HPB1 is stopped.
  1: The AHB clock for the HPB1 is enabled.
- **Bit 0 – HPB0: HPB0 AHB Clock Mask**
  0: The AHB clock for the HPB0 is stopped.
  1: The AHB clock for the HPB0 is enabled.

15.8.9 APBA Mask

Name: APBAMASK

Offset: 0x18

Reset: 0x0000007F

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		EIC	RTC	WDT	GCLK	SYSCTRL	PM	PAC0
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	1	1	1	1	1	1

- **Bits 31:7 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 6 – EIC: EIC APB Clock Enable**
  0: The APBA clock for the EIC is stopped.
  1: The APBA clock for the EIC is enabled.
- **Bit 5 – RTC: RTC APB Clock Enable**
  0: The APBA clock for the RTC is stopped.
  1: The APBA clock for the RTC is enabled.
- **Bit 4 – WDT: WDT APB Clock Enable**
  0: The APBA clock for the WDT is stopped.
  1: The APBA clock for the WDT is enabled.
- **Bit 3 – GCLK: GCLK APB Clock Enable**
  0: The APBA clock for the GCLK is stopped.

1: The APBA clock for the GCLK is enabled.

- **Bit 2 – SYSCTRL: SYSCTRL APB Clock Enable**
  0: The APBA clock for the SYSCTRL is stopped.
  1: The APBA clock for the SYSCTRL is enabled.
- **Bit 1 – PM: PM APB Clock Enable**
  0: The APBA clock for the PM is stopped.
  1: The APBA clock for the PM is enabled.
- **Bit 0 – PAC0: PAC0 APB Clock Enable**
  0: The APBA clock for the PAC0 is stopped.
  1: The APBA clock for the PAC0 is enabled.

15.8.10 APBB Mask

Name: APBBMASK

Offset: 0x1C

Reset: 0x0000007F

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	1	1	1	1	1

- **Bits 31:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 4 – DMAC: DMAC APB Clock Enable**
  0: The APBB clock for the DMAC is stopped.
  1: The APBB clock for the DMAC is enabled.

- **Bit 3 – PORT: PORT APB Clock Enable**
  0: The APBB clock for the PORT is stopped.
  1: The APBB clock for the PORT is enabled.

- **Bit 2 – NVMCTRL: NVMCTRL APB Clock Enable**
  0: The APBB clock for the NVMCTRL is stopped.
  1: The APBB clock for the NVMCTRL is enabled.

- **Bit 1 – DSU: DSU APB Clock Enable**
  0: The APBB clock for the DSU is stopped.

1: The APBB clock for the DSU is enabled.

- **Bit 0 – PAC1: PAC1 APB Clock Enable**
  0: The APBB clock for the PAC1 is stopped.
  1: The APBB clock for the PAC1 is enabled.

15.8.11 APBC Mask

Name: APBCMASK

Offset: 0x20

Reset: 0x00000100

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
								ADC
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
	TC2	TC1			SERCOM1	SERCOM0	EVSYS	PAC2
Access	R/W	R/W	R	R/	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 8 – ADC: ADC APB Clock Enable**
  0: The APBC clock for the ADC is stopped.
  1: The APBC clock for the ADC is enabled.

- **Bit 7 – TC2: TC2 APB Clock Enable**

- **Bit 6 – TC1: TC1 APB Clock Enable**

- **Bits 5:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – SERCOM1: SERCOM1 APB Clock Enable**
  0: The APBC clock for the SERCOM1 is stopped.
  1: The APBC clock for the SERCOM1 is enabled.
- **Bit 2 – SERCOM0: SERCOM0 APB Clock Enable**
  0: The APBC clock for the SERCOM0 is stopped.
  1: The APBC clock for the SERCOM0 is enabled.
- **Bit 1 – EVSYS: EVSYS APB Clock Enable**
  0: The APBC clock for the EVSYS is stopped.
  1: The APBC clock for the EVSYS is enabled.
- **Bit 0 – PAC2: PAC2 APB Clock Enable**
  0: The APBC clock for the PAC2 is stopped.
  1: The APBC clock for the PAC2 is enabled.

15.8.12 Interrupt Enable Clear

Name: INTENCLR

Offset: 0x34

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
							CFD	CKRDY
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (INTENSET) register.

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – CFD: Clock Failure Detector Interrupt Enable**
  0: The Clock Failure Detector interrupt is disabled.
  1: The Clock Failure Detector interrupt is enabled and an interrupt request will be generated when the Clock Failure Detector Interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Clock Failure Detector Interrupt Enable bit and the corresponding interrupt request.

- **Bit 0 – CKRDY: Clock Ready Interrupt Enable**
  0: The Clock Ready interrupt is disabled.
  1: The Clock Ready interrupt is enabled and will generate an interrupt request when the Clock Ready Interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Clock Ready Interrupt Enable bit and the corresponding interrupt request.

15.8.13 Interrupt Enable Set

Name: INTENSET

Offset: 0x35

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
							CFD	CKRDY
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – CFD: Clock Failure Detector Interrupt Enable**
  0: The Clock Failure Detector interrupt is disabled.
  1: The Clock Failure Detector interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Clock Failure Detector Interrupt Enable bit and enable the Clock Failure Detector interrupt.

- **Bit 0 – CKRDY: Clock Ready Interrupt Enable**
  0: The Clock Ready interrupt is disabled.
  1: The Clock Ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Clock Ready Interrupt Enable bit and enable the Clock Ready interrupt.

15.8.14 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x36

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
							CFD	CKRDY
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – CFD: Clock Failure Detector**
  This flag is cleared by writing a one to the flag.

  This flag is set on the next cycle after a clock failure detector occurs and will generate an interrupt request if INTENCLR/SET.CFD is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the Clock Failure Detector Interrupt flag.

- **Bit 0 – CKRDY: Clock Ready**
  This flag is cleared by writing a one to the flag.

  This flag is set when the synchronous CPU and APBx clocks have frequencies as indicated in the CPUSEL and APBxSEL registers, and will generate an interrupt if INTENCLR/SET.CKRDY is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the Clock Ready Interrupt flag.

15.8.15 Reset Cause

Name: RCAUSE

Offset: 0x38

Reset: 0x01

Access: Read-Only

Property: -

Bit	7	6	5	4	3	2	1	0
		SYST	WDT	EXT		BOD33	BOD12	POR
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bit 6 – SYST: System Reset Request**
  This bit is set if a system reset request has been performed. Refer to the Cortex processor documentation for more details.
- **Bit 5 – WDT: Watchdog Reset**
  This flag is set if a Watchdog Timer reset occurs.
- **Bit 4 – EXT: External Reset**
  This flag is set if an external reset occurs.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bit 2 – BOD33: Brown Out 33 Detector Reset**
  This flag is set if a BOD33 reset occurs.
- **Bit 1 – BOD12: Brown Out 12 Detector Reset**
  This flag is set if a BOD12 reset occurs.
- **Bit 0 – POR: Power On Reset**
  This flag is set if a POR occurs.

16. SYSCTRL – System Controller

16.1 Overview

The System Controller (SYSCTRL) provides a user interface to the clock sources, brown out detectors, on-chip voltage regulator and voltage reference of the device.

Through the interface registers, it is possible to enable, disable, calibrate and monitor the SYSCTRL sub-peripherals.

All sub-peripheral statuses are collected in the Power and Clocks Status register (PCLKSR - refer to PCLKSR). They can additionally trigger interrupts upon status changes via the INTENSET (INTENSET), INTENCLR (INTENCLR) and INTFLAG (INTFLAG) registers.

Additionally, BOD33 and BOD12 interrupts can be used to wake up the device from standby mode upon a programmed brown-out detection.

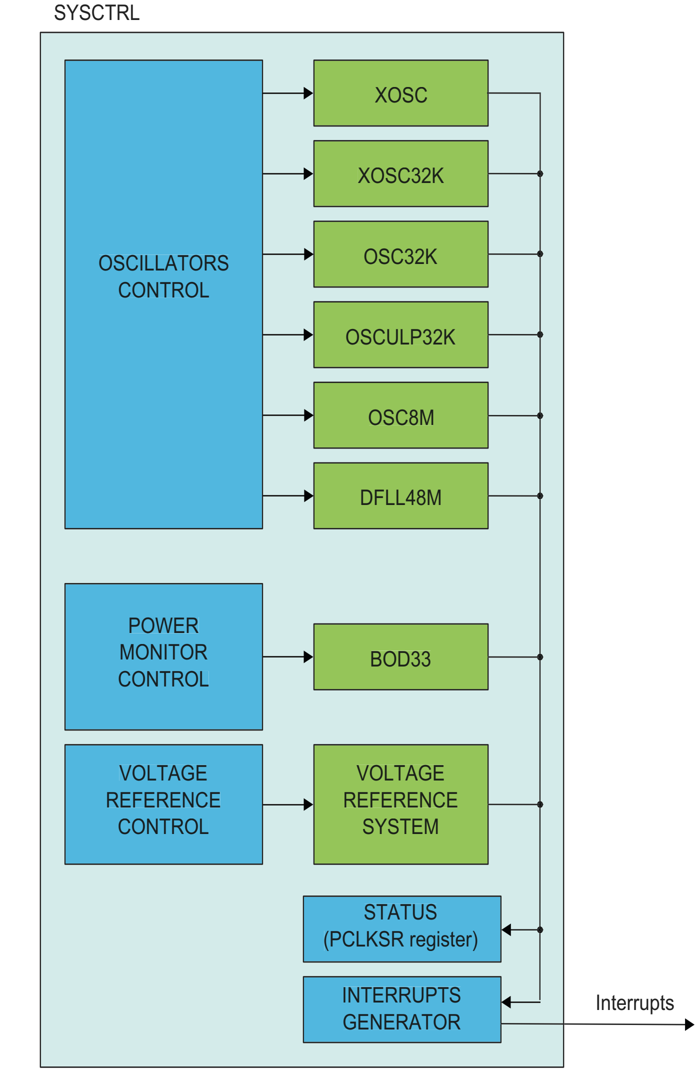
16.2 Features

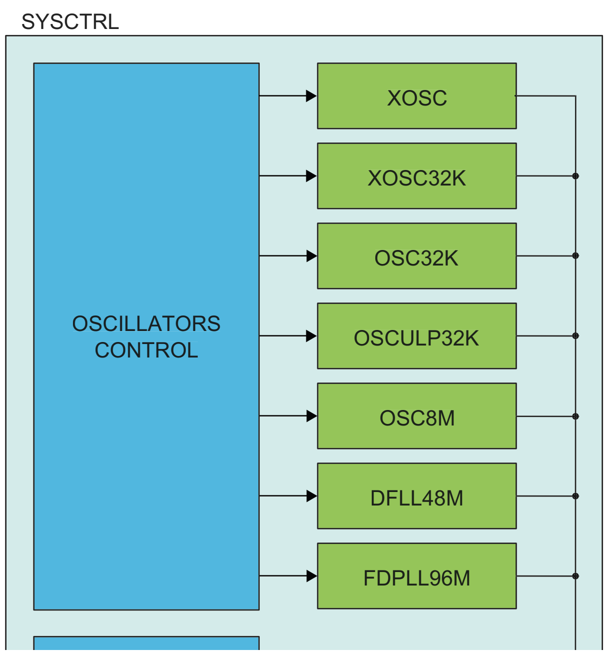
- 0.4-32MHz Crystal Oscillator (XOSC)
  - Tunable gain control
  - Programmable start-up time
  - Crystal or external input clock on XIN I/O
- 32.768kHz Crystal Oscillator (XOSC32K)
  - Automatic or manual gain control
  - Programmable start-up time
  - Crystal or external input clock on XIN32 I/O
- 32.768kHz High Accuracy Internal Oscillator (OSC32K)
  - Frequency fine tuning
  - Programmable start-up time
- 32.768kHz Ultra Low Power Internal Oscillator (OSCULP32K)
  - Ultra low power, always-on oscillator
  - Frequency fine tuning
  - Calibration value loaded from Flash Factory Calibration at reset
- 8MHz Internal Oscillator (OSC8M)
  - Fast startup
  - Output frequency fine tuning
  - 4/2/1MHz divided output frequencies available
  - Calibration value loaded from Flash Factory Calibration at reset
- Digital Frequency Locked Loop (DFLL48M)
  - Internal oscillator with no external components
  - 48MHz output frequency
  - Operates standalone as a high-frequency programmable oscillator in open loop mode
  - Operates as an accurate frequency multiplier against a known frequency in closed loop mode
- Fractional Digital Phase Locked Loop (FDPLL96M)
  - 48MHz to 96MHz output clock frequency
  - 32KHz to 2MHz input reference clock frequency range
  - Three possible sources for the reference clock
  - Adjustable proportional integral controller
  - Fractional part used to achieve 1/16th of reference clock step
- 3.3V Brown-Out Detector (BOD33)
  - Programmable threshold
  - Threshold value loaded from Flash User Calibration at startup
  - Triggers resets or interrupts
  - Operating modes:
    - Continuous mode

    - Sampled mode for low power applications (programmable refresh frequency)
  - Hysteresis
- 1.2V Brown-Out Detector (BOD12)
  - Programmable threshold
  - Threshold value loaded from Flash User Calibration at startup
  - Triggers resets or interrupts
  - Operating modes:
    - Continuous mode
    - Sampled mode for low power applications (programmable refresh frequency)
  - Hysteresis
- Voltage Reference System (VREF)
  - Bandgap voltage generator with programmable calibration value
  - Temperature sensor
  - Bandgap calibration value loaded from Flash Factory Calibration at startup
- Voltage Regulator System (VREG)
  - Trimable core supply voltage level
  - Voltage regulator trim value loaded from Flash Factory Calibration at startup

16.3 Block Diagram

Figure 16-1. SYSCTRL Block Diagram





16.4 Signal Description

Signal Name	Types	Description
XIN	Analog Input	Multipurpose Crystal Oscillator or external clock generator input
XOUT	Analog Output	External Multipurpose Crystal Oscillator output
XIN32	Analog Input	32kHz Crystal Oscillator or external clock generator input
XOUT32	Analog Output	32kHz Crystal Oscillator output

The I/O lines are automatically selected when XOSC or XOSC32K are enabled.

16.5 Product Dependencies

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

16.5.1 I/O Lines

I/O lines are configured by SYSCTRL when either XOSC or XOSC32K are enabled, and need no user configuration.

16.5.2 Power Management

The SYSCTRL can continue to operate in any sleep mode where the selected source clock is running. The SYSCTRL interrupts can be used to wake up the device from sleep modes. The events can trigger other operations in the system without exiting sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

16.5.3 Clocks

The SYSCTRL gathers controls for all device oscillators and provides clock sources to the Generic Clock Controller (GCLK). The available clock sources are: XOSC, XOSC32K, OSC32K, OSCULP32K, OSC8M, and DFLL48M and FDPLL96M.

The SYSCTRL bus clock (CLK_SYSCTRL_APB) can be enabled and disabled in the Power Manager, and the default state of CLK_SYSCTRL_APB can be found in the Peripheral Clock Masking section in the "PM – Power Manager" on page 107.

The clock used by BOD33 and BOD12 in sampled mode is asynchronous to the user interface clock (CLK_SYSCTRL_APB). Likewise, the DFLL48M control logic uses the DFLL oscillator output, which is also asynchronous to the user interface clock (CLK_SYSCTRL_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 158 for further details.

The FDPLL96M reference clock (CLK_FDPLL96M_REF) can be selected among three different clock sources:

Table 16-1. Oscillators and Generic Clock inputs for FDPLL96M clock sources connections

Oscillator / Generic Clock	CLK_FDPLL96M_REF Reference Clock source connection
XOSC32K	CLK_DPLL_REF0
XOSC	CLK_DPLL_REF1
GCLK (FDPLL96M)	GCLK_DPLL

The selected clock must be configured and enabled before using the FDPLL96M. If the GCLK is selected as reference clock, it must be configured and enabled in the Generic Clock Controller before using the FDPLL96M. Refer to "GCLK – Generic Clock Controller" on page 85 for details.If the GCLK_DPLL is selected as the source for the

CLK_FDPLL96M_REF, care must be taken to make sure the source for this GCLK is within the valid frequency range for the FDPLL96M.

The XOSC source can be divided inside the FDPLL96M. The user must make sure that the programmable clock divider and XOSC frequency provides a valid CLK_FDPLL96M_REF clock frequency that meets the FDPLL96M input frequency range.

The FDPLL96M requires a 32kHz clock from the GCLK when the FDPLL96M internal lock timer is used. This clock must be configured and enabled in the Generic Clock Controller before using the FDPLL96M. Refer to "GCLK – Generic Clock Controller" on page 85 for details.

Table 16-2. Generic Clock Input for FDPLL96M

Generic Clock	FDPLL96M
FDPLL96M 32kHz clock	GCLK_DPLL_32K for internal lock timer
FDPLL96M	GCLK_DPLL for CLK_FDPLL96M_REF

16.5.4 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the SYSCTRL interrupts requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

16.5.5 Debug Operation

When the CPU is halted in debug mode, the SYSCTRL continues normal operation. If the SYSCTRL 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.

If debugger cold-plugging is detected by the system, BOD33 reset will be masked. The BOD reset keeps running under hot-plugging.

16.5.6 Register Access Protection

All registers with write-access are optionally write-protected by the peripheral access controller (PAC), except the following registers:

- Interrupt Flag Status and Clear register (INTFLAG - refer to INTFLAG)

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode, all write-protection is automatically disabled.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

16.5.7 Analog Connections

When used, the 32.768kHz crystal must be connected between the XIN32 and XOUT32 pins, and the 0.4-32MHz crystal must be connected between the XIN and XOUT pins, along with any required load capacitors. For details on recommended oscillator characteristics and capacitor load, refer to the "Electrical Characteristics" on page 648 for details.

16.6 Functional Description

16.6.1 Principle of Operation

XOSC, XOSC32K, OSC32K, OSCULP32K, OSC8M, DFLL48M, FDPLL96M, BOD33, BOD12, VREG and VREF are configured via SYSCTRL control registers. Through this interface, the sub-peripherals are enabled, disabled or have their calibration values updated.

The Power and Clocks Status register gathers different status signals coming from the sub-peripherals controlled by the SYSCTRL. The status signals can be used to generate system interrupts, and in some cases wake up the system from standby mode, provided the corresponding interrupt is enabled.

The oscillator must be enabled to run. The oscillator is enabled by writing a one to the ENABLE bit in the respective oscillator control register, and disabled by writing a zero to the oscillator control register. In idle mode, the default operation of the oscillator is to run only when requested by a peripheral. In standby mode, the default operation of the oscillator is to stop. This behavior can be changed by the user, see below for details.

The behavior of the oscillators in the different sleep modes is shown in Table 16-3 on page 146

Table 16-3. Behavior of the Oscillators

Oscillator	Idle 0, 1, 2	Standby
XOSC	Run on request	Stop
XOSC32K	Run on request	Stop
OSC32K	Run on request	Stop
OSCULP32K	Run	Run
OSC8M	Run on request	Stop
DFLL48M	Run on request	Stop
FDPLL96M	Run on request	Stop

To force an oscillator to always run in idle mode, and not only when requested by a peripheral, the oscillator ONDEMAND bit must be written to zero. The default value of this bit is one, and thus the default operation in idle mode is to run only when requested by a peripheral.

To force the oscillator to run in standby mode except for DFLL and DPLL, the RUNSTDBY bit must be written to one. The oscillator will then run in standby mode when requested by a peripheral (ONDEMAND is one). To force an oscillator to always run in standby mode, and not only when requested by a peripheral, the ONDEMAND bit must be written to zero and RUNSTDBY must be written to one.

Table 16-4 on page 146 shows the behavior in the different sleep modes, depending on the settings of ONDEMAND and RUNSTDBY.

Table 16-4. Behavior in the different sleep modes

Sleep mode	ONDEMAND	RUNSTDBY	Behavior
Idle 0, 1, 2	0	X	Run
Idle 0, 1, 2	1	X	Run when requested by a peripheral
Standby	0	0	Stop
Standby	0	1	Run
Standby	1	0	Stop
Standby	1	1	Run when requested by a peripheral

Note that this does not apply to the OSCULP32K oscillator, which is always running and cannot be disabled.

16.6.2 External Multipurpose Crystal Oscillator (XOSC) Operation

The XOSC can operate in two different modes:

- External clock, with an external clock signal connected to the XIN pin
- Crystal oscillator, with an external 0.4-32MHz crystal

The XOSC can be used as a clock source for generic clock generators, as described in the "GCLK – Generic Clock Controller" on page 85.

At reset, the XOSC is disabled, and the XIN/XOUT pins can be used as General Purpose I/O (GPIO) pins or by other peripherals in the system. When XOSC is enabled, the operating mode determines the GPIO usage. When in crystal oscillator mode, the XIN and XOUT pins are controlled by the SYSCTRL, and GPIO functions are overridden on both pins. When in external clock mode, only the XIN pin will be overridden and controlled by the SYSCTRL, while the XOUT pin can still be used as a GPIO pin.

The XOSC is enabled by writing a one to the Enable bit in the External Multipurpose Crystal Oscillator Control register (XOSC.ENABLE). To enable the XOSC as a crystal oscillator, a one must be written to the XTAL Enable bit (XOSC.XTALEN). If XOSC.XTALEN is zero, external clock input will be enabled.

When in crystal oscillator mode (XOSC.XTALEN is one), the External Multipurpose Crystal Oscillator Gain (XOSC.GAIN) must be set to match the external crystal oscillator frequency. If the External Multipurpose Crystal Oscillator Automatic Amplitude Gain Control (XOSC.AMPGC) is one, the oscillator amplitude will be automatically adjusted, and in most cases result in a lower power consumption.

The XOSC will behave differently in different sleep modes based on the settings of XOSC.RUNSTDBY, XOSC.ONDEMAND and XOSC.ENABLE:

XOSC.RUNSTDBY	XOSC.ONDEMAND	XOSC.ENABLE	Sleep Behavior
-	-	0	Disabled
0	0	1	Always run in IDLE sleep modes. Disabled in STANDBY sleep mode.
0	1	1	Only run in IDLE sleep modes if requested by a peripheral. Disabled in STANDBY sleep mode.
1	0	1	Always run in IDLE and STANDBY sleep modes.
1	1	1	Only run in IDLE or STANDBY sleep modes if requested by a peripheral.

After a hard reset, or when waking up from a sleep mode where the XOSC was disabled, the XOSC will need a certain amount of time to stabilize on the correct frequency. This start-up time can be configured by changing the Oscillator Start-Up Time bit group (XOSC.STARTUP) in the External Multipurpose Crystal Oscillator Control register. During the start-up time, the oscillator output is masked to ensure that no unstable clock propagates to the digital logic. The External Multipurpose Crystal Oscillator Ready bit in the Power and Clock Status register (PCLKSR.XOSCRDY) is set when the user-selected startup time is over. An interrupt is generated on a zero-to-one transition on PCLKSR.XOSCRDY if the External Multipurpose Crystal Oscillator Ready bit in the Interrupt Enable Set register (INTENSET.XOSCRDY) is set.

Note: Do not enter standby mode when an oscillator is in startup:
Wait for the OSCxRDY bit in SYSCTRL.PCLKSR register to be set before going into standby mode.

16.6.3 32kHz External Crystal Oscillator (XOSC32K) Operation

The XOSC32K can operate in two different modes:

- External clock, with an external clock signal connected to XIN32
- Crystal oscillator, with an external 32.768kHz crystal connected between XIN32 and XOUT32

The XOSC32K can be used as a source for generic clock generators, as described in the "GCLK – Generic Clock Controller" on page 85.

At power-on reset (POR) the XOSC32K is disabled, and the XIN32/XOUT32 pins can be used as General Purpose I/O (GPIO) pins or by other peripherals in the system. When XOSC32K is enabled, the operating mode determines the GPIO usage. When in crystal oscillator mode, XIN32 and XOUT32 are controlled by the SYSCTRL, and GPIO functions are overridden on both pins. When in external clock mode, only the XIN32 pin will be overridden and controlled by the SYSCTRL, while the XOUT32 pin can still be used as a GPIO pin.

The external clock or crystal oscillator is enabled by writing a one to the Enable bit (XOSC32K.ENABLE) in the 32kHz External Crystal Oscillator Control register. To enable the XOSC32K as a crystal oscillator, a one must be written to the XTAL Enable bit (XOSC32K.XTALEN). If XOSC32K.XTALEN is zero, external clock input will be enabled.

The oscillator is disabled by writing a zero to the Enable bit (XOSC32K.ENABLE) in the 32kHz External Crystal Oscillator Control register while keeping the other bits unchanged. Writing to the XOSC32K.ENABLE bit while writing to other bits may result in unpredictable behavior. The oscillator remains enabled in all sleep modes if it has been enabled beforehand. The start-up time of the 32kHz External Crystal Oscillator is selected by writing to the Oscillator Start-Up Time bit group (XOSC32K.STARTUP) in the in the 32kHz External Crystal Oscillator Control register. The SYSCTRL masks the oscillator output during the start-up time to ensure that no unstable clock propagates to the digital logic. The 32kHz External Crystal Oscillator Ready bit (PCLKSR.XOSC32KRDY) in the Power and Clock Status register is set when the user-selected startup time is over. An interrupt is generated on a zero-to-one transition of PCLKSR.XOSC32KRDY if the 32kHz External Crystal Oscillator Ready bit (INTENSET.XOSC32KRDY) in the Interrupt Enable Set Register is set.

As a crystal oscillator usually requires a very long start-up time (up to one second), the 32kHz External Crystal Oscillator will keep running across resets, except for power-on reset (POR).

XOSC32K can provide two clock outputs when connected to a crystal. The XOSC32K has a 32.768kHz output enabled by writing a one to the 32kHz External Crystal Oscillator 32kHz Output Enable bit (XOSC32K.EN32K) in the 32kHz External Crystal Oscillator Control register. The XOSC32K also has a 1.024kHz clock output enabled by writing a one to the 32kHz External Crystal Oscillator 1kHz Output Enable bit (XOSC32K.EN1K) in the External 32kHz Crystal Oscillator Control register. XOSC32K.EN32K and XOSC32K.EN1K are only usable when XIN32 is connected to a crystal, and not when an external digital clock is applied on XIN32.

Note: Do not enter standby mode when an oscillator is in startup:
Wait for the OSCxRDY bit in SYSCTRL.PCLKSR register to be set before going into standby mode.

16.6.4 32kHz Internal Oscillator (OSC32K) Operation

The OSC32K provides a tunable, low-speed and low-power clock source.

The OSC32K can be used as a source for the generic clock generators, as described in the "GCLK – Generic Clock Controller" on page 85.

The OSC32K is disabled by default. The OSC32K is enabled by writing a one to the 32kHz Internal Oscillator Enable bit (OSC32K.ENABLE) in the 32kHz Internal Oscillator Control register. It is disabled by writing a zero to OSC32K.ENABLE. The OSC32K has a 32.768kHz output enabled by writing a one to the 32kHz Internal Oscillator 32kHz Output Enable bit (OSC32K.EN32K). The OSC32K also has a 1.024kHz clock output enabled by writing a one to the 32kHz Internal Oscillator 1kHz Output Enable bit (OSC32K.EN1K).

The frequency of the OSC32K oscillator is controlled by the value in the 32kHz Internal Oscillator Calibration bits (OSC32K.CALIB) in the 32kHz Internal Oscillator Control register. The OSC32K.CALIB value must be written by the user. Flash Factory Calibration values are stored in the NVM Software Calibration Area (refer to "NVM Software Calibration Row Mapping" on page 21). When writing to the Calibration bits, the user must wait for the PCLKSR.OSC32KRDY bit to go high before the value is committed to the oscillator.

16.6.5 32kHz Ultra Low Power Internal Oscillator (OSCULP32K) Operation

The OSCULP32K provides a tunable, low-speed and ultra-low-power clock source. The OSCULP32K is factory-calibrated under typical voltage and temperature conditions. The OSCULP32K should be preferred to the OSC32K whenever the power requirements are prevalent over frequency stability and accuracy.

The OSCULP32K can be used as a source for the generic clock generators, as described in the "GCLK – Generic Clock Controller" on page 85.

The OSCULP32K is enabled by default after a power-on reset (POR) and will always run except during POR. The OSCULP32K has a 32.768kHz output and a 1.024kHz output that are always running.

The frequency of the OSCULP32K oscillator is controlled by the value in the 32kHz Ultra Low Power Internal Oscillator Calibration bits (OSCULP32K.CALIB) in the 32kHz Ultra Low Power Internal Oscillator Control register. OSCULP32K.CALIB is automatically loaded from Flash Factory Calibration during startup, and is used to compensate for process variation, as described in the "Electrical Characteristics" on page 648. The calibration value can be overridden by the user by writing to OSCULP32K.CALIB.

16.6.6 8MHz Internal Oscillator (OSC8M) Operation

OSC8M is an internal oscillator operating in open-loop mode and generating an 8MHz frequency. The OSC8M is factory-calibrated under typical voltage and temperature conditions.

OSC8M is the default clock source that is used after a power-on reset (POR). The OSC8M can be used as a source for the generic clock generators, as described in the "GCLK – Generic Clock Controller" on page 85, as well as function as the backup clock if a main clock failure is detected.

In order to enable OSC8M, the Oscillator Enable bit in the OSC8M Control register (OSC8M.ENABLE) must be written to one. OSC8M will not be enabled until OSC8M.ENABLE is set. In order to disable OSC8M, OSC8M.ENABLE must be written to zero. OSC8M will not be disabled until OSC8M is cleared.

The frequency of the OSC8M oscillator is controlled by the value in the calibration bits (OSC8M.CALIB) in the OSC8M Control register. CALIB is automatically loaded from Flash Factory Calibration during startup, and is used to compensate for process variation, as described in the "Electrical Characteristics" on page 648.

The user can control the oscillation frequency by writing to the Frequency Range (FRANGE) and Calibration (CALIB) bit groups in the 8MHz RC Oscillator Control register (OSC8M). It is not recommended to update the FRANGE and CALIB bits when the OSC8M is enabled. As this is in open-loop mode, the frequency will be voltage, temperature and process dependent. Refer to the "Electrical Characteristics" on page 648 for details.

OSC8M is automatically switched off in certain sleep modes to reduce power consumption, as described in the "PM – Power Manager" on page 107.

16.6.7 Digital Frequency Locked Loop (DFLL48M) Operation

The DFLL48M can operate in both open-loop mode and closed-loop mode. In closed-loop mode, a low-frequency clock with high accuracy can be used as the reference clock to get high accuracy on the output clock (CLK_DFLL48M).

The DFLL48M can be used as a source for the generic clock generators, as described in the "GCLK – Generic Clock Controller" on page 85.

16.6.7.1 Basic Operation

Open-Loop Operation

After any reset, the open-loop mode is selected. When operating in open-loop mode, the output frequency of the DFLL48M will be determined by the values written to the DFLL Coarse Value bit group and the DFLL Fine Value bit group (DFLLVAL.COARSE and DFLLVAL.FINE) in the DFLL Value register.

Using "DFLL48M COARSE CAL" value from Table in DFLL.COARSE helps to output a frequency close to 48 MHz.

It is possible to change the values of DFLLVAL.COARSE and DFLLVAL.FINE and thereby the output frequency of the DFLL48M output clock, CLK_DFLL48M, while the DFLL48M is enabled and in use. CLK_DFLL48M is ready to be used when PCLKSR.DFLLRDY is set after enabling the DFLL48M.

Closed-Loop Operation

In closed-loop operation, the output frequency is continuously regulated against a reference clock. Once the multiplication factor is set, the oscillator fine tuning is automatically adjusted. The DFLL48M must be correctly configured before closed-loop operation can be enabled. After enabling the DFLL48M, it must be configured in the following way:

1. Enable and select a reference clock (CLK_DFLL48M_REF). CLK_DFLL48M_REF is Generic Clock Channel 0 (DFLL48M_Reference). Refer to "GCLK – Generic Clock Controller" on page 85 for details.
2. Select the maximum step size allowed in finding the Coarse and Fine values by writing the appropriate values to the DFLL Coarse Maximum Step and DFLL Fine Maximum Step bit groups (DFLLMUL.CSTEP and DFLL-MUL.FSTEP) in the DFLL Multiplier register. A small step size will ensure low overshoot on the output frequency, but will typically result in longer lock times. A high value might give a large overshoot, but will typically provide faster locking. DFLLMUL.CSTEP and DFLLMUL.FSTEP should not be higher than 50% of the maximum value of DFLLVAL.COARSE and DFLLVAL.FINE, respectively.
3. Select the multiplication factor in the DFLL Multiply Factor bit group (DFLLMUL.MUL) in the DFLL Multiplier register. Care must be taken when choosing DFLLMUL.MUL so that the output frequency does not exceed the maximum frequency of the DFLL. If the target frequency is below the minimum frequency of the DFLL48M, the output frequency will be equal to the DFLL minimum frequency.
4. Start the closed loop mode by writing a one to the DFLL Mode Selection bit (DFLLCTRL.MODE) in the DFLL Control register.

The frequency of CLK_DFLL48M ($F_{clkdfll48m}$) is given by:

$$F_{clkdfll48m} = DFLLMUL \cdot MUL \times F_{clkdfll48mref}$$

where $F_{clkdfll48mref}$ is the frequency of the reference clock (CLK_DFLL48M_REF). DFLLVAL.COARSE and DFLLVAL.FINE are read-only in closed-loop mode, and are controlled by the frequency tuner to meet user specified frequency. In closed-loop mode, the value in DFLLVAL.COARSE is used by the frequency tuner as a starting point for Coarse. Writing DFLLVAL.COARSE to a value close to the final value before entering closed-loop mode will reduce the time needed to get a lock on Coarse.

Using "DFLL48M COARSE CAL" from Table for DFLL.COARSE will start DFLL with a frequency close to 48 MHz.

Following Software sequence should be followed while using the same.

1. load "DFLL48M COARSE CAL" from Table in DFLL.COARSE register
2. Set DFLLCTRL.BPLCKC bit
3. Start DFLL close loop

This procedure will reduce DFLL Lock time to DFLL Fine lock time.

Frequency Locking

The locking of the frequency in closed-loop mode is divided into two stages. In the first, coarse stage, the control logic quickly finds the correct value for DFLLVAL.COARSE and sets the output frequency to a value close to the correct frequency. On coarse lock, the DFLL Locked on Coarse Value bit (PCLKSR.DFLLLOCKC) in the Power and Clocks Status register will be set.

In the second, fine stage, the control logic tunes the value in DFLLVAL.FINE so that the output frequency is very close to the desired frequency. On fine lock, the DFLL Locked on Fine Value bit (PCLKSR.DFLLLOCKF) in the Power and Clocks Status register will be set.

Interrupts are generated by both PCLKSR.DFLLLOCKC and PCLKSR.DFLLLOCKF if INTENSET.DFLLOCKC or INTENSET.DFLLOCKF are written to one.

CLK_DFLL48M is ready to be used when the DFLL Ready bit (PCLKSR.DFLLRDY) in the Power and Clocks Status register is set, but the accuracy of the output frequency depends on which locks are set. For lock times, refer to the "Electrical Characteristics" on page 648.

Frequency Error Measurement

The ratio between CLK_DFLL48M_REF and CLK48M_DFLL is measured automatically when the DFLL48M is in closed-loop mode. The difference between this ratio and the value in DFLLMUL.MUL is stored in the DFLL

Multiplication Ratio Difference bit group(DFLLVAL.DIFF) in the DFLL Value register. The relative error on CLK_DFLL48M compared to the target frequency is calculated as follows:

$$ERROR = \frac{DIFF}{MUL}$$

Drift Compensation

If the Stable DFLL Frequency bit (DFLLCTRL.STABLE) in the DFLL Control register is zero, the frequency tuner will automatically compensate for drift in the CLK_DFLL48M without losing either of the locks. This means that DFLLVAL.FINE can change after every measurement of CLK_DFLL48M.

The DFLLVAL.FINE value overflows or underflows can occur in close loop mode when the clock source reference drifts or is unstable. This will set the DFLL Out Of Bounds bit (PCLKSR.DFLLOOB) in the Power and Clocks Status register.

To avoid this error, the reference clock in close loop mode must be stable, an external oscillator is recommended and internal oscillator forbidden. The better choice is to use an XOSC32K.

Reference Clock Stop Detection

If CLK_DFLL48M_REF stops or is running at a very low frequency (slower than CLK_DFLL48M/(2 * MUL_{MAX})), the DFLL Reference Clock Stopped bit (PCLKSR.DFLLRCS) in the Power and Clocks Status register will be set. Detecting a stopped reference clock can take a long time, on the order of 2^{17} CLK_DFLL48M cycles. When the reference clock is stopped, the DFLL48M will operate as if in open-loop mode. Closed-loop mode operation will automatically resume if the CLK_DFLL48M_REF is restarted. An interrupt is generated on a zero-to-one transition on PCLKSR.DFLLRCS if the DFLL Reference Clock Stopped bit (INTENSET.DFLLRCS) in the Interrupt Enable Set register is set.

16.6.7.2 Additional Features

Dealing with Delay in the DFLL in Closed-Loop Mode

The time from selecting a new CLK_DFLL48M frequency until this frequency is output by the DFLL48M can be up to several microseconds. If the value in DFLLMUL.MUL is small, this can lead to instability in the DFLL48M locking mechanism, which can prevent the DFLL48M from achieving locks. To avoid this, a chill cycle, during which the CLK_DFLL48M frequency is not measured, can be enabled. The chill cycle is enabled by default, but can be disabled by writing a one to the DFLL Chill Cycle Disable bit (DFLLCTRL.CCDIS) in the DFLL Control register. Enabling chill cycles might double the lock time.

Another solution to this problem consists of using less strict lock requirements. This is called Quick Lock (QL), which is also enabled by default, but it can be disabled by writing a one to the Quick Lock Disable bit (DFLLCTRL.QLDIS) in the DFLL Control register. The Quick Lock might lead to a larger spread in the output frequency than chill cycles, but the average output frequency is the same.

Wake from Sleep Modes

DFLL48M can optionally reset its lock bits when it is disabled. This is configured by the Lose Lock After Wake bit (DFLLCTRL.LLAW) in the DFLL Control register. If DFLLCTRL.LLAW is zero, the DFLL48M will be re-enabled and start running with the same configuration as before being disabled, even if the reference clock is not available. The locks will not be lost. When the reference clock has restarted, the Fine tracking will quickly compensate for any frequency drift during sleep if DFLLCTRL.STABLE is zero. If DFLLCTRL.LLAW is one when the DFLL is turned off, the DFLL48M will lose all its locks, and needs to regain these through the full lock sequence.

Accuracy

There are three main factors that determine the accuracy of $F_{clkdfll48m}$. These can be tuned to obtain maximum accuracy when fine lock is achieved.

- Fine resolution: The frequency step between two Fine values. This is relatively smaller for high output frequencies.

- Resolution of the measurement: If the resolution of the measured $F_{clkdfll48m}$ is low, i.e., the ratio between the CLK_DFLL48M frequency and the CLK_DFLL48M_REF frequency is small, then the DFLL48M might lock at a frequency that is lower than the targeted frequency. It is recommended to use a reference clock frequency of 32kHz or lower to avoid this issue for low target frequencies.
- The accuracy of the reference clock.

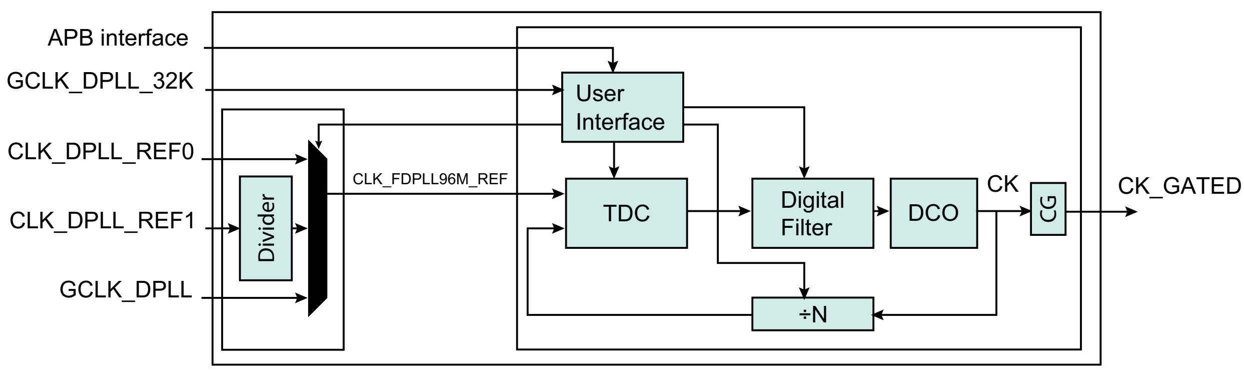
16.6.8 FDPLL96M – Fractional Digital Phase-Locked Loop Controller

16.6.8.1 Overview

The FDPLL96M controller allows flexible interface to the core digital function of the Digital Phase Locked Loop (DPLL). The FDPLL96M integrates a digital filter with a proportional integral controller, a Time-to-Digital Converter (TDC), a test mode controller, a Digitally Controlled Oscillator (DCO) and a PLL controller. It also provides a fractional multiplier of frequency N between the input and output frequency. The CLK_FDPLL96M_REF is the DPLL input clock reference. The selectable sources for the reference clock are CLK_DPLL_REF0, CLK_DPLL_REF1 and GCLK_DPLL. The path between CLK_DPLL_REF1 and input multiplexer integrates a clock divider. The output clock of the FDPLL96M is CK_GATED. The state of the CK_GATED clock only depends on the FDPLL96M internal control of the final clock gater CG. A valid 32khz clock is required (GCLK_DPLL_32K clock) when the FDPLL96M internal lock timer is used.

16.6.8.2 Block Diagram

Figure 16-2. FDPLL96M Block Diagram.



16.6.8.3 Principle of Operation

The task of the FDPLL96M is to maintain coherence between the input reference clock signal (CLK_FDPLL96M_REF) and the respective output frequency CK via phase comparison. The FDPLL96M supports three independent sources of clocks CLK_DPLL_REF0, CLK_DPLL_REF1 and GCLK_DPLL. When the FDPLL96M is enabled, the relationship between the reference clock (CLK_FDPLL96M_REF) frequency and the output clock (CK_GATED) frequency is defined below.

$$f_{ck\_gated} = f_{clk\_fdpll96m\_ref} \times \left(LDR + 1 + \frac{LDRFRAC}{16}\right)$$

Where LDR is the loop divider ratio integer part, LDRFRAC is the loop divider ratio fractional part, f_{ckrx} is the frequency of the selected reference clock and f_{ck} is the frequency of the FDPLL96M output clock. As previously stated a clock divider exist between CLK_DPLL_REF1 and CLK_FDPLL96M_REF. The frequency between the two clocks is defined below.

$$f_{clk\_fdpll96m\_ref} = f_{clk\_dpll\_ref1} \times \left(\frac{1}{2 \times (DIV + 1)}\right)$$

When the FDPLL96M is disabled, the output clock is reset. If the loop divider ratio fractional part (DPLLRATIO.LDRFRAC) field is reset, the FDPLL96M works in integer mode, otherwise the fractional mode is activated. It shall be noted that fractional part has a negative impact on the jitter of the FDPLL96M.

Example (integer mode only): assuming f_{ckr} = 32kHz and f_{ck} = 48MHz, the multiplication ratio is 1500. It means that LDR shall be set to 1499.

Example (fractional mode): assuming f_{ckr} = 32 kHz and f_{ck} = 48.006MHz, the multiplication ratio is 1500.1875 (1500 + 3/16). Thus LDR is set to 1499 and LDRFRAC to 3.

16.6.8.4 Initialization, Enabling, Disabling and Resetting

The FDPLL96M is enabled by writing a one to the Enable bit in the DPLL Control A register (DPLLCTRLA.ENABLE). The FDPLL96M is disabled by writing a zero to DPLLCTRLA.ENABLE. The frequency of the FDPLL96M output clock CK is stable when the module is enabled and when the DPLL Lock Status bit in the DPLL Status register (DPLLSTATUS.LOCK) bit is set. When DPLLCTRLB.LTIME is different from 0, a user defined lock time is used to validate the lock operation. In this case the lock time is constant. If DPLLCTRLB.LTIME is reset, the lock signal is linked with the status bit of the DPLL, the lock time vary depending on the filter selection and final target frequency.

When DPLLCTRLB.WUF is set, the wake up fast mode is activated. In that mode the clock gating cell is enabled at the end of the startup time. At that time, the final frequency is not stable as it is still in the acquisition period, but it allows to save several milliseconds. After first acquisition, DPLLCTRLB.LBYPASS indicates if the Lock signal is discarded from the control of the clock gater generating the output clock CK_GATED.

Table 16-5. CK_GATED behavior from startup to first edge detection.

WUF	LTIME	CK_GATED Behavior
0	0	Normal Mode: First Edge when lock is asserted
0	Not Equal To Zero	Lock Timer Timeout mode: First Edge when the timer downcounts to 0.
1	X	Wake Up Fast Mode: First Edge when CK is active (startup time)

Table 16-6. CK_GATED behavior after First Edge detection.

LBYPASS	CK_GATED Behavior
0	Normal Mode: the CK_GATED is turned off when lock signal is low.
1	Lock Bypass Mode: the CK_GATED is always running, lock is irrelevant.

Figure 16-3. CK and CK_GATED Output from FDPLL96M Off Mode to Running Mode

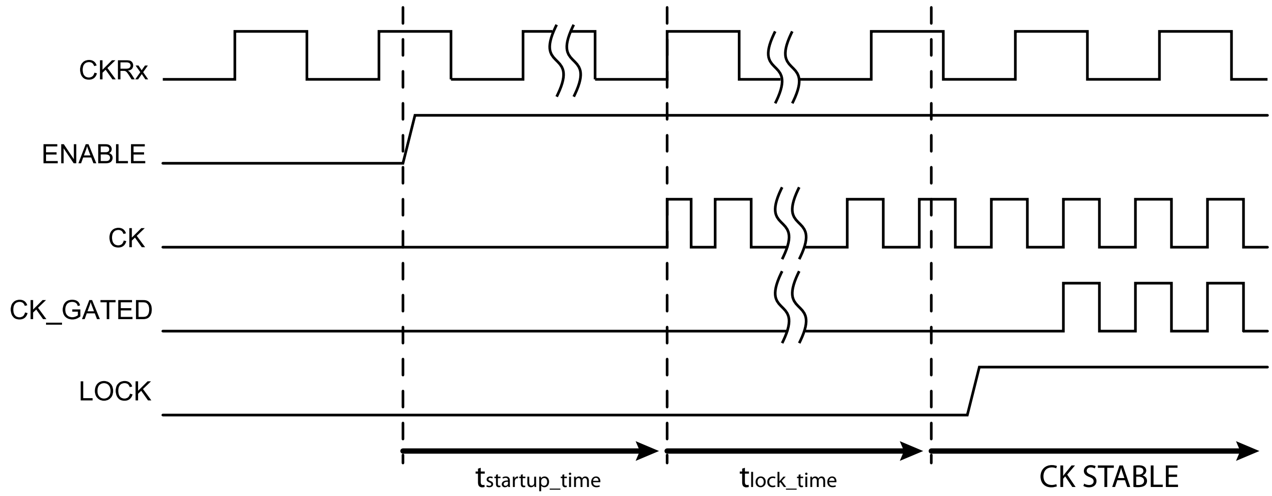


Figure 16-4. CK and CK_GATED Output from FDPLL96M Off Mode to Running Mode when Wake-Up Fast is Activated

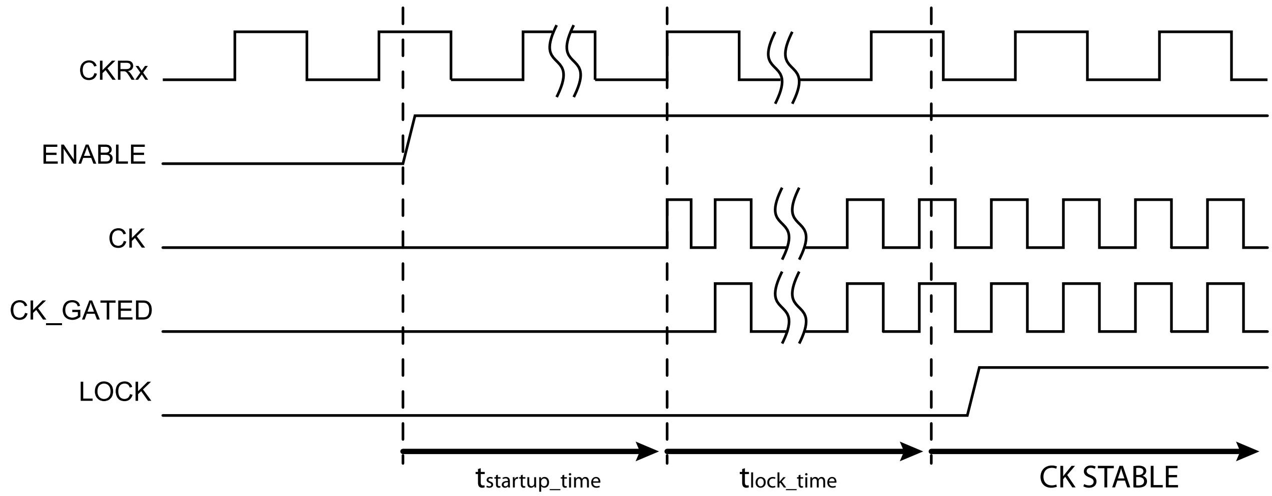
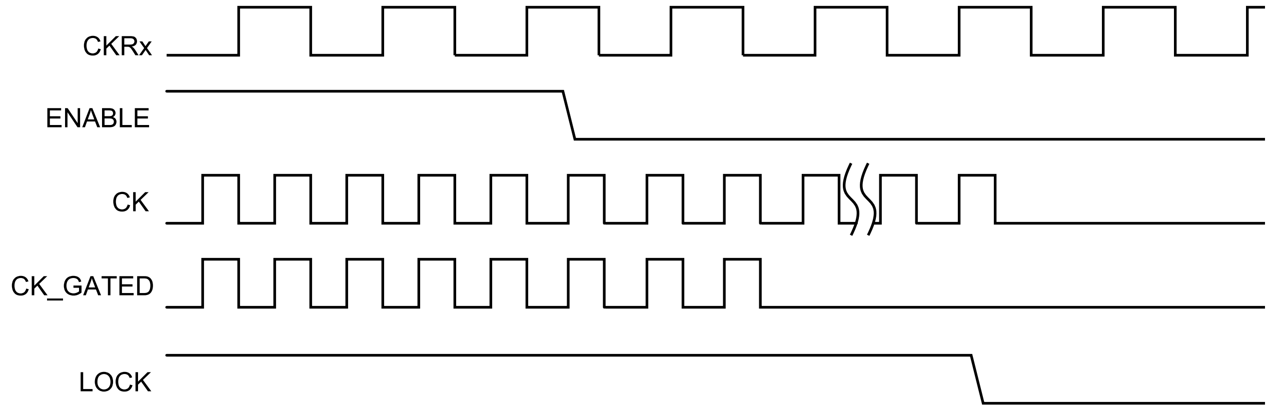


Figure 16-5. CK and CK_GATED Output from Running Mode to FDPLL96M Off Mode



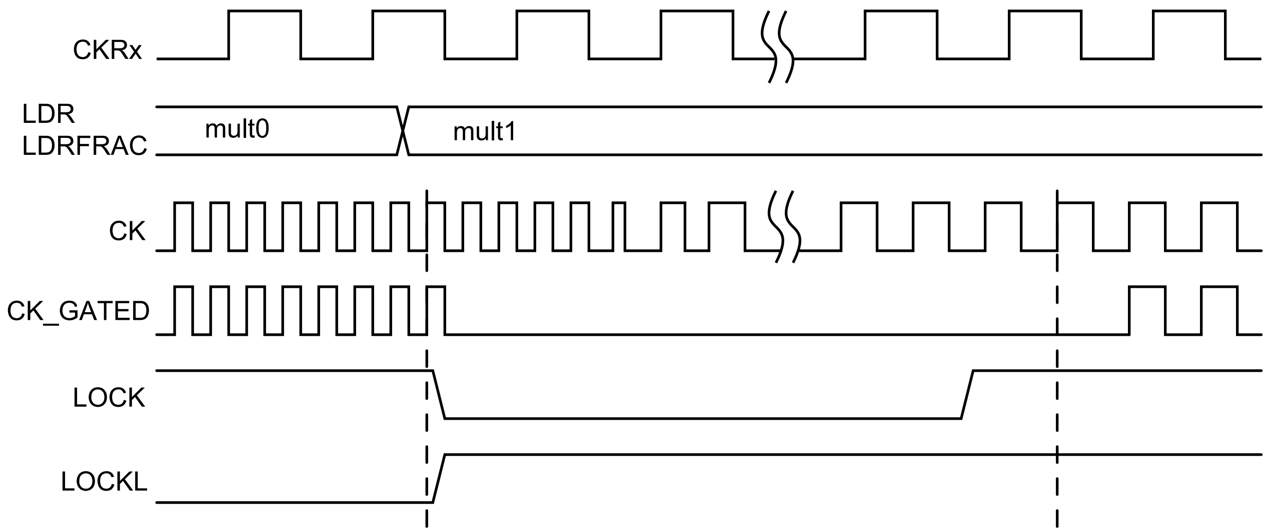
16.6.8.5 Reference Clock Switching

When a software operation requires reference clock switching, the normal operation is to disable the FDPLL96M, modify the DPLLCTRLB.REFCLK to select the desired reference source and activate the FDPLL96M again.

16.6.8.6 Loop Divider Ratio updates

The FDPLL96M supports on-the-fly update of the DPLLRATIO register, so it is allowed to modify the loop divider ratio and the loop divider ratio fractional part when the FDPLL96M is enabled. At that time, the DPLLSTATUS.LOCK bit is cleared and set again by hardware when the output frequency reached a stable state. The DPLL Lock Fail bit in the Interrupt Flag Status and Clear register (INTFLAG.DPLLLCK) is set when a falling edge has been detected. The flag is cleared when the software write a one to the interrupt flag bit location.

Figure 16-6. RATIOCTRL Register Update Operation



16.6.8.7 Digital Filter Selection

The PLL digital filter (PI controller) is automatically adjusted in order to provide a good compromise between stability and jitter. Nevertheless a software operation can override the filter setting using the DPLLCTRLB.FILTER field. The DPLLCTRLB.LPEN field can be use to bypass the TDC module.

16.6.9 Brown-Out Detector Operation

The SYSCTRL provides user control to two Brown-Out Detectors (BOD) monitoring two supply domains. One BOD monitors the 3.3V VDDANA supply (BOD33), and a second BOD monitors the 1.2V VDDCORE supply (BOD12).

Both Brown-Out Detectors support continuous or sampling modes.

For each BOD, the threshold value action (reset the device or generate an interrupt), the Hysteresis configuration, as well as the enable/disable settings are loaded from Flash User Calibration at startup, and can be overridden by writing to the corresponding user register bit groups.

16.6.10 3.3V Brown-Out Detector Operation

The 3.3V BOD monitors the 3.3V VDDANA supply (BOD33). It supports continuous or sampling modes.

The threshold value action (reset the device or generate an interrupt), the Hysteresis configuration, as well as the enable/disable settings are loaded from Flash User Calibration at startup, and can be overridden by writing to the corresponding BOD33 register bit groups.

16.6.10.1 3.3V Brown-Out Detector (BOD33)

The 3.3V Brown-Out Detector (BOD33) monitors the VDDANA supply and compares the voltage with the brown-out threshold level set in the BOD33 Level bit group (BOD33.LEVEL) in the BOD33 register. The BOD33 can generate either an interrupt or a reset when VDDANA crosses below the brown-out threshold level. The BOD33 detection status can be read from the BOD33 Detection bit (PCLKSR.BOD33DET) in the Power and Clocks Status register.

At startup or at power-on reset (POR), the BOD33 register values are loaded from the Flash User Row. Refer to "NVM User Row Mapping" on page 20 for more details.

16.6.10.2 Continuous Mode

When the BOD33 Mode bit (BOD33.MODE) in the BOD33 register is written to zero and the BOD33 is enabled, the BOD33 operates in continuous mode. In this mode, the BOD33 is continuously monitoring the VDDANA supply voltage.

When the BOD12 Mode bit (BOD12.MODE) in the BOD12 register is written to zero and the BOD12 is enabled(BOD12.ENABLE is written to one), the BOD12 operates in continuous mode. In this mode, the BOD12 is continuously monitoring the VDDCORE supply voltage. Continues mode is not available for BOD12 when running in standby sleep mode.

Continuous mode is the default mode for both BOD12 and BOD33.

16.6.10.3 Sampling Mode

The sampling mode is a low-power mode where the BOD33 or BOD12 is being repeatedly enabled on a sampling clock's ticks. The BOD33 or BOD12 will monitor the supply voltage for a short period of time and then go to a low-power disabled state until the next sampling clock tick.

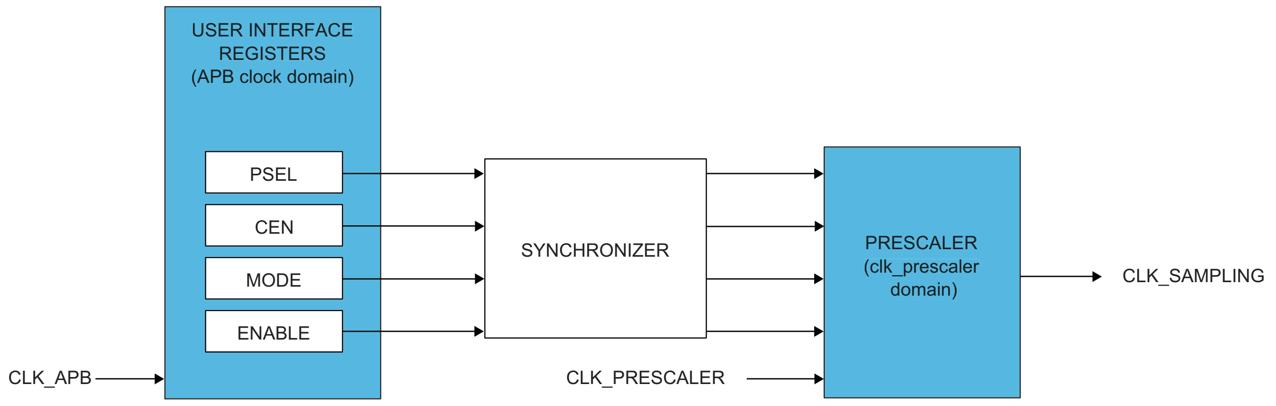
Sampling mode is enabled by writing one to BOD33.MODE for BOD33, and by writing one to BOD12.MODE for BOD12. The frequency of the clock ticks ($F_{clksampling}$) is controlled by the BOD33 Prescaler Select bit group (BOD33.PSEL) in the BOD33 register and Prescaler Select bit group(BOD12.PSEL) in the BOD12 BOD12 register for BOD33 and BOD12, respectively.

$$F_{clksampling} = \frac{F_{clkprescaler}}{2^{(PSEL + 1)}}$$

The prescaler signal ($F_{clkprescaler}$) is a 1kHz clock, output from the32kHz Ultra Low Power Oscillator, OSCULP32K.

As the sampling mode clock is different from the APB clock domain, synchronization among the clocks is necessary. Figure 16-7 shows a block diagram of the sampling mode. The BOD33 and BOD12 Synchronization Ready bits (PCLKSR.B33SRDY and PCLKSR.B12SRDY, respectively) in the Power and Clocks Status register show the synchronization ready status of the synchronizer. Writing attempts to the BOD33 register are ignored while PCLKSR.B33SRDY is zero. Writing attempts to the BOD12 register are ignored while PCLKSR.B12SRDY is zero.

Figure 16-7. Sampling Mode Block diagram



The BOD33 Clock Enable bit (BOD33.CEN) in the BOD33 register and the BOD12 Clock Enable bit (BOD12.CEN) in the BOD12 register should always be disabled before changing the prescaler value. To change the prescaler value for the BOD33 or BOD12 during sampling mode, the following steps need to be taken:

1. Wait until the PCLKSR.B33SRDY bit or the PCLKSR.B12SRDY bit is set.
2. Write the selected value to the BOD33.PSEL or BOD12.PSEL bit group.

16.6.10.4 Hysteresis

The hysteresis functionality can be used in both continuous and sampling mode. Writing a one to the BOD33 Hysteresis bit (BOD33.HYST) in the BOD33 register will add hysteresis to the BOD33 threshold level. Writing a one to the BOD12 Hysteresis bit (BOD12.HYST) in the BOD12 register will add hysteresis to the BOD12 threshold level.

16.6.11 Voltage Regulator System Operation

The embedded Voltage Regulator (VREG) is an internal voltage regulator that provides the core logic supply (VDDCORE).

16.6.11.1 User Control of the Voltage Regulator System

The Voltage Regulator is enabled after any reset, and can be disabled by writing a zero to the Enable bit (VREG.ENABLE) of the VREG register.

The Voltage Regulator output supply level is determined by the LEVEL bit group (VREG.LEVEL) value in the VREG register. At reset, the VREG.LEVEL register value is loaded from Flash Factory Calibration.

Via the VDDCORE Monitoring bit group (VREG.VDDMON), it is possible to monitor the core supply voltage so that if it drops below a critical level, a power-on reset is applied. The device is allowed to restart executing code only after the core supply voltage is restored to an acceptable level. The threshold at which this system triggers is significantly lower than the 1.2V Brown-Out Detector's own threshold (BOD12). This can, therefore, be seen as a complementary voltage monitoring feature.

16.6.12 Voltage Reference System Operation

The Voltage Reference System (VREF) consists of a Bandgap Reference Voltage Generator and a temperature sensor.

The Bandgap Reference Voltage Generator is factory-calibrated under typical voltage and temperature conditions.

At reset, the VREF.CAL register value is loaded from Flash Factory Calibration.

The temperature sensor can be used to get an absolute temperature in the temperature range of CMIN to CMAX degrees Celsius. The sensor will output a linear voltage proportional to the temperature. The output voltage and temperature range are located in the "Electrical Characteristics" on page 648. To calculate the temperature from a measured voltage, the following formula can be used:

$$C_{MIN} + (Vmes - Vout_{MAX})\frac{\Delta temperature}{\Delta voltage}$$

16.6.12.1 User Control of the Voltage Reference System

To enable the temperature sensor, write a one the Temperature Sensor Enable bit (VREF.TSEN) in the VREF register.

The temperature sensor can be redirected to the ADC for conversion. The Bandgap Reference Voltage Generator output can also be routed to the ADC if the Bandgap Output Enable bit (VREF.BGOUTEN) in the VREF register is set.

The Bandgap Reference Voltage Generator output level is determined by the CALIB bit group (VREF.CALIB) value in the VREF register.The default calibration value can be overridden by the user by writing to the CALIB bit group.

16.6.13 Interrupts

The SYSCTRL has the following interrupt sources:

- XOSCRDY - Multipurpose Crystal Oscillator Ready: A “0-to-1” transition on the PCLKSR.XOSCRDY bit is detected
- XOSC32KRDY - 32kHz Crystal Oscillator Ready: A “0-to-1” transition on the PCLKSR.XOSC32KRDY bit is detected
- OSC32KRDY - 32kHz Internal Oscillator Ready: A “0-to-1” transition on the PCLKSR.OSC32KRDY bit is detected
- OSC8MRDY - 8MHz Internal Oscillator Ready: A “0-to-1” transition on the PCLKSR.OSC8MRDY bit is detected
- DFLLRDY - DFLL48M Ready: A “0-to-1” transition on the PCLKSR.DFLLRDY bit is detected
- DFLLOOB - DFLL48M Out Of Boundaries: A “0-to-1” transition on the PCLKSR.DFLLOOB bit is detected
- DFLLLOCKF - DFLL48M Fine Lock: A “0-to-1” transition on the PCLKSR.DFLLLOCKF bit is detected
- DFLLLOCKC - DFLL48M Coarse Lock: A “0-to-1” transition on the PCLKSR.DFLLLOCKC bit is detected
- DFLLRCS - DFLL48M Reference Clock has Stopped: A “0-to-1” transition on the PCLKSR.DFLLRCS bit is detected
- BOD33RDY - BOD33 Ready: A “0-to-1” transition on the PCLKSR.BOD33RDY bit is detected
- BOD33DET - BOD33 Detection: A “0-to-1” transition on the PCLKSR.BOD33DET bit is detected
- B33SRDY - BOD33 Synchronization Ready: A “0-to-1” transition on the PCLKSR.B33SRDY bit is detected
- BOD12RDY - BOD12 Ready: A “0-to-1” transition on the PCLKSR.BOD12RDY bit is detected
- BOD12DET - BOD12 Detection: A “0-to-1” transition on the PCLKSR.BOD12DET bit is detected
- B12SRDY - BOD12 Synchronization Ready: A “0-to-1” transition on the PCLKSR.B12SRDY bit is detected
- PLL Lock (LOCK): Indicates that the DPLL Lock bit is asserted.
- PLL Lock Lost (LOCKL): Indicates that a falling edge has been detected on the Lock bit during normal operation mode.
- PLL Lock Timer Timeout (LTTO): This interrupt flag indicates that the software defined time DPLLCTRLB.LTIME has elapsed since the start of the FDPLL96M.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAG) register is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register. An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled, or the SYSCTRL is reset. See Interrupt Flag Status and Clear (INTFLAG) register for details on how to clear interrupt flags.

All interrupt requests from the peripheral are ORed together on system level to generate one combined interrupt request to the NVIC. Refer to “Nested Vector Interrupt Controller” on page 23 for details. The user must read the INTFLAG register to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to “Nested Vector Interrupt Controller” on page 23 for details.

16.6.14 Synchronization

Due to the multiple clock domains, values in the DFLL48M control registers need to be synchronized to other clock domains. The status of this synchronization can be read from the Power and Clocks Status register (PCLKSR). Before writing to any of the DFLL48M control registers, the user must check that the DFLL Ready bit (PCLKSR.DFLLRDY) in PCLKSR is set to one. When this bit is set, the DFLL48M can be configured and CLK_DFLL48M is ready to be used. Any write to any of the DFLL48M control registers while DFLLRDY is zero will be ignored. An interrupt is generated on a zero-to-one transition of DFLLRDY if the DFLLRDY bit (INTENSET.DFLLDY) in the Interrupt Enable Set register is set.

In order to read from any of the DFLL48M configuration registers, the user must request a read synchronization by writing a one to DFLLSYNC.READREQ. The registers can be read only when PCLKSR.DFLLRDY is set. If DFLLSYNC.READREQ is not written before a read, a synchronization will be started, and the bus will be halted until the synchronization is complete. Reading the DFLL48M registers when the DFLL48M is disabled will not halt the bus.

If the bus does not support waiting, a one must be written to the READREQ bit in the DFLL Synchronization register (DFLLSYNC.READREQ) before reading the value of a DFLL core register. The DFLL control registers are ready to be read when PCLKSR.DFLLRDY is set. If the bus does support waiting, this method can still be used to save time, but it would then also be possible to simply read the register directly. In this case, the bus will be halted until the synchronization has completed.

The prescaler counter used to trigger one-shot brown-out detections also operates asynchronously from the peripheral bus. As a consequence, the prescaler registers require synchronization when written or read. The synchronization results in a delay from when the initialization of the write or read operation begins until the operation is complete.

The write-synchronization is triggered by a write to the BOD12 or BOD33 control register. The Synchronization Ready bit (PCLKSR.B12SRDY or PCLKSR.B33SRDY) in the PCLKSR register will be cleared when the write-synchronization starts and set when the write-synchronization is complete. When the write-synchronization is ongoing (PCLKSR.B33SRDY or PCLKSR.B12SRDY is zero), an attempt to do any of the following will cause the peripheral bus to stall until the synchronization is complete:

- Writing to the BOD33 or BOD12 control register
- Reading the BOD33 or BOD12 control register that was written

The user can either poll PCLKSR.B12SRDY or PCLKSR.B33SRDY or use the INTENSET.B12SRDY or INTENSET.B33SRDY interrupts to check when the synchronization is complete. It is also possible to perform the next read/write operation and wait, as this next operation will be completed after the ongoing read/write operation is synchronized.

16.7 Register Summary

Table 16-7. Register Summary

Offset	Name	Bit Pos.								
0x00	INTENCLR	7:0	DFLLLCKC	DFLLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
0x01		15:8	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS
0x02		23:16							DPLLLTO	DPLLLCKF
0x03		31:24								
0x04	INTENSET	7:0	DFLLLCKC	DFLLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
0x05		15:8	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS
0x06		23:16							DPLLLTO	DPLLLCKF
0x07		31:24								
0x08	INTFLAG	7:0	DFLLLCKC	DFLLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
0x09		15:8	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS
0x0A		23:16							DPLLLTO	DPLLLCKF
0x0B		31:24								
0x0C	PCLKSR	7:0	DFLLLCKC	DFLLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
0x0D		15:8	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS
0x0E		23:16							DPLLLTO	DPLLLCKF
0x0F		31:24								
0x10	XOSC	7:0	ONDEMAND	RUNSTDBY				XTALEN	ENABLE	
0x11		15:8	STARTUP[3:0]				AMPGC	GAIN[2:0]		
0x12	Reserved									
0x13	Reserved									
0x14	XOSC32K	7:0	ONDEMAND	RUNSTDBY	AAMPEN	EN1K	EN32K	XTALEN	ENABLE	
0x15		15:8				WRTLOCK		STARTUP[2:0]		
0x16	Reserved									
0x17	Reserved									
0x18	OSC32K	7:0	ONDEMAND	RUNSTDBY			EN1K	EN32K	ENABLE	
0x19		15:8				WRTLOCK		STARTUP[2:0]		
0x1A		23:16		CALIB[6:0]						
0x1B		31:24								
0x1C	OSCULP32K	7:0	WRTLOCK			CALIB[4:0]				
0x1D ... 0x1F	Reserved									
0x20	OSC8M	7:0	ONDEMAND	RUNSTDBY					ENABLE	
0x21		15:8							PRESC[1:0]	
0x22		23:16	CALIB[7:0]							
0x23		31:24	FRANGE[1:0]				CALIB[11:8]			
0x24	DFLLCTRL	7:0	ONDEMAND			LLAW	STABLE	MODE	ENABLE	
0x25		15:8					WAITLOCK	BPLCKC	QLDIS	CCDIS
0x26	Reserved									
0x27	Reserved									

Offset	Name	Bit Pos.								
0x28	DFLLVAL	7:0	FINE[7:0]							
0x29		15:8	COARSE[5:0]						FINE[9:8]	
0x2A		23:16	DIFF[7:0]							
0x2B		31:24	DIFF[15:8]							
0x2C	DFLLMUL	7:0	MUL[7:0]							
0x2D		15:8	MUL[15:8]							
0x2E		23:16	FSTEP[7:0]							
0x2F		31:24	CSTEP[5:0]						FSTEP[9:8]	
0x30	DFLLSYNC	7:0	READREQ							
0x31 ... 0x33	Reserved									
0x34	BOD33	7:0		RUNSTDBY		ACTION[1:0]		HYST	ENABLE	
0x35		15:8	PSEL[3:0]						CEN	MODE
0x36		23:16			LEVEL[5:0]					
0x37		31:24								
0x38 ... 0x3F	Reserved									
0x40	VREF	7:0						BGOUTEN	TSEN	
0x41		15:8								
0x42		23:16	CALIB[7:0]							
0x43		31:24						CALIB[10:8]		
0x44	DPLLCTRLA	7:0	ONDEMAND						ENABLE	
0x45 ... 0x47	Reserved									
0x48	DPLLRATIO	7:0	LDR[7:0]							
0x49		15:8					LDR[11:8]			
0x4A		23:16					LDRFRAC[3:0]			
0x4B		31:24								
0x4C	DPLLCTRLB	7:0			REFCLK[1:0]		WUF	LPEN	FILTER[1:0]	
0x4D		15:8				LBYPASS		LTIME[2:0]		
0x4E		23:16	DIV[7:0]							
0x4F		31:24						DIV[10:8]		
0x50	DPLLSTATUS	7:0					DIV	ENABLE	CLKRDY	LOCK

16.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to "Register Access Protection" on page 145 and the "PAC – Peripheral Access Controller" on page 27 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Synchronized property in each individual register description. Refer to "Synchronization" on page 158 for details.

16.8.1 Interrupt Enable Clear

Name: INTENCLR

Offset: 0x00

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
							DPLLLTO	DPLLLCKF
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS
Access	R/W	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DFLLLCKC	DFLLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

- **Bits 31:18 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 17 – DPLLLTO: DPLL Lock Timeout Interrupt Enable**
  0: The DPLL Lock Timeout interrupt is disabled.

  1: The DPLL Lock Timeout interrupt is enabled, and an interrupt request will be generated when the DPLL Lock Timeout Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the DPLL Lock Timeout Interrupt Enable bit, which disables the DPLL Lock Time-out interrupt.

- **Bit 16 – DPLLLCKF: DPLL Lock Fall Interrupt Enable**
  0: The DPLL Lock Fall interrupt is disabled.

1: The DPLL Lock Fall interrupt is enabled, and an interrupt request will be generated when the DPLL Lock Fall Interrupt flag is set.

Writing a zero to this bit has no effect.

Writing a one to this bit will clear the DPLL Lock Fall Interrupt Enable bit, which disables the DPLL Lock Fall interrupt.

- **Bit 15 – DPLLLCKR: DPLL Lock Rise Interrupt Enable**

  0: The DPLL Lock Rise interrupt is disabled.

  1: The DPLL Lock Rise interrupt is enabled, and an interrupt request will be generated when the DPLL Lock Rise Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the DPLL Lock Rise Interrupt Enable bit, which disables the DPLL Lock Rise interrupt.

- **Bits 14:12 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 11 – B33SRDY: BOD33 Synchronization Ready Interrupt Enable**

  0: The BOD33 Synchronization Ready interrupt is disabled.

  1: The BOD33 Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the BOD33 Synchronization Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the BOD33 Synchronization Ready Interrupt Enable bit, which disables the BOD33 Synchronization Ready interrupt.

- **Bit 10 – BOD33DET: BOD33 Detection Interrupt Enable**

  0: The BOD33 Detection interrupt is disabled.

  1: The BOD33 Detection interrupt is enabled, and an interrupt request will be generated when the BOD33 Detection Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the BOD33 Detection Interrupt Enable bit, which disables the BOD33 Detection interrupt.

- **Bit 9 – BOD33RDY: BOD33 Ready Interrupt Enable**

  0: The BOD33 Ready interrupt is disabled.

  1: The BOD33 Ready interrupt is enabled, and an interrupt request will be generated when the BOD33 Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the BOD33 Ready Interrupt Enable bit, which disables the BOD33 Ready interrupt.

- **Bit 8 – DFLLRCS: DFLL Reference Clock Stopped Interrupt Enable**

  0: The DFLL Reference Clock Stopped interrupt is disabled.

  1: The DFLL Reference Clock Stopped interrupt is enabled, and an interrupt request will be generated when the DFLL Reference Clock Stopped Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the DFLL Reference Clock Stopped Interrupt Enable bit, which disables the DFLL Reference Clock Stopped interrupt.

- **Bit 7 – DFLLLCKC: DFLL Lock Coarse Interrupt Enable**

  0: The DFLL Lock Coarse interrupt is disabled.

1: The DFLL Lock Coarse interrupt is enabled, and an interrupt request will be generated when the DFLL Lock Coarse Interrupt flag is set.

Writing a zero to this bit has no effect.

Writing a one to this bit will clear the DFLL Lock Coarse Interrupt Enable bit, which disables the DFLL Lock Coarse interrupt.

- **Bit 6 – DFLLLCKF: DFLL Lock Fine Interrupt Enable**

  0: The DFLL Lock Fine interrupt is disabled.

  1: The DFLL Lock Fine interrupt is enabled, and an interrupt request will be generated when the DFLL Lock Fine Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the DFLL Lock Fine Interrupt Enable bit, which disables the DFLL Lock Fine interrupt.

- **Bit 5 – DFLLOOB: DFLL Out Of Bounds Interrupt Enable**

  0: The DFLL Out Of Bounds interrupt is disabled.

  1: The DFLL Out Of Bounds interrupt is enabled, and an interrupt request will be generated when the DFLL Out Of Bounds Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the DFLL Out Of Bounds Interrupt Enable bit, which disables the DFLL Out Of Bounds interrupt.

- **Bit 4 – DFLLRDY: DFLL Ready Interrupt Enable**

  0: The DFLL Ready interrupt is disabled.

  1: The DFLL Ready interrupt is enabled, and an interrupt request will be generated when the DFLL Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the DFLL Ready Interrupt Enable bit, which disables the DFLL Ready interrupt.

- **Bit 3 – OSC8MRDY: OSC8M Ready Interrupt Enable**

  0: The OSC8M Ready interrupt is disabled.

  1: The OSC8M Ready interrupt is enabled, and an interrupt request will be generated when the OSC8M Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the OSC8M Ready Interrupt Enable bit, which disables the OSC8M Ready interrupt.

- **Bit 2 – OSC32KRDY: OSC32K Ready Interrupt Enable**

  0: The OSC32K Ready interrupt is disabled.

  1: The OSC32K Ready interrupt is enabled, and an interrupt request will be generated when the OSC32K Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the OSC32K Ready Interrupt Enable bit, which disables the OSC32K Ready interrupt.

- **Bit 1 – XOSC32KRDY: XOSC32K Ready Interrupt Enable**

  0: The XOSC32K Ready interrupt is disabled.

  1: The XOSC32K Ready interrupt is enabled, and an interrupt request will be generated when the XOSC32K Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the XOSC32K Ready Interrupt Enable bit, which disables the XOSC32K Ready interrupt.

- **Bit 0 – XOSCRDY: XOSC Ready Interrupt Enable**

  0: The XOSC Ready interrupt is disabled.

  1: The XOSC Ready interrupt is enabled, and an interrupt request will be generated when the XOSC Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the XOSC Ready Interrupt Enable bit, which disables the XOSC Ready interrupt.

16.8.2 Interrupt Enable Set

Name: INTENSET

Offset: 0x04

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
							DPLLLTO	DPLLLCKF
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS
Access	R/W	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DFLLLCKC	DFLLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

- **Bits 31:18 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 17 – DPLLLTO: DPLL Lock Timeout Interrupt Enable**
  0: The DPLL Lock Timeout interrupt is disabled.

  1: The DPLL Lock Timeout interrupt is enabled, and an interrupt request will be generated when the DPLL Lock Timeout Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DPLL Lock Timeout Interrupt Enable bit, which enables the DPLL Lock Timeout interrupt.

- **Bit 16 – DPLLLCKF: DPLL Lock Fall Interrupt Enable**
  0: The DPLL Lock Fall interrupt is disabled.

1: The DPLL Lock Fall interrupt is enabled, and an interrupt request will be generated when the DPLL Lock Fall Interrupt flag is set.

Writing a zero to this bit has no effect.

Writing a one to this bit will set the DPLL Lock Fall Interrupt Enable bit, which enables the DPLL Lock Fall interrupt.

- **Bit 15 – DPLLLCKR: DPLL Lock Rise Interrupt Enable**

  0: The DPLL Lock Rise interrupt is disabled.

  1: The DPLL Lock Rise interrupt is enabled, and an interrupt request will be generated when the DPLL Lock Rise Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DPLL Lock Rise Interrupt Enable bit, which enables the DPLL Lock Rise interrupt.

- **Bits 14:12 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 11 – B33SRDY: BOD33 Synchronization Ready Interrupt Enable**

  0: The BOD33 Synchronization Ready interrupt is disabled.

  1: The BOD33 Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the BOD33 Synchronization Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the BOD33 Synchronization Ready Interrupt Enable bit, which enables the BOD33 Synchronization Ready interrupt.

- **Bit 10 – BOD33DET: BOD33 Detection Interrupt Enable**

  0: The BOD33 Detection interrupt is disabled.

  1: The BOD33 Detection interrupt is enabled, and an interrupt request will be generated when the BOD33 Detection Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the BOD33 Detection Interrupt Enable bit, which enables the BOD33 Detection interrupt.

- **Bit 9 – BOD33RDY: BOD33 Ready Interrupt Enable**

  0: The BOD33 Ready interrupt is disabled.

  1: The BOD33 Ready interrupt is enabled, and an interrupt request will be generated when the BOD33 Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the BOD33 Ready Interrupt Enable bit, which enables the BOD33 Ready interrupt.

- **Bit 8 – DFLLRCS: DFLL Reference Clock Stopped Interrupt Enable**

  0: The DFLL Reference Clock Stopped interrupt is disabled.

  1: The DFLL Reference Clock Stopped interrupt is enabled, and an interrupt request will be generated when the DFLL Reference Clock Stopped Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DFLL Reference Clock Stopped Interrupt Enable bit, which enables the DFLL Reference Clock Stopped interrupt.

- **Bit 7 – DFLLLCKC: DFLL Lock Coarse Interrupt Enable**

  0: The DFLL Lock Coarse interrupt is disabled.

  1: The DFLL Lock Coarse interrupt is enabled, and an interrupt request will be generated when the DFLL Lock Coarse Interrupt flag is set.

Writing a zero to this bit has no effect.

Writing a one to this bit will set the DFLL Lock Coarse Interrupt Enable bit, which enables the DFLL Lock Coarse interrupt.

- **Bit 6 – DFLLLCKF: DFLL Lock Fine Interrupt Enable**

  0: The DFLL Lock Fine interrupt is disabled.

  1: The DFLL Lock Fine interrupt is enabled, and an interrupt request will be generated when the DFLL Lock Fine Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DFLL Lock Fine Interrupt Disable/Enable bit, disable the DFLL Lock Fine interrupt and set the corresponding interrupt request.

- **Bit 5 – DFLLOOB: DFLL Out Of Bounds Interrupt Enable**

  0: The DFLL Out Of Bounds interrupt is disabled.

  1: The DFLL Out Of Bounds interrupt is enabled, and an interrupt request will be generated when the DFLL Out Of Bounds Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DFLL Out Of Bounds Interrupt Enable bit, which enables the DFLL Out Of Bounds interrupt.

- **Bit 4 – DFLLRDY: DFLL Ready Interrupt Enable**

  0: The DFLL Ready interrupt is disabled.

  1: The DFLL Ready interrupt is enabled, and an interrupt request will be generated when the DFLL Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DFLL Ready Interrupt Enable bit, which enables the DFLL Ready interrupt and set the corresponding interrupt request.

- **Bit 3 – OSC8MRDY: OSC8M Ready Interrupt Enable**

  0: The OSC8M Ready interrupt is disabled.

  1: The OSC8M Ready interrupt is enabled, and an interrupt request will be generated when the OSC8M Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the OSC8M Ready Interrupt Enable bit, which enables the OSC8M Ready interrupt.

- **Bit 2 – OSC32KRDY: OSC32K Ready Interrupt Enable**

  0: The OSC32K Ready interrupt is disabled.

  1: The OSC32K Ready interrupt is enabled, and an interrupt request will be generated when the OSC32K Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the OSC32K Ready Interrupt Enable bit, which enables the OSC32K Ready interrupt.

- **Bit 1 – XOSC32KRDY: XOSC32K Ready Interrupt Enable**

  0: The XOSC32K Ready interrupt is disabled.

  1: The XOSC32K Ready interrupt is enabled, and an interrupt request will be generated when the XOSC32K Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the XOSC32K Ready Interrupt Enable bit, which enables the XOSC32K Ready interrupt.

- **Bit 0 – XOSCRDY: XOSC Ready Interrupt Enable**

  0: The XOSC Ready interrupt is disabled.

  1: The XOSC Ready interrupt is enabled, and an interrupt request will be generated when the XOSC Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the XOSC Ready Interrupt Enable bit, which enables the XOSC Ready interrupt.

16.8.3 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x08

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
							DPLLLTO	DPLLLCKF
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS
Access	R/W	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DFLLLCKC	DFLLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Note: Depending on the fuse settings, various bits of the INTFLAG register can be set to one at startup. Therefore the user should clear those bits before using the corresponding interrupts.

- **Bits 31:18 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 17 – DPLLLTO: DPLL Lock Timeout**
  This flag is cleared by writing a one to it.

  This flag is set on a zero-to-one transition of the DPLL Lock Timeout bit in the Status register (PCLKSR.DPLLLTO) and will generate an interrupt request if INTENSET.DPLLLTO is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the DPLL Lock Timeout interrupt flag.

- **Bit 16 – DPLLLCKF: DPLL Lock Fall**
  This flag is cleared by writing a one to it.

  This flag is set on a zero-to-one transition of the DPLL Lock Fall bit in the Status register (PCLKSR.DPLLLCKF) and will generate an interrupt request if INTENSET.DPLLLCKF is one.

Writing a zero to this bit has no effect.
Writing a one to this bit clears the DPLL Lock Fall interrupt flag.

- **Bit 15 – DPLLLCKR: DPLL Lock Rise**
  This flag is cleared by writing a one to it.
  This flag is set on a zero-to-one transition of the DPLL Lock Rise bit in the Status register (PCLKSR.DPLLLCKR) and will generate an interrupt request if INTENSET.DPLLLCKR is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the DPLL Lock Rise interrupt flag.

- **Bits 14:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 11 – B33SRDY: BOD33 Synchronization Ready**
  This flag is cleared by writing a one to it.
  This flag is set on a zero-to-one transition of the BOD33 Synchronization Ready bit in the Status register (PCLKSR.B33SRDY) and will generate an interrupt request if INTENSET.B33SRDY is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the BOD33 Synchronization Ready interrupt flag

- **Bit 10 – BOD33DET: BOD33 Detection**
  This flag is cleared by writing a one to it.
  This flag is set on a zero-to-one transition of the BOD33 Detection bit in the Status register (PCLKSR.BOD33DET) and will generate an interrupt request if INTENSET.BOD33DET is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the BOD33 Detection interrupt flag.

- **Bit 9 – BOD33RDY: BOD33 Ready**
  This flag is cleared by writing a one to it.
  This flag is set on a zero-to-one transition of the BOD33 Ready bit in the Status register (PCLKSR.BOD33RDY) and will generate an interrupt request if INTENSET.BOD33RDY is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the BOD33 Ready interrupt flag.

- **Bit 8 – DFLLRCS: DFLL Reference Clock Stopped**
  This flag is cleared by writing a one to it.
  This flag is set on a zero-to-one transition of the DFLL Reference Clock Stopped bit in the Status register (PCLKSR.DFLLRCS) and will generate an interrupt request if INTENSET.DFLLRCS is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the DFLL Reference Clock Stopped interrupt flag.

- **Bit 7 – DFLLLCKC: DFLL Lock Coarse**
  This flag is cleared by writing a one to it.
  This flag is set on a zero-to-one transition of the DFLL Lock Coarse bit in the Status register (PCLKSR.DFLLL-CKC) and will generate an interrupt request if INTENSET.DFLLLCKC is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the DFLL Lock Coarse interrupt flag.

- **Bit 6 – DFLLLCKF: DFLL Lock Fine**
  This flag is cleared by writing a one to it.
  This flag is set on a zero-to-one transition of the DFLL Lock Fine bit in the Status register (PCLKSR.DFLLLCKF) and will generate an interrupt request if INTENSET.DFLLLCKF is one.

Writing a zero to this bit has no effect.

Writing a one to this bit clears the DFLL Lock Fine interrupt flag.

- **Bit 5 – DFLLOOB: DFLL Out Of Bounds**

  This flag is cleared by writing a one to it.

  This flag is set on a zero-to-one transition of the DFLL Out Of Bounds bit in the Status register (PCLKSR.DFLLOOB) and will generate an interrupt request if INTENSET.DFLLOOB is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the DFLL Out Of Bounds interrupt flag.

- **Bit 4 – DFLLRDY: DFLL Ready**

  This flag is cleared by writing a one to it.

  This flag is set on a zero-to-one transition of the DFLL Ready bit in the Status register (PCLKSR.DFLLRDY) and will generate an interrupt request if INTENSET.DFLLRDY is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the DFLL Ready interrupt flag.

- **Bit 3 – OSC8MRDY: OSC8M Ready**

  This flag is cleared by writing a one to it.

  This flag is set on a zero-to-one transition of the OSC8M Ready bit in the Status register (PCLKSR.OSC8MRDY) and will generate an interrupt request if INTENSET.OSC8MRDY is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the OSC8M Ready interrupt flag.

- **Bit 2 – OSC32KRDY: OSC32K Ready**

  This flag is cleared by writing a one to it.

  This flag is set on a zero-to-one transition of the OSC32K Ready bit in the Status register (PCLKSR.OSC32KRDY) and will generate an interrupt request if INTENSET.OSC32KRDY is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the OSC32K Ready interrupt flag.

- **Bit 1 – XOSC32KRDY: XOSC32K Ready**

  This flag is cleared by writing a one to it.

  This flag is set on a zero-to-one transition of the XOSC32K Ready bit in the Status register (PCLKSR.XOSC32KRDY) and will generate an interrupt request if INTENSET.XOSC32KRDY is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the XOSC32K Ready interrupt flag.

- **Bit 0 – XOSCRDY: XOSC Ready**

  This flag is cleared by writing a one to it.

  This flag is set on a zero-to-one transition of the XOSC Ready bit in the Status register (PCLKSR.XOSCRDY) and will generate an interrupt request if INTENSET.XOSCRDY is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the XOSC Ready interrupt flag.

16.8.4 Power and Clocks Status

Name: PCLKSR

Offset: 0x0C

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
							DPLLLTO	DPLLLCKF
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DFLLLCKC	DFLLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:18 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 17 – DPLLLTO: DPLL Lock Timeout**
  0: DPLL Lock time-out not detected.

  1: DPLL Lock time-out detected.

- **Bit 16 – DPLLLCKF: DPLL Lock Fall**
  0: DPLL Lock fall edge not detected.

  1: DPLL Lock fall edge detected.

- **Bit 15 – DPLLLCKR: DPLL Lock Rise**
  0: DPLL Lock rise edge not detected.

  1: DPLL Lock fall edge detected.

- **Bits 14:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 11 – B33SRDY: BOD33 Synchronization Ready**
  0: BOD33 synchronization is complete.
  1: BOD33 synchronization is ongoing.

- **Bit 10 – BOD33DET: BOD33 Detection**
  0: No BOD33 detection.
  1: BOD33 has detected that the I/O power supply is going below the BOD33 reference value.

- **Bit 9 – BOD33RDY: BOD33 Ready**
  0: BOD33 is not ready.
  1: BOD33 is ready.

- **Bit 8 – DFLLRCS: DFLL Reference Clock Stopped**
  0: DFLL reference clock is running.
  1: DFLL reference clock has stopped.

- **Bit 7 – DFLLLCKC: DFLL Lock Coarse**
  0: No DFLL coarse lock detected.
  1: DFLL coarse lock detected.

- **Bit 6 – DFLLLCKF: DFLL Lock Fine**
  0: No DFLL fine lock detected.
  1: DFLL fine lock detected.

- **Bit 5 – DFLLOOB: DFLL Out Of Bounds**
  0: No DFLL Out Of Bounds detected.
  1: DFLL Out Of Bounds detected.

- **Bit 4 – DFLLRDY: DFLL Ready**
  0: The Synchronization is ongoing.
  1: The Synchronization is complete.
  This bit is cleared when the synchronization of registers between clock domains is complete.
  This bit is set when the synchronization of registers between clock domains is started.

- **Bit 3 – OSC8MRDY: OSC8M Ready**
  0: OSC8M is not ready.
  1: OSC8M is stable and ready to be used as a clock source.

- **Bit 2 – OSC32KRDY: OSC32K Ready**
  0: OSC32K is not ready.
  1: OSC32K is stable and ready to be used as a clock source.

- **Bit 1 – XOSC32KRDY: XOSC32K Ready**
  0: XOSC32K is not ready.
  1: XOSC32K is stable and ready to be used as a clock source.

- **Bit 0 – XOSCRDY: XOSC Ready**
  0: XOSC is not ready.
  1: XOSC is stable and ready to be used as a clock source.

16.8.5 External Multipurpose Crystal Oscillator (XOSC) Control

Name: XOSC

Offset: 0x10

Reset: 0x0080

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
	STARTUP[3:0]				AMPGC	GAIN[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY				XTALEN	ENABLE	
Access	R/W	R/W	R	R	R	R/W	R/W	R
Reset	1	0	0	0	0	0	0	0

- **Bits 15:12 – STARTUP[3:0]: Start-Up Time**
  These bits select start-up time for the oscillator according to the table below.
  The OSCULP32K oscillator is used to clock the start-up counter.

Table 16-8. Start-UpTime for External Multipurpose Crystal Oscillator

STARTUP[3:0]	Number of OSCULP32K Clock Cycles	Number of XOSC Clock Cycles	Approximate Equivalent Time[(1)(2)(3)]
0x0	1	3	31µs
0x1	2	3	61µs
0x2	4	3	122µs
0x3	8	3	244µs
0x4	16	3	488µs
0x5	32	3	977µs
0x6	64	3	1953µs
0x7	128	3	3906µs
0x8	256	3	7813µs
0x9	512	3	15625µs
0xA	1024	3	31250µs
0xB	2048	3	62500µs
0xC	4096	3	125000µs
0xD	8192	3	250000µs
0xE	16384	3	500000µs
0xF	32768	3	1000000µs

Notes: 1. Number of cycles for the start-up counter
2. Number of cycles for the synchronization delay, before PCLKSR.XOSCRDY is set.
3. Actual start-up time is n OSCULP32K cycles + 3 XOSC cycles, but given the time neglects the 3 XOSC cycles.

- **Bit 11 – AMPGC: Automatic Amplitude Gain Control**
  0: The automatic amplitude gain control is disabled.
  1: The automatic amplitude gain control is enabled. Amplitude gain will be automatically adjusted during Crystal Oscillator operation.

- **Bits 10:8 – GAIN[2:0]: Oscillator Gain**
  These bits select the gain for the oscillator. The listed maximum frequencies are recommendations, and might vary based on capacitive load and crystal characteristics. Setting this bit group has no effect when the Automatic Amplitude Gain Control is active.

Table 16-9. Oscillator Gain

GAIN[2:0]	Name	Recommended Max Frequency
0x0	0	2MHz
0x1	1	4MHz
0x2	2	8MHz

GAIN[2:0]	Name	Recommended Max Frequency
0x3	3	16MHz
0x4	4	30MHz
0x5-0x7		Reserved

- **Bit 7 – ONDEMAND: On Demand Control**
  The On Demand operation mode allows an oscillator to be enabled or disabled, depending on peripheral clock requests.
  In On Demand operation mode, i.e., if the XOSC.ONDEMAND bit has been previously written to one, the oscillator will be running only when requested by a peripheral. If there is no peripheral requesting the oscillator s clock source, the oscillator will be in a disabled state.
  If On Demand is disabled, the oscillator will always be running when enabled.
  In standby sleep mode, the On Demand operation is still active if the XOSC.RUNSTDBY bit is one. If XOSC.RUNSTDBY is zero, the oscillator is disabled.
  0: The oscillator is always on, if enabled.
  1: The oscillator is enabled when a peripheral is requesting the oscillator to be used as a clock source. The oscillator is disabled if no peripheral is requesting the clock source.

- **Bit 6 – RUNSTDBY: Run in Standby**
  This bit controls how the XOSC behaves during standby sleep mode:
  0: The oscillator is disabled in standby sleep mode.
  1: The oscillator is not stopped in standby sleep mode. If XOSC.ONDEMAND is one, the clock source will be running when a peripheral is requesting the clock. If XOSC.ONDEMAND is zero, the clock source will always be running in standby sleep mode.

- **Bits 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – XTALEN: Crystal Oscillator Enable**
  This bit controls the connections between the I/O pads and the external clock or crystal oscillator:
  0: External clock connected on XIN. XOUT can be used as general-purpose I/O.
  1: Crystal connected to XIN/XOUT.

- **Bit 1 – ENABLE: Oscillator Enable**
  0: The oscillator is disabled.
  1: The oscillator is enabled.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

16.8.6 32kHz External Crystal Oscillator (XOSC32K) Control

Name: XOSC32K

Offset: 0x14

Reset: 0x0080

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
				WRTLOCK		STARTUP[2:0]		
Access	R	R	R	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY	AAMPEN	EN1K	EN32K	XTALEN	ENABLE	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	1	0	0	0	0	0	0	0

- **Bits 15:13 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 12 – WRTLOCK: Write Lock**
  This bit locks the XOSC32K register for futur writes to fix the XOSC32K configuration.

  0: The XOSC32K configuration is not locked.

  1: The XOSC32K configuration is locked.

- **Bit 11 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 10:8 – STARTUP[2:0]: Oscillator Start-Up Time**
  These bits select the start-up time for the oscillator according to Table 16-10.

  The OSCULP32K oscillator is used to clock the start-up counter.

Table 16-10. Start-Up Time for 32kHz External Crystal Oscillator

STARTUP[2:0]	Number of OSCULP32K Clock Cycles	Number of XOSC32K Clock Cycles	Approximate Equivalent Time (OSCULP = 32kHz)[1][2][3]
0x0	1	3	122μs
0x1	32	3	1068μs
0x2	2048	3	62592μs
0x3	4096	3	125092μs
0x4	16384	3	500092μs
0x5	32768	3	1000092μs
0x6	65536	3	2000092μs
0x7	131072	3	4000092μs

Notes:
1. Number of cycles for the start-up counter.
2. Number of cycles for the synchronization delay, before PCLKSR.XOSC32KRDY is set.
3. Start-up time is n OSCULP32K cycles + 3 XOSC32K cycles.

- **Bit 7 – ONDEMAND: On Demand Control**
  The On Demand operation mode allows an oscillator to be enabled or disabled depending on peripheral clock requests.
  In On Demand operation mode, i.e., if the ONDEMAND bit has been previously written to one, the oscillator will only be running when requested by a peripheral. If there is no peripheral requesting the oscillator s clock source, the oscillator will be in a disabled state.
  If On Demand is disabled the oscillator will always be running when enabled.
  In standby sleep mode, the On Demand operation is still active if the XOSC32K.RUNSTDBY bit is one. If XOSC32K.RUNSTDBY is zero, the oscillator is disabled.
  0: The oscillator is always on, if enabled.
  1: The oscillator is enabled when a peripheral is requesting the oscillator to be used as a clock source. The oscillator is disabled if no peripheral is requesting the clock source.

- **Bit 6 – RUNSTDBY: Run in Standby**
  This bit controls how the XOSC32K behaves during standby sleep mode:
  0: The oscillator is disabled in standby sleep mode.
  1: The oscillator is not stopped in standby sleep mode. If XOSC32K.ONDEMAND is one, the clock source will be running when a peripheral is requesting the clock. If XOSC32K.ONDEMAND is zero, the clock source will always be running in standby sleep mode.

- **Bit 5 – AAMPEN: Automatic Amplitude Control Enable**
  0: The automatic amplitude control for the crystal oscillator is disabled.
  1: The automatic amplitude control for the crystal oscillator is enabled.

- **Bit 4 – EN1K: 1kHz Output Enable**
  0: The 1kHz output is disabled.
  1: The 1kHz output is enabled.

- **Bit 3 – EN32K: 32kHz Output Enable**
  0: The 32kHz output is disabled.

1: The 32kHz output is enabled.

- **Bit 2 – XTALEN: Crystal Oscillator Enable**
  This bit controls the connections between the I/O pads and the external clock or crystal oscillator:
  0: External clock connected on XIN32. XOUT32 can be used as general-purpose I/O.
  1: Crystal connected to XIN32/XOUT32.

- **Bit 1 – ENABLE: Oscillator Enable**
  0: The oscillator is disabled.
  1: The oscillator is enabled.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

16.8.7 32kHz Internal Oscillator (OSC32K) Control

Name: OSC32K

Offset: 0x18

Reset: 0x003F0080

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		CALIB[6:0]						
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
				WRTLOCK		STARTUP[2:0]		
Access	R	R	R	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY			EN1K	EN32K	ENABLE	
Access	R/W	R/W	R	R	R/W	R/W	R/W	R
Reset	1	0	0	0	0	0	0	0

- **Bits 31:23 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 22:16 – CALIB[6:0]: Oscillator Calibration**
  These bits control the oscillator calibration.

  This value must be written by the user.

  Factory calibration values can be loaded from the non-volatile memory. Refer to "NVM Software Calibration Row Mapping" on page 21.

- **Bits 15:13 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 12 – WRTLOCK: Write Lock**
  This bit locks the OSC32K register for futur writes to fix the OSC32K configuration.

  0: The OSC32K configuration is not locked.

  1: The OSC32K configuration is locked.

- **Bit 11 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 10:8 – STARTUP[2:0]: Oscillator Start-Up Time**
  These bits select start-up time for the oscillator according to Table 16-11.
  The OSCULP32K oscillator is used as input clock to the startup counter.

Table 16-11. Start-Up Time for 32kHz Internal Oscillator

STARTUP[2:0]	Number of OSC32K clock cycles	Approximate Equivalent Time (OSCULP= 32 kHz)(1)(2)(3)
0x0	3	92µs
0x1	4	122µs
0x2	6	183µs
0x3	10	305µs
0x4	18	549µs
0x5	34	1038µs
0x6	66	2014µs
0x7	130	3967µs

Notes: 1. Number of cycles for the start-up counter.
2. Number of cycles for the synchronization delay, before PCLKSR.OSC32KRDY is set.
3. Start-up time is n OSC32K cycles + 2 OSC32K cycles.

- **Bit 7 – ONDEMAND: On Demand Control**
  The On Demand operation mode allows an oscillator to be enabled or disabled depending on peripheral clock requests.
  In On Demand operation mode, i.e., if the ONDEMAND bit has been previously written to one, the oscillator will only be running when requested by a peripheral. If there is no peripheral requesting the oscillator s clock source, the oscillator will be in a disabled state.
  If On Demand is disabled the oscillator will always be running when enabled.
  In standby sleep mode, the On Demand operation is still active if the OSC32K.RUNSTDBY bit is one. If OSC32K.RUNSTDBY is zero, the oscillator is disabled.
  0: The oscillator is always on, if enabled.
  1: The oscillator is enabled when a peripheral is requesting the oscillator to be used as a clock source. The oscillator is disabled if no peripheral is requesting the clock source.

- **Bit 6 – RUNSTDBY: Run in Standby**
  This bit controls how the OSC32K behaves during standby sleep mode:
  0: The oscillator is disabled in standby sleep mode.
  1: The oscillator is not stopped in standby sleep mode. If OSC32K.ONDEMAND is one, the clock source will be running when a peripheral is requesting the clock. If OSC32K.ONDEMAND is zero, the clock source will always be running in standby sleep mode.

- **Bits 5:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – EN1K: 1kHz Output Enable**
  0: The 1kHz output is disabled.
  1: The 1kHz output is enabled.
- **Bit 2 – EN32K: 32kHz Output Enable**
  0: The 32kHz output is disabled.
  1: The 32kHz output is enabled.
- **Bit 1 – ENABLE: Oscillator Enable**
  0: The oscillator is disabled.
  1: The oscillator is enabled.
- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

16.8.8 32kHz Ultra Low Power Internal Oscillator (OSCULP32K) Control

Name: OSCULP32K

Offset: 0x1C

Reset: 0X000XXXXX

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	WRTLOCK			CALIB[4:0]				
Access	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	X	X	X	X	X

- **Bit 7 – WRTLOCK: Write Lock**
  This bit locks the OSCULP32K register for futur writes to fix the OSCULP32K configuration.
  0: The OSCULP32K configuration is not locked.
  1: The OSCULP32K configuration is locked.

- **Bits 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:0 – CALIB[4:0]: Oscillator Calibration**
  These bits control the oscillator calibration.
  These bits are loaded from Flash Calibration at startup.

16.8.9 8MHz Internal Oscillator (OSC8M) Control

Name: OSC8M

Offset: 0x20

Reset: 0x87070382

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	FRANGE[1:0]				CALIB[11:8]			
Access	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	1	1	1

Bit	23	22	21	20	19	18	17	16
	CALIB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	1

Bit	15	14	13	12	11	10	9	8
							PRESC[1:0]	
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	1	1

Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY					ENABLE	
Access	R/W	R/W	R	R	R	R	R/W	R
Reset	1	0	0	0	0	0	1	0

- **Bits 31:30 – FRANGE[1:0]: Oscillator Frequency Range**
  These bits control the oscillator frequency range according to the table below. These bits are loaded from Flash Calibration at startup.

Table 16-12. Oscillator Frequency Range

FRANGE[1:0]	Name	Description
0x0	0	4 to 6MHz
0x1	1	6 to 8MHz
0x2	2	8 to 11MHz
0x3	3	11 to 15MHz

- **Bits 29:28 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 27:16 – CALIB[11:0]: Oscillator Calibration**
  These bits control the oscillator calibration. The calibration field is split in two:
  CALIB[11:6] is for temperature calibration
  CALIB[5:0] is for overall process calibration
  These bits are loaded from Flash Calibration at startup.

- **Bits 15:10 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 9:8 – PRESC[1:0]: Oscillator Prescaler**
  These bits select the oscillator prescaler factor setting according to the table below.

Table 16-13. Oscillator Prescaler

PRESC[1:0]	Name	Description
0x0	0	1
0x1	1	2
0x2	2	4
0x3	3	8

- **Bit 7 – ONDEMAND: On Demand Control**
  The On Demand operation mode allows an oscillator to be enabled or disabled depending on peripheral clock requests.
  In On Demand operation mode, i.e., if the ONDEMAND bit has been previously written to one, the oscillator will only be running when requested by a peripheral. If there is no peripheral requesting the oscillator s clock source, the oscillator will be in a disabled state.
  If On Demand is disabled the oscillator will always be running when enabled.
  In standby sleep mode, the On Demand operation is still active if the OSC8M.RUNSTDBY bit is one. If OSC8M.RUNSTDBY is zero, the oscillator is disabled.
  0: The oscillator is always on, if enabled.
  1: The oscillator is enabled when a peripheral is requesting the oscillator to be used as a clock source. The oscillator is disabled if no peripheral is requesting the clock source.

- **Bit 6 – RUNSTDBY: Run in Standby**
  This bit controls how the OSC8M behaves during standby sleep mode:
  0: The oscillator is disabled in standby sleep mode.
  1: The oscillator is not stopped in standby sleep mode. If OSC8M.ONDEMAND is one, the clock source will be running when a peripheral is requesting the clock. If OSC8M.ONDEMAND is zero, the clock source will always be running in standby sleep mode.

- **Bits 5:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – ENABLE: Oscillator Enable**
  0: The oscillator is disabled or being enabled.
  1: The oscillator is enabled or being disabled.
  The user must ensure that the OSC8M is fully disabled before enabling it, and that the OSC8M is fully enabled before disabling it by reading OSC8M.ENABLE.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

16.8.10 DFLL48M Control

Name: DFLLCTRL

Offset: 0x24

Reset: 0x0080

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
					WAITLOCK	BPLCKC	QLDIS	CCDIS
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ONDEMAND			LLAW	STABLE	MODE	ENABLE	
Access	R/W	R	R	R/W	R/W	R/W	R/W	R
Reset	1	0	0	0	0	0	0	0

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 11 – WAITLOCK: Wait Lock**
  This bit controls the DFLL output clock, depending on lock status:
  0: Output clock before the DFLL is locked.
  1: Output clock when DFLL is locked.

- **Bit 10 – BPLCKC: Bypass Coarse Lock**
  This bit controls the coarse lock procedure:
  0: Bypass coarse lock is disabled.
  1: Bypass coarse lock is enabled.

- **Bit 9 – QLDIS: Quick Lock Disable**
  0: Quick Lock is enabled.
  1: Quick Lock is disabled.

- **Bit 8 – CCDIS: Chill Cycle Disable**
  0: Chill Cycle is enabled.
  1: Chill Cycle is disabled.

- **Bit 7 – ONDEMAND: On Demand Control**
  The On Demand operation mode allows an oscillator to be enabled or disabled depending on peripheral clock requests.

  In On Demand operation mode, i.e., if the ONDEMAND bit has been previously written to one, the oscillator will only be running when requested by a peripheral. If there is no peripheral requesting the oscillator s clock source, the oscillator will be in a disabled state.

  If On Demand is disabled the oscillator will always be running when enabled.

  In standby sleep mode, the On Demand operation is still active if the DFLLCTRL.RUNSTDBY bit is one. If DFLLCTRL.RUNSTDBY is zero, the oscillator is disabled.

0: The oscillator is always on, if enabled.

1: The oscillator is enabled when a peripheral is requesting the oscillator to be used as a clock source. The oscillator is disabled if no peripheral is requesting the clock source.

- **Bit 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 4 – LLAW: Lose Lock After Wake**
  0: Locks will not be lost after waking up from sleep modes if the DFLL clock has been stopped.

  1: Locks will be lost after waking up from sleep modes if the DFLL clock has been stopped.

- **Bit 3 – STABLE: Stable DFLL Frequency**
  0: FINE calibration tracks changes in output frequency.

  1: FINE calibration register value will be fixed after a fine lock.

- **Bit 2 – MODE: Operating Mode Selection**
  0: The DFLL operates in open-loop operation.

  1: The DFLL operates in closed-loop operation.

- **Bit 1 – ENABLE: DFLL Enable**
  0: The DFLL oscillator is disabled.

  1: The DFLL oscillator is enabled.

  Due to synchronization, there is delay from updating the register until the peripheral is enabled/disabled. The value written to DFLLCTRL.ENABLE will read back immediately after written.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

16.8.11 DFLL48M Value

Name: DFLLVAL

Offset: 0x28

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	DIFF[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIFF[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COARSE[5:0]						FINE[9:8]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FINE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:16 – DIFF[15:0]: Multiplication Ratio Difference**
  In closed-loop mode (DFLLCTRL.MODE is written to one), this bit group indicates the difference between the ideal number of DFLL cycles and the counted number of cycles. This value is not updated in open-loop mode, and should be considered invalid in that case.

- **Bits 15:10 – COARSE[5:0]: Coarse Value**
  Set the value of the Coarse Calibration register. In closed-loop mode, this field is read-only.

- **Bits 9:0 – FINE[9:0]: Fine Value**
  Set the value of the Fine Calibration register. In closed-loop mode, this field is read-only.

16.8.12 DFLL48M Multiplier

Name: DFLLMUL

Offset: 0x2C

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	CSTEP[5:0]						FSTEP[9:8]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FSTEP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MUL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MUL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:26 – CSTEP[5:0]: Coarse Maximum Step**
  This bit group indicates the maximum step size allowed during coarse adjustment in closed-loop mode. When adjusting to a new frequency, the expected output frequency overshoot depends on this step size.

- **Bits 25:16 – FSTEP[9:0]: Fine Maximum Step**
  This bit group indicates the maximum step size allowed during fine adjustment in closed-loop mode. When adjusting to a new frequency, the expected output frequency overshoot depends on this step size.

- **Bits 15:0 – MUL[15:0]: DFLL Multiply Factor**
  This field determines the ratio of the CLK_DFLL output frequency to the CLK_DFLL_REF input frequency. Writing to the MUL bits will cause locks to be lost and the fine calibration value to be reset to its midpoint.

16.8.13 DFLL48M Synchronization

Name: DFLLSYNC

Offset: 0x30

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	READREQ							
Access	W	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – READREQ: Read Request**
  To be able to read the current value of DFLLVAL in closed-loop mode, this bit should be written to one. The updated value is available in DFLLVAL when PCLKSR.DFLLRDY is set.

- **Bits 6:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

16.8.14 3.3V Brown-Out Detector (BOD33) Control

Name: BOD33

Offset: 0x34

Reset: 0X0000000000XXXXXX00000000000XXXX0

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			LEVEL[5:0]					
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	X	X	X	X	X	X

Bit	15	14	13	12	11	10	9	8
	PSEL[3:0]						CEN	MODE
Access	R/W	R/W	R/W	R/W	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY		ACTION[1:0]		HYST	ENABLE	
Access	R	R/W	R	R/W	R/W	R/W	R/W	R
Reset	0	0	0	X	X	X	X	0

- **Bits 31:22 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 21:16 – LEVEL[5:0]: BOD33 Threshold Level**
  This field sets the triggering voltage threshold for the BOD33. See the “Electrical Characteristics” on page 648 for actual voltage levels. Note that any change to the LEVEL field of the BOD33 register should be done when the BOD33 is disabled in order to avoid spurious resets or interrupts.

  These bits are loaded from Flash User Row at startup. Refer to “NVM User Row Mapping” on page 20 for more details.

- **Bits 15:12 – PSEL[3:0]: Prescaler Select**
  Selects the prescaler divide-by output for the BOD33 sampling mode according to the table below. The input clock comes from the OSCULP32K 1kHz output.

Table 16-14. Prescaler Select

PSEL[3:0]	Name	Description
0x0	DIV2	Divide clock by 2
0x1	DIV4	Divide clock by 4
0x2	DIV8	Divide clock by 8
0x3	DIV16	Divide clock by 16
0x4	DIV32	Divide clock by 32
0x5	DIV64	Divide clock by 64
0x6	DIV128	Divide clock by 128
0x7	DIV256	Divide clock by 256
0x8	DIV512	Divide clock by 512
0x9	DIV1K	Divide clock by 1024
0xA	DIV2K	Divide clock by 2048
0xB	DIV4K	Divide clock by 4096
0xC	DIV8K	Divide clock by 8192
0xD	DIV16K	Divide clock by 16384
0xE	DIV32K	Divide clock by 32768
0xF	DIV64K	Divide clock by 65536

- **Bits 11:10 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 9 – CEN: Clock Enable**
  0: The BOD33 sampling clock is either disabled and stopped, or enabled but not yet stable.
  1: The BOD33 sampling clock is either enabled and stable, or disabled but not yet stopped.
  Writing a zero to this bit will stop the BOD33 sampling clock.
  Writing a one to this bit will start the BOD33 sampling clock.
- **Bit 8 – MODE: Operation Mode**
  0: The BOD33 operates in continuous mode.
  1: The BOD33 operates in sampling mode.
- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bit 6 – RUNSTDBY: Run in Standby**
  0: The BOD33 is disabled in standby sleep mode.
  1: The BOD33 is enabled in standby sleep mode.
- **Bit 5 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 4:3 – ACTION[1:0]: BOD33 Action**
  These bits are used to select the BOD33 action when the supply voltage crosses below the BOD33 threshold.
  These bits are loaded from Flash User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.

Table 16-15. BOD33 Action

ACTION[1:0]	Name	Description
0x0	NONE	No action
0x1	RESET	The BOD33 generates a reset
0x2	INTERRUPT	The BOD33 generates an interrupt
0x3		Reserved

- **Bit 2 – HYST: Hysteresis**
  This bit indicates whether hysteresis is enabled for the BOD33 threshold voltage:
  0: No hysteresis.
  1: Hysteresis enabled.
  This bit is loaded from Flash User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.
- **Bit 1 – ENABLE: Enable**
  0: BOD33 is disabled.
  1: BOD33 is enabled.
  This bit is loaded from Flash User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.
- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

16.8.15 Voltage References System (VREF) Control

Name: VREF

Offset: 0x40

Reset: 0X00000XXXXXXXXXXX0000000000000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
						CALIB[10:8]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	X	X	X

Bit	23	22	21	20	19	18	17	16
	CALIB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	X	X	X	X	X	X	X	X

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						BGOUTEN	TSEN	
Access	R	R	R	R	R	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:27 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 26:16 – CALIB[10:0]: Bandgap Voltage Generator Calibration**
  These bits are used to calibrate the output level of the bandgap voltage reference. These bits are loaded from Flash Calibration Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.

- **Bits 15:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – BGOUTEN: Bandgap Output Enable**
  0: The bandgap output is not available as an ADC input channel.
  1: The bandgap output is routed to an ADC input channel.

- **Bit 1 – TSEN: Temperature Sensor Enable**
  0: Temperature sensor is disabled.

1: Temperature sensor is enabled and routed to an ADC input channel.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

16.8.16 DPLL Control A

Name: DPLLCTRLA

Offset: 0x44

Reset: 0x80

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ONDEMAND						ENABLE	
Access	R/W	R	R	R	R	R	R/W	R
Reset	1	0	0	0	0	0	0	0

- **Bit 7 – ONDEMAND: On Demand Clock Activation**
  0: The DPLL is always on when enabled.
  1: The DPLL is activated only when a peripheral request the DPLL as a source clock. The DPLLCTRLA.ENABLE bit must be one to validate that operation, otherwise the peripheral request has no effect.

- **Bits 6:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – ENABLE: DPLL Enable**
  0: The DPLL is disabled.
  1: The DPLL is enabled.
  The software operation of enabling or disabling the DPLL takes a few clock cycles, so check the DPLLSTATUS.ENABLE status bit to identify when the DPLL is successfully activated or disabled.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

16.8.17 DPLL Ratio Control

Name: DPLLRATIO

Offset: 0x48

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
					LDRFRAC[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
					LDR[11:8]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:20 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 19:16 – LDRFRAC[3:0]: Loop Divider Ratio Fractional Part**
  Write this field with the fractional part of the frequency multiplier.

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:0 – LDR[11:0]: Loop Divider Ratio**
  Write this field with the integer part of the frequency multiplier.

16.8.18 DPLL Control B

Name: DPLLCTRLB

Offset: 0x4C

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
						DIV[10:8]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				LBYPASS		LTIME[2:0]		
Access	R	R	R	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			REFCLK[1:0]		WUF	LPEN	FILTER[1:0]	
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:27 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 26:16 – DIV[10:0]: Clock Divider**
  These bits are used to set the CLK_DPLL_REF1 clock division factor and can be calculated with the following formula:

  f_div = f_CLKDPLLREF1/(2x(DIV+1))

- **Bits 15:13 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 12 – LBYPASS: Lock Bypass**
  0: Normal Mode: the CK_GATED is turned off when lock signal is low.

  1: Lock Bypass Mode: the CK_GATED is always running, lock is irrelevant.

- **Bit 11 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 10:8 – LTIME[2:0]: Lock Time**
  These bits select the DPLL lock timeout.

Table 16-16. Lock Time

LTIME[2:0]	Name	Description
0x0	DEFAULT	No time-out
0x1 - 0x3		Reserved
0x4	8MS	Time-out if no lock within 8ms
0x5	9MS	Time-out if no lock within 9ms
0x6	10MS	Time-out if no lock within 10ms
0x7	11MS	Time-out if no lock within 11ms

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:4 – REFCLK[1:0]: Reference Clock Selection**
  These bits select the DPLL clock reference.

Table 16-17. Reference Clock Selection

REFCLK[2:0]	Name	Description
0x0	REF0	CLK_DPLL_REF0 clock reference
0x1	REF1	CLK_DPLL_REF1 clock reference
0x2	GCLK	GCLK_DPLL clock reference
0x3		Reserved

- **Bit 3 – WUF: Wake Up Fast**
  0: DPLL CK output is gated until complete startup time and lock time.
  1: DPLL CK output is gated until startup time only.

- **Bit 2 – LPEN: Low-Power Enable**
  0: The time to digital converter is selected.
  1: The time to digital converter is not selected, this will improve power consumption but increase the output jitter.

- **Bits 1:0 – FILTER[1:0]: Proportional Integral Filter Selection**
  These bits select the DPLL filter type.

Table 16-18. Proportional Integral Filter Selection

FILTER[1:0]	Name	Description
0x0	DEFAULT	Default filter mode
0x1	LBFILT	Low bandwidth filter
0x2	HBFILT	High bandwidth filter
0x3	HDFILT	High damping filter

16.8.19 DPLL Status

Name: DPLLSTATUS

Offset: 0x50

Reset: 0x00

Access: Read-Only

Property: -

Bit	7	6	5	4	3	2	1	0
					DIV	ENABLE	CLKRDY	LOCK
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – DIV: Divider Enable**
  0: The reference clock divider is disabled.
  1: The reference clock divider is enabled.

- **Bit 2 – ENABLE: DPLL Enable**
  0: The DPLL is disabled.
  1: The DPLL is enabled.

- **Bit 1 – CLKRDY: Output Clock Ready**
  0: The DPLL output clock is off
  1: The DPLL output clock in on.

- **Bit 0 – LOCK: DPLL Lock Status**
  0: The DPLL Lock signal is cleared.
  1: The DPLL Lock signal is asserted.

17. WDT – Watchdog Timer

17.1 Overview

The Watchdog Timer (WDT) is a system function for monitoring correct program operation. It makes it possible to recover from error situations such as runaway or deadlocked code. The WDT is configured to a predefined time-out period, and is constantly running when enabled. If the WDT is not cleared within the time-out period, it will issue a system reset. An early-warning interrupt is available to indicate an upcoming watchdog time-out condition.

The window mode makes it possible to define a time slot (or window) inside the total time-out period during which the WDT must be cleared. If the WDT is cleared outside this window, either too early or too late, a system reset will be issued. Compared to the normal mode, this can also catch situations where a code error causes the WDT to be cleared frequently.

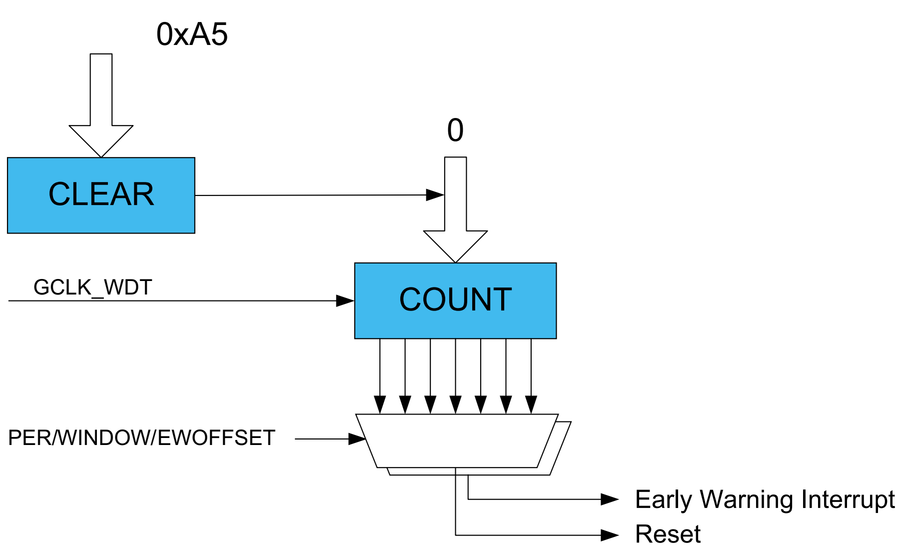
When enabled, the WDT will run in active mode and all sleep modes. It is asynchronous and runs from a CPU-independent clock source.The WDT will continue operation and issue a system reset or interrupt even if the main clocks fail.

17.2 Features

- Issues a system reset if the Watchdog Timer is not cleared before its time-out period
- Early Warning interrupt generation
- Asynchronous operation from dedicated oscillator
- Two types of operation:
  - Normal mode
  - Window mode
- Selectable time-out periods, from 8 cycles to 16,000 cycles in normal mode or 16 cycles to 32,000 cycles in window mode
- Always-on capability

17.3 Block Diagram

Figure 17-1. WDT Block Diagram



17.4 Signal Description

Not applicable.

17.5 Product Dependencies

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

17.5.1 I/O Lines

Not applicable.

17.5.2 Power Management

The WDT can continue to operate in any sleep mode where the selected source clock is running. The WDT interrupts can be used to wake up the device from sleep modes. The events can trigger other operations in the system without exiting sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

17.5.3 Clocks

The WDT bus clock (CLK_WDT_APB) is enabled by default, and can be enabled and disabled in the Power Manager. Refer to "PM – Power Manager" on page 107 for details.

A generic clock (GCLK_WDT) is required to clock the WDT. This clock must be configured and enabled in the Generic Clock Controller before using the WDT. Refer to "GCLK – Generic Clock Controller" on page 85 for details.

This generic clock is asynchronous to the user interface clock (CLK_WDT_APB). Due to this asynchronicity, accessing certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 211 for further details.

GCLK_WDT is intended to be sourced from the clock of the internal ultra-low-power (ULP) oscillator. Due to the ultra-low-power design, the oscillator is not very accurate, and so the exact time-out period may vary from device to device. This variation must be kept in mind when designing software that uses the WDT to ensure that the time-out periods used are valid for all devices. For more information on ULP oscillator accuracy, consult the "Ultra Low Power Internal 32kHz RC Oscillator (OSCULP32K) Characteristics" on page 676.

GCLK_WDT can also be clocked from other sources if a more accurate clock is needed, but at the cost of higher power consumption.

17.5.4 DMA

Not applicable.

17.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the WDT interrupts requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

17.5.6 Events

Not applicable.

17.5.7 Debug Operation

When the CPU is halted in debug mode, the WDT will halt normal operation. This peripheral can be forced to continue operation during debugging.

17.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the peripheral access controller (PAC), except the following registers:

- Interrupt Flag Status and Clear register (INTFLAG)

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode, all write-protection is automatically disabled.

Write-protection does not apply for accesses through an external debugger. Refer to “PAC – Peripheral Access Controller” on page 27 for details.

17.5.9 Analog Connections

Not applicable.

17.6 Functional Description

17.6.1 Principle of Operation

The Watchdog Timer (WDT) is a system for monitoring correct program operation, making it possible to recover from error situations such as runaway code by issuing a reset. When enabled, the WDT is a constantly running timer that is configured to a predefined time-out period. Before the end of the time-out period, the WDT should be reconfigured.

The WDT has two modes of operation, normal and window. Additionally, the user can enable Early Warning interrupt generation in each of the modes. The description for each of the basic modes is given below. The settings in the Control register (CTRL) and the Interrupt Enable register (INTENCLR/SET - refer to INTENCLR) determine the mode of operation, as illustrated in Table 17-1.

Table 17-1. WDT Operating Modes

CTRL.ENABLE	CTRL.WEN	INTENSET.EW	Mode
0	x	x	Stopped
1	0	0	Normal
1	0	1	Normal with Early Warning interrupt
1	1	0	Window
1	1	1	Window with Early Warning interrupt

17.6.2 Basic Operation

17.6.2.1 Initialization

The following bits are enable-protected:

- Window Mode Enable in the Control register (CTRL.WEN)
- Always-On in the Control register (CTRL.ALWAYSON)

The following registers are enable-protected:

- Control register (CTRL), except the Enable bit (CTRL.ENABLE)
- Configuration register (CONFIG)
- Early Warning Interrupt Control register (EWCTRL)

Any writes to these bits or registers when the WDT is enabled or is being enabled (CTRL.ENABLE is one) will be discarded. Writes to these registers while the WDT is being disabled will be completed after the disabling is complete.

Enable-protection is denoted by the Enable-Protected property in the register description.

Initialization of the WDT can be done only while the WDT is disabled.

Normal Mode

- Defining the required Time-Out Period bits in the Configuration register (CONFIG.PER).

Normal Mode with Early Warning interrupt

- Defining the required Time-Out Period bits in the Configuration register (CONFIG.PER).
- Defining Early Warning Interrupt Time Offset bits in the Early Warning Interrupt Control register (EWCTRL. EWOFFSET).
- Setting Early Warning Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.EW).

Window Mode

- Defining Time-Out Period bits in the Configuration register (CONFIG.PER).
- Defining Window Mode Time-Out Period bits in the Configuration register (CONFIG.WINDOW).
- Setting Window Enable bit in the Control register (CTRL.WEN).

Window Mode with Early Warning interrupt

- Defining Time-Out Period bits in the Configuration register (CONFIG.PER).
- Defining Window Mode Time-Out Period bits in the Configuration register (CONFIG.WINDOW).
- Setting Window Enable bit in the Control register (CTRL.WEN).
- Defining Early Warning Interrupt Time Offset bits in the Early Warning Interrupt Control register (EWCTRL. EWOFFSET).
- Setting Early Warning Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.EW).

17.6.2.2 Configurable Reset Values

On a power-on reset, some registers will be loaded with initial values from the NVM User Row. Refer to "NVM User Row Mapping" on page 20 for more details.

This encompasses the following bits and bit groups:

- Enable bit in the Control register (CTRL.ENABLE)
- Always-On bit in the Control register (CTRL.ALWAYSON)
- Watchdog Timer Windows Mode Enable bit in the Control register (CTRL.WEN)
- Watchdog Timer Windows Mode Time-Out Period bits in the Configuration register (CONFIG.WINDOW)
- Time-Out Period in the Configuration register (CONFIG.PER)
- Early Warning Interrupt Time Offset bits in the Early Warning Interrupt Control register (EWCTRL.EWOFFSET)

For more information about fuse locations, see "NVM User Row Mapping" on page 20.

17.6.2.3 Enabling and Disabling

The WDT is enabled by writing a one to the Enable bit in the Control register (CTRL.ENABLE). The WDT is disabled by writing a zero to CTRL.ENABLE.

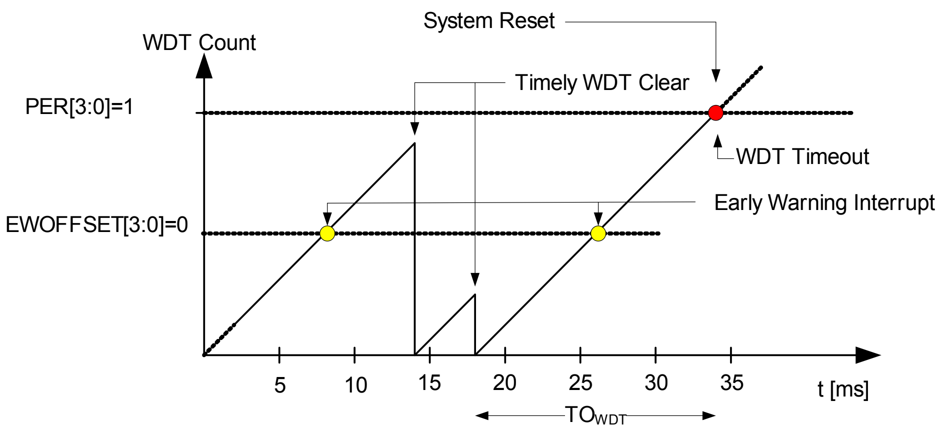
The WDT can be disabled only while the Always-On bit in the Control register (CTRL.ALWAYSON) is zero.

17.6.2.4 Normal Mode

In normal-mode operation, the length of a time-out period is configured in CONFIG.PER. The WDT is enabled by writing a one to the Enable bit in the Control register (CTRL.ENABLE). Once enabled, if the WDT is not cleared from the application code before the time-out occurs, the WDT will issue a system reset. There are 12 possible WDT time-out (TO_{WDT}) periods, selectable from 8ms to 16s, and the WDT can be cleared at any time during the time-out period. A new WDT time-out period will be started each time the WDT is cleared by writing 0xA5 to the Clear register (CLEAR). Writing any value other than 0xA5 to CLEAR will issue an immediate system reset.

By default, WDT issues a system reset upon a time-out, and the early warning interrupt is disabled. If an early warning interrupt is required, the Early Warning Interrupt Enable bit in the Interrupt Enable register (INTENSET.EW) must be enabled. Writing a one to the Early Warning Interrupt bit in the Interrupt Enable Set register (INTENSET.EW) enables the interrupt, and writing a one to the Early Warning Interrupt bit in the Interrupt Enable Clear register (INTENCLR.EW) disables the interrupt. If the Early Warning Interrupt is enabled, an interrupt is generated prior to a watchdog time-out condition. In normal mode, the Early Warning Offset bits in the Early Warning Interrupt Control register (EWCTRL.EWOFFSET) define the time where the early warning interrupt occurs. The normal-mode operation is illustrated in Figure 17-2.

Figure 17-2. Normal-Mode Operation



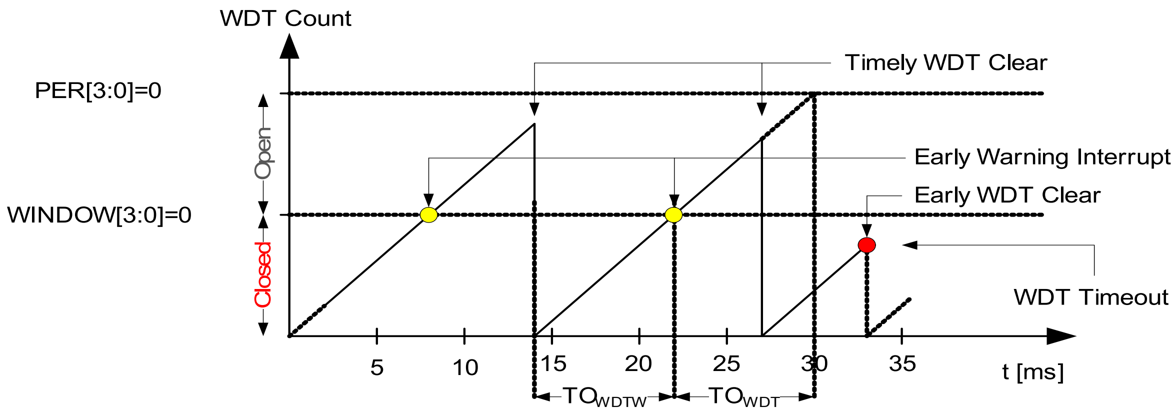
17.6.2.5 Window Mode

In window-mode operation, the WDT uses two different time-out periods, a closed window time-out period (TO_{WDTW}) and the normal, or open, time-out period (TO_{WDT}). The closed window time-out period defines a duration from 8ms to 16s where the WDT cannot be reset. If the WDT is cleared during this period, the WDT will issue a system reset. The normal WDT time-out period, which is also from 8ms to 16s, defines the duration of the open period during which the WDT can be cleared. The open period will always follow the closed period, and so the total duration of the time-out period is the sum of the closed window and the open window time-out periods. The closed window is defined by the Window Period bits in the Configuration register (CONFIG.WINDOW), and the open window is defined by the Period bits in the Configuration register (CONFIG.PER).

By default, the WDT issues a system reset upon a time-out and the Early Warning interrupt is disabled. If an Early Warning interrupt is required, INTENCLR/SET.EW must be set. Writing a one to INTENSET.EW enables the interrupt, and writing a one to INTENCLR.EW disables the interrupt. If the Early Warning interrupt is enabled in window mode, the interrupt is generated at the start of the open window period.

The window mode operation is illustrated in Figure 17-3.

Figure 17-3. Window-Mode Operation



17.6.3 Additional Features

17.6.3.1 Always-On Mode

The always-on mode is enabled by writing a one to the Always-On bit in the Control register (CTRL.ALWAYSON). When the always-on mode is enabled, the WDT runs continuously, regardless of the state of CTRL.ENABLE. Once written, the Always-On bit can only be cleared by a power-on reset. The Configuration (CONFIG) and Early Warning Control (EWCTRL) registers are read-only registers while the CTRL.ALWAYSON bit is set. Thus, the time period configuration bits (CONFIG.PER, CONFIG.WINDOW, EWCTRL.EWOFFSET) of the WDT cannot be changed.

Enabling or disabling window-mode operation by writing the Window Enable bit (CTRL.WEN) is allowed while in the always-on mode, but note that CONFIG.PER cannot be changed.

The Interrupt Clear and Interrupt Set registers are accessible in the always-on mode. The Early Warning interrupt can still be enabled or disabled while in the always-on mode, but note that EWCTRL.EWOFFSET cannot be changed.

Table 17-2 shows the operation of the WDT when CTRL.ALWAYSON is set.

Table 17-2. WDT Operating Modes With Always-On

CTRL.WEN	INTENSET.EW	Mode
0	0	Always-on and normal mode
0	1	Always-on and normal mode with Early Warning interrupt
1	0	Always-on and window mode
1	1	Always-on and window mode with Early Warning interrupt

17.6.4 Interrupts

The WDT has the following interrupt sources:

- Early Warning

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the WDT is reset. See INTFLAG for details on how to clear interrupt flags.

The WDT has one common interrupt request line for all the interrupt sources. The user must read INTFLAG to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

The Early Warning interrupt behaves differently in normal mode and in window mode. In normal mode, the Early Warning interrupt generation is defined by the Early Warning Offset in the Early Warning Control register (EWCTRL.EWOFFSET). The Early Warning Offset bits define the number of GCLK_WDT clocks before the interrupt is generated, relative to the start of the watchdog time-out period. For example, if the WDT is operating in normal mode with CONFIG.PER = 0x2 and EWCTRL.EWOFFSET = 0x1, the Early Warning interrupt is generated 16 GCLK_WDT clock cycles from the start of the watchdog time-out period, and the watchdog time-out system reset is generated 32 GCLK_WDT clock cycles from the start of the watchdog time-out period. The user must take caution when programming the Early Warning Offset bits. If these bits define an Early Warning interrupt generation time greater than the watchdog time-out period, the watchdog time-out system reset is generated prior to the Early Warning interrupt. Thus, the Early Warning interrupt will never be generated.

In window mode, the Early Warning interrupt is generated at the start of the open window period. In a typical application where the system is in sleep mode, it can use this interrupt to wake up and clear the Watchdog Timer, after which the system can perform other tasks or return to sleep mode.

17.6.5 Synchronization

Due to the asynchronicity between CLK_WDT_APB and GCLK_WDT some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set immediately, and cleared when synchronization is complete. The synchronization Ready interrupt can be used to signal when sync is complete. This can be accessed via the Synchronization Ready Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.SYNCRDY).

If an operation that requires synchronization is executed while STATUS.SYNCBUSY is one, the bus will be stalled. All operations will complete successfully, but the CPU will be stalled and interrupts will be pending as long as the bus is stalled.

The following registers need synchronization when written:

- Control register (CTRL)
- Clear register (CLEAR)

Write-synchronization is denoted by the Write-Synchronized property in the register description.

17.7 Register Summary

Table 17-3. Register Summary

Offset	Name	Bit Pos.								
0x0	CTRL	7:0	ALWAYSON					WEN	ENABLE	
0x1	CONFIG	7:0	WINDOW[3:0]				PER[3:0]			
0x2	EWCTRL	7:0					EWOFFSET[3:0]			
0x3	Reserved									
0x4	INTENCLR	7:0								EW
0x5	INTENSET	7:0								EW
0x6	INTFLAG	7:0								EW
0x7	STATUS	7:0	SYNCBUSY							
0x8	CLEAR	7:0	CLEAR[7:0]							

17.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Please refer to “Register Access Protection” on page 206 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Write-Synchronized or the Read-Synchronized property in each individual register description. Please refer to “Synchronization” on page 211 for details.

Some registers are enable-protected, meaning they can be written only when the WDT is disabled. Enable-protection is denoted by the Enable-Protected property in each individual register description.

17.8.1 Control

Name: CTRL

Offset: 0x0

Reset: 0xXX

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
	ALWAYSON					WEN	ENABLE	
Access	R/W	R	R	R	R	R/W	R/W	R
Reset	X	0	0	0	0	X	X	0

- **Bit 7 – ALWAYSON: Always-On**

  This bit allows the WDT to run continuously. After being written to one, this bit cannot be written to zero, and the WDT will remain enabled until a power-on reset is received. When this bit is one, the Control register (CTRL), the Configuration register (CONFIG) and the Early Warning Control register (EWCTRL) will be read-only, and any writes to these registers are not allowed. Writing a zero to this bit has no effect.

  0: The WDT is enabled and disabled through the ENABLE bit.

  1: The WDT is enabled and can only be disabled by a power-on reset (POR).

  This bit is not enable-protected.

  This bit is loaded from NVM User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.

- **Bits 6:3 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – WEN: Watchdog Timer Window Mode Enable**

  This bit enables window mode.

  This bit can only be written when CTRL.ENABLE is zero or CTRL.ALWAYSON is one:

  When CTRL.ALWAYSON=0, this bit is enable-protected by CTRL.ENABLE.

  When CTRL.ALWAYSON=1 this bit is not enable-protected by CTRL.ENABLE.

  The initial value of this bit is loaded from Flash Calibration.

  The initial value of this bit is loaded from Flash Calibration.

  0: Window mode is disabled (normal operation).

  1: Window mode is enabled.

  This bit is loaded from NVM User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.

- **Bit 1 – ENABLE: Enable**

  This bit enables or disables the WDT. Can only be written while CTRL.ALWAYSON is zero.

  0: The WDT is disabled.

  1: The WDT is enabled.

  Due to synchronization, there is delay from writing CTRL.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately, and the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set. STATUS.SYNCBUSY will be cleared when the operation is complete.

  This bit is not enable-protected.

  This bit is loaded from NVM User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

17.8.2 Configuration

Name: CONFIG

Offset: 0x1

Reset: 0xXX

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
	WINDOW[3:0]				PER[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	X	X	X	X	X	X	X	X

- **Bits 7:4 – WINDOW[3:0]: Window Mode Time-Out Period**
  In window mode, these bits determine the watchdog closed window period as a number of oscillator cycles. The closed window periods are defined in Table 17-4.
  These bits are loaded from NVM User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.

Table 17-4. Window Mode Time-Out Period

WINDOW[3:0]	Name	Description
0x0	8	8 clock cycles
0x1	16	16 clock cycles
0x2	32	32 clock cycles
0x3	64	64 clock cycles
0x4	128	128 clock cycles
0x5	256	256 clock cycles
0x6	512	512 clock cycles
0x7	1K	1024 clock cycles
0x8	2K	2048 clock cycles
0x9	4K	4096 clock cycles
0xA	8K	8192 clock cycles
0xB	16K	16384 clock cycles
0xc-0xf		Reserved

- **Bits 3:0 – PER[3:0]: Time-Out Period**
  These bits determine the watchdog time-out period as a number of GCLK_WDT clock cycles. In window mode operation, these bits define the open window period. The different typical time-out periods are found in Table 17-5.
  These bits are loaded from NVM User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.

Table 17-5. Time-Out Period

PER[3:0]	Name	Description
0x0	8	8 clock cycles
0x1	16	16 clock cycles
0x2	32	32 clock cycles
0x3	64	64 clock cycles
0x4	128	128 clock cycles
0x5	256	256 clock cycles
0x6	512	512 clock cycles
0x7	1K	1024 clock cycles
0x8	2K	2048 clock cycles
0x9	4K	4096 clock cycles
0xA	8K	8192 clock cycles
0xB	16K	16384 clock cycles
0xc-0xf		Reserved

17.8.3 Early Warning Interrupt Control

Name: EWCTRL

Offset: 0x2

Reset: 0x0X

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
					EWOFFSET[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	X	X	X	X

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 3:0 – EWOFFSET[3:0]: Early Warning Interrupt Time Offset**
  These bits determine the number of GCLK_WDT clocks in the offset from the start of the watchdog time-out period to when the Early Warning interrupt is generated. The Early Warning Offset is defined in Table 17-6. These bits are loaded from NVM User Row at startup. Refer to "NVM User Row Mapping" on page 20 for more details.

Table 17-6. Early Warning Interrupt Time Offset

EWOFFSET[3:0]	Name	Description
0x0	8	8 clock cycles
0x1	16	16 clock cycles
0x2	32	32 clock cycles
0x3	64	64 clock cycles
0x4	128	128 clock cycles
0x5	256	256 clock cycles
0x6	512	512 clock cycles
0x7	1K	1024 clock cycles
0x8	2K	2048 clock cycles
0x9	4K	4096 clock cycles
0xA	8K	8192 clock cycles
0xB	16K	16384 clock cycles
0xc-0xf		Reserved

17.8.4 Interrupt Enable Clear

Name: INTENCLR

Offset: 0x4

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
								EW
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – EW: Early Warning Interrupt Enable**
  0: The Early Warning interrupt is disabled.
  1: The Early Warning interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the Early Warning interrupt.

17.8.5 Interrupt Enable Set

Name: INTENSET

Offset: 0x5

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
								EW
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – EW: Early Warning Interrupt Enable**
  0: The Early Warning interrupt is disabled.
  1: The Early Warning interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the Early Warning interrupt.

17.8.6 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x6

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
								EW
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – EW: Early Warning**
  This flag is set when an Early Warning interrupt occurs, as defined by the EWOFFSET bit group in EWCTRL.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the Early Warning interrupt flag.

17.8.7 Status

Name: STATUS

Offset: 0x7

Reset: 0x00

Access: Read-Only

Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCBUSY							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – SYNCBUSY: Synchronization Busy**
  This bit is cleared when the synchronization of registers between clock domains is complete.
  This bit is set when the synchronization of registers between clock domains is started.

- **Bits 6:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

17.8.8 Clear

Name: CLEAR

Offset: 0x8

Reset: 0x00

Access: Write-Only

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	CLEAR[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:0 – CLEAR[7:0]: Watchdog Clear**
  Writing 0xA5 to this register will clear the Watchdog Timer and the watchdog time-out period is restarted. Writing any other value will issue an immediate system reset.

Table 17-7. Watchdog Clear

CLEAR[7:0]	Name	Description
0x0-0xa4		Reserved
0xA5	KEY	Clear Key
0xa6-0xff		Reserved

18. RTC – Real-Time Counter

18.1 Overview

The Real-Time Counter (RTC) is a 32-bit counter with a 10-bit programmable prescaler that typically runs continuously to keep track of time. The RTC can wake up the device from sleep modes using the alarm/compare wake up, periodic wake up or overflow wake up mechanisms.

The RTC is typically clocked by the 1.024kHz output from the 32.768kHz High-Accuracy Internal Crystal Oscillator(OSC32K) and this is the configuration optimized for the lowest power consumption. The faster 32.768kHz output can be selected if the RTC needs a resolution higher than 1ms. The RTC can also be clocked from other sources, selectable through the Generic Clock module (GCLK).

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

The 10-bit programmable prescaler can scale down the clock source, and so a wide range of resolutions and time-out periods can be configured. With a 32.768kHz clock source, the minimum counter tick interval is 30.5µs, and time-out periods can range up to 36 hours. With the counter tick interval configured to 1s, the maximum time-out period is more than 136 years.

18.2 Features

- 32-bit counter with 10-bit prescaler
- Multiple clock sources
- 32-bit or 16-bit Counter mode
  - One 32-bit or two 16-bit compare values
- Clock/Calendar mode
  - Time in seconds, minutes and hours (12/24)
  - Date in day of month, month and year
  - Leap year correction
- Digital prescaler correction/tuning for increased accuracy
- Overflow, alarm/compare match and prescaler interrupts and events
  - Optional clear on alarm/compare match

18.3 Block Diagram

Figure 18-1. RTC Block Diagram (Mode 0 — 32-Bit Counter)

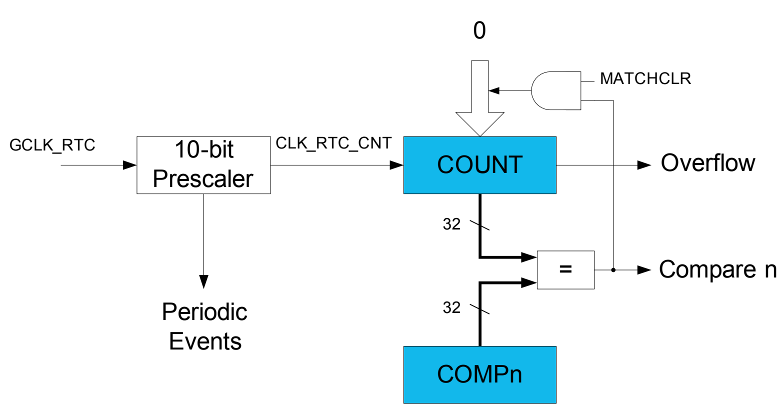


Figure 18-2. RTC Block Diagram (Mode 1 — 16-Bit Counter)

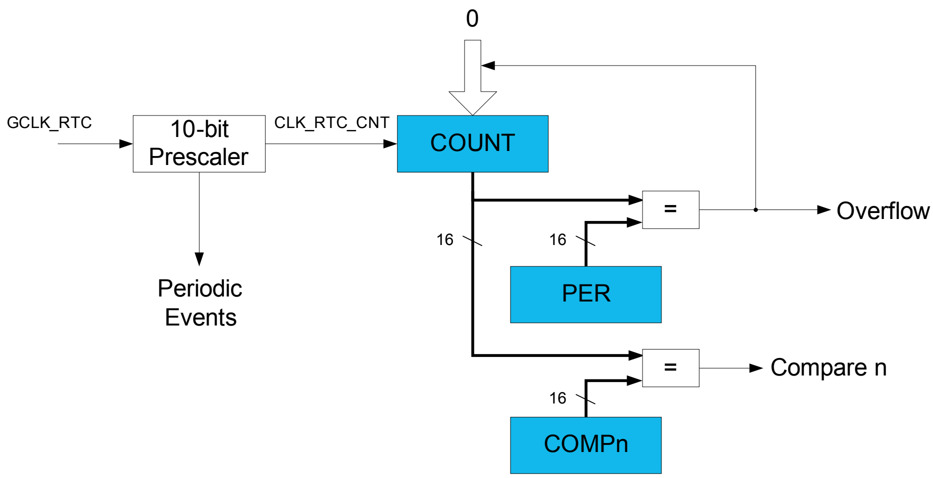
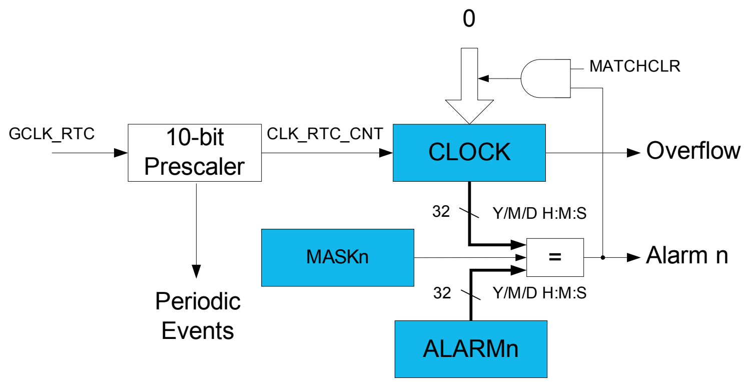


Figure 18-3. RTC Block Diagram (Mode 2 — Clock/Calendar)



18.4 Signal Description

Not applicable.

18.5 Product Dependencies

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

18.5.1 I/O Lines

Not applicable.

18.5.2 Power Management

The RTC can continue to operate in any sleep mode. The RTC interrupts can be used to wake up the device from sleep modes. The events can trigger other operations in the system without exiting sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

The RTC will be reset only at power-on (POR) or by writing a one to the Software Reset bit in the Control register (CTRL.SWRST).

18.5.3 Clocks

The RTC bus clock (CLK_RTC_APB) can be enabled and disabled in the Power Manager, and the default state of CLK_RTC_APB can be found in the Peripheral Clock Masking section in the "PM – Power Manager" on page 107.

A generic clock (GCLK_RTC) is required to clock the RTC. This clock must be configured and enabled in the Generic Clock Controller before using the RTC. Refer to "GCLK – Generic Clock Controller" on page 85 for details.

This generic clock is asynchronous to the user interface clock (CLK_RTC_APB). Due to this asynchronicity, accessing certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 231 for further details.

The RTC should never be used with the generic clock generator 0.

18.5.4 DMA

Not applicable.

18.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the RTC interrupts requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

18.5.6 Events

To use the RTC event functionality, the corresponding events need to be configured in the event system. Refer to "EVSYS – Event System" on page 402 for details.

18.5.7 Debug Operation

When the CPU is halted in debug mode the RTC will halt normal operation. The RTC can be forced to continue operation during debugging. Refer to the DBGCTRL register for details.

18.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the peripheral access controller (PAC), except the following registers:

- Interrupt Flag Status and Clear register (INTFLAG)
- Read Request register (READREQ)
- Status register (STATUS)
- Debug register (DBGCTRL)

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode, all write-protection is automatically disabled.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

18.5.9 Analog Connections

A 32.768kHz crystal can be connected to the XIN32 and XOUT32 pins, along with any required load capacitors. For details on recommended crystal characteristics and load capacitors, refer to "Electrical Characteristics" on page 648 for details.

18.6 Functional Description

18.6.1 Principle of Operation

The RTC keeps track of time in the system and enables periodic events, as well as interrupts and events at a specified time. The RTC consists of a 10-bit prescaler that feeds a 32-bit counter. The actual format of the 32-bit counter depends on the RTC operating mode.

18.6.2 Basic Operation

18.6.2.1 Initialization

The following bits are enable-protected, meaning that they can only be written when the RTC is disabled (CTRL.ENABLE is zero):

- Operating Mode bits in the Control register (CTRL.MODE)
- Prescaler bits in the Control register (CTRL.PRESCALER)
- Clear on Match bit in the Control register (CTRL.MATCHCLR)
- Clock Representation bit in the Control register (CTRL.CLKREP)

The following register is enable-protected:

- Event Control register (EVCTRL)

Any writes to these bits or registers when the RTC is enabled or being enabled (CTRL.ENABLE is one) will be discarded. Writes to these bits or registers while the RTC is being disabled will be completed after the disabling is complete.

Enable-protection is denoted by the Enable-Protection property in the register description.

Before the RTC is enabled, it must be configured, as outlined by the following steps:

- RTC operation mode must be selected by writing the Operating Mode bit group in the Control register (CTRL.MODE)
- Clock representation must be selected by writing the Clock Representation bit in the Control register (CTRL.CLKREP)
- Prescaler value must be selected by writing the Prescaler bit group in the Control register (CTRL.PRESCALER)

The RTC prescaler divides down the source clock for the RTC counter. The frequency of the RTC clock (CLK_RTC_CNT) is given by the following formula:

$$f_{\mathrm{CLK\_RTC\_CNT}} = \frac{f_{\mathrm{GCLK\_RTC}}}{2^{\mathrm{PRESCALER}}}$$

The frequency of the generic clock, GCLK_RTC, is given by f_{GCLK_RTC}, and $f_{CLK\_RTC\_CNT}$ is the frequency of the internal prescaled RTC clock, CLK_RTC_CNT.

Note that in the Clock/Calendar mode, the prescaler must be configured to provide a 1Hz clock to the counter for correct operation.

18.6.2.2 Enabling, Disabling and Resetting

The RTC is enabled by writing a one to the Enable bit in the Control register (CTRL.ENABLE). The RTC is disabled by writing a zero to CTRL.ENABLE.

The RTC should be disabled before resetting it.

The RTC is reset by writing a one to the Software Reset bit in the Control register (CTRL.SWRST). All registers in the RTC, except DBGCTRL, will be reset to their initial state, and the RTC will be disabled.

Refer to the CTRL register for details.

18.6.3 Operating Modes

The RTC counter supports three RTC operating modes: 32-bit Counter, 16-bit Counter and Clock/Calendar. The operating mode is selected by writing to the Operating Mode bit group in the Control register (CTRL.MODE).

18.6.3.1 32-Bit Counter (Mode 0)

When the RTC Operating Mode bits in the Control register (CTRL.MODE) are zero, the counter operates in 32-bit Counter mode. The block diagram of this mode is shown in Figure 18-1. When the RTC is enabled, the counter will increment on every 0-to-1 transition of CLK_RTC_CNT. The counter will increment until it reaches the top value of 0xFFFFFFFF, and then wrap to 0x00000000. This sets the Overflow interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF).

The RTC counter value can be read from or written to the Counter Value register (COUNT) in 32-bit format.

The counter value is continuously compared with the 32-bit Compare registers (COMP0n, n=0–1). When a compare match occurs, the Compare 0n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMP0n) is set on the next 0-to-1 transition of CLK_RTC_CNT.

If the Clear on Match bit in the Control register (CTRL.MATCHCLR) is one, the counter is cleared on the next counter cycle when a compare match with COMP0n occurs. This allows the RTC to generate periodic interrupts or events with longer periods than are possible with the prescaler events. Note that when CTRL.MATCHCLR is one, INTFLAG.CMP0n and INTFLAG.OVF will both be set simultaneously on a compare match with COMP0n.

18.6.3.2 16-Bit Counter (Mode 1)

When CTRL.MODE is one, the counter operates in 16-bit Counter mode as shown in Figure 18-2. When the RTC is enabled, the counter will increment on every 0-to-1 transition of CLK_RTC_CNT. In 16-bit Counter mode, the 16-bit Period register (PER) holds the maximum value of the counter. The counter will increment until it reaches the PER value, and then wrap to 0x0000. This sets the Overflow interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF).

The RTC counter value can be read from or written to the Counter Value register (COUNT) in 16-bit format.

The counter value is continuously compared with the 16-bit Compare registers (COMPn, n=0–1). When a compare match occurs, the Compare n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn, n=0–1) is set on the next 0-to-1 transition of CLK_RTC_CNT.

18.6.3.3 Clock/Calendar (Mode 2)

When CTRL.MODE is two, the counter operates in Clock/Calendar mode, as shown in Figure 18-3. When the RTC is enabled, the counter will increment on every 0-to-1 transition of CLK_RTC_CNT. The selected clock source and RTC prescaler must be configured to provide a 1Hz clock to the counter for correct operation in this mode.

The time and date can be read from or written to the Clock Value register (CLOCK) in a 32-bit time/date format. Time is represented as:

- Seconds
- Minutes
- Hours

Hours can be represented in either 12- or 24-hour format, selected by the Clock Representation bit in the Control register (CTRL.CLKREP). This bit can be changed only while the RTC is disabled.

Date is represented as:

- Day as the numeric day of the month (starting at 1)
- Month as the numeric month of the year (1 = January, 2 = February, etc.)
- Year as a value counting the offset from a reference value that must be defined in software

The date is automatically adjusted for leap years, assuming every year divisible by 4 is a leap year. Therefore, the reference value must be a leap year, e.g. 2000. The RTC will increment until it reaches the top value of 23:59:59 December 31st of year 63, and then wrap to 00:00:00 January 1st of year 0. This will set the Overflow interrupt flag in the Interrupt Flag Status and Clear registers (INTFLAG.OVF).

The clock value is continuously compared with the 32-bit Alarm registers (ALARM0n, n=0–1). When an alarm match occurs, the Alarm 0n Interrupt flag in the Interrupt Flag Status and Clear registers (INTFLAG.ALARMn0) is set on the next 0-to-1 transition of CLK_RTC_CNT. For a 1 Hz clock counter, it means the Alarm 0 Interrupt flag is set with a delay of 1 sec after the occurrence of alarm match.

A valid alarm match depends on the setting of the Alarm Mask Selection bits in the Alarm 0n Mask register (MASK0n.SEL). These bits determine which time/date fields of the clock and alarm values are valid for comparison and which are ignored.

If the Clear on Match bit in the Control register (CTRL.MATCHCLR) is one, the counter is cleared on the next counter cycle when an alarm match with ALARM0n occurs. This allows the RTC to generate periodic interrupts or events with longer periods than are possible with the prescaler events (see "Periodic Events" on page 229). Note that when CTRL.MATCHCLR is one, INTFLAG.ALARM0 and INTFLAG.OVF will both be set simultaneously on an alarm match with ALARM0n.

18.6.4 Additional Features

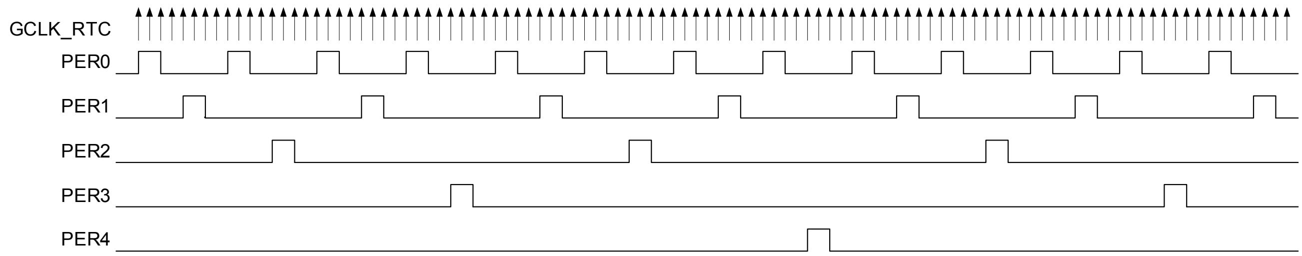
18.6.4.1 Periodic Events

The RTC prescaler can generate events at periodic intervals, allowing flexible system tick creation. Any of the upper eight bits of the prescaler (bits 2 to 9) can be the source of an event. When one of the Periodic Event Output bits in the Event Control register (EVCTRL.PEREOn) is one, an event is generated on the 0-to1 transition of the related bit in the prescaler, resulting in a periodic event frequency of:

$$f_{PERIODIC} = \frac{f_{GCLK\_RTC}}{2^{n+3}}$$

f_{GCLK_RTC} is the frequency of the internal prescaler clock, GCLK_RTC, and *n* is the position of the EVCTRL.PEREOn bit. For example, PEREO will generate an event every 8 GCLK_RTC cycles, PEREO1 every 16 cycles, etc. This is shown in Figure 18-4. Periodic events are independent of the prescaler setting used by the RTC counter, except if CTRL.PRESCALER is zero. Then, no periodic events will be generated.

Figure 18-4. Example Periodic Events



18.6.4.2 Frequency Correction

The RTC Frequency Correction module employs periodic counter corrections to compensate for a too-slow or too-fast oscillator. Frequency correction requires that CTRL.PRESCALER is greater than 1.

The digital correction circuit adds or subtracts cycles from the RTC prescaler to adjust the frequency in approximately 1PPM steps. Digital correction is achieved by adding or skipping a single count in the prescaler once every 1024 GCLK_RTC cycles. The Value bit group in the Frequency Correction register (FREQCORR.VALUE) determines the number of times the adjustment is applied over 976 of these periods. The resulting correction is as follows:

$$\text{Correction in PPM} = \frac{\text{FREQCORR.VALUE}}{1024 \cdot 976} \cdot 10^6\text{PPM}$$

This results in a resolution of 1.0006PPM.

The Sign bit in the Frequency Correction register (FREQCORR.SIGN) determines the direction of the correction. A positive value will speed up the frequency, and a negative value will slow down the frequency.

Digital correction also affects the generation of the periodic events from the prescaler. When the correction is applied at the end of the correction cycle period, the interval between the previous periodic event and the next occurrence may also be shortened or lengthened depending on the correction value.

18.6.5 DMA Operation

Not applicable.

18.6.6 Interrupts

The RTC has the following interrupt sources:

- Overflow (INTFLAG.OVF)
- Compare n (INTFLAG.CMPn)
- Alarm n (INTFLAG.ALARMn)
- Synchronization Ready (INTFLAG.SYNCRDY)

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the RTC is reset. See INTFLAG for details on how to clear interrupt flags. The RTC has one common interrupt request line for all the interrupt sources. The user must read INTFLAG to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to “Nested Vector Interrupt Controller” on page 23 for details.

18.6.7 Events

The RTC can generate the following output events, which are generated in the same way as the corresponding interrupts:

- Overflow (OVF)
- Period n (PERn)
- Compare n (CMPn)
- Alarm n (ALARMn)

Output events must be enabled to be generated. Writing a one to an Event Output bit in the Event Control register (EVCTRL.xxEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. Refer to “EVSYS – Event System” on page 402 for details.

18.6.8 Sleep Mode Operation

The RTC will continue to operate in any sleep mode where the source clock is active. The RTC interrupts can be used to wake up the device from a sleep mode, or the RTC events can trigger other operations in the system without exiting the sleep mode.

An interrupt request will be generated after the wake-up if the Interrupt Controller is configured accordingly. Otherwise the CPU will wake up directly, without triggering an interrupt. In this case, the CPU will continue executing from the instruction following the entry into sleep.

The periodic events can also wake up the CPU through the interrupt function of the Event System. In this case, the event must be enabled and connected to an event channel with its interrupt enabled. See "EVSYS – Event System" on page 402 for more information.

18.6.8.1 Shutdown Mode

Any of the RTC interrupt sources can wake up the device from the Shutdown mode if the RTC clock is configured to use a clock that is available in Shutdown mode. The wake-up sources are enabled if the corresponding bit in the Interrupt Enable registers (INTENCLR/SET) is one.

When waking up from Shutdown mode, all RTC registers will have the same value as before the Shutdown mode was entered, except the following registers: Interrupt Enable (INTENCLR/SET), Read Request (READREQ), Interrupt Flag Status and Clear (INTFLAG) and Debug (DBGCTRL). Note that INTENCLR/SET will be reset, with all interrupts turned off. The software must first reconfigure the Interrupt Controller, and then enable the interrupts again in the RTC. The Status register (STATUS) will show the status of the RTC, including the status bits set during shutdown operation.

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

18.6.9 Synchronization

Due to the asynchronicity between CLK_RTC_APB and GCLK_RTC some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set immediately, and cleared when synchronization is complete. The synchronization Ready interrupt can be used to signal when sync is complete. This can be accessed via the Synchronization Ready Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.SYNCRDY).

If an operation that requires synchronization is executed while STATUS.SYNCBUSY is one, the bus will be stalled. All operations will complete successfully, but the CPU will be stalled and interrupts will be pending as long as the bus is stalled.

The following bits need synchronization when written:

- Software Reset bit in the Control register (CTRL.SWRST)
- Enable bit in the Control register (CTRL.ENABLE)

The following registers need synchronization when written:

- The Counter Value register (COUNT)
- The Clock Value register (CLOCK)
- The Counter Period register (PER)
- The Compare n Value registers (COMPn)
- The Alarm n Value registers (ALARMn)
- The Frequency Correction register (FREQCORR)
- The Alarm n Mask register (MASKn)

Write-synchronization is denoted by the Write-Synchronization property in the register description.

The following registers need synchronization when read:

- The Counter Value register (COUNT)
- The Clock Value register (CLOCK)

Read-synchronization is denoted by the Read-Synchronization property in the register description.

18.7 Register Summary

The register mapping depends on the Operating Mode bits in the Control register (CTRL.MODE). The register summary is presented for each of the three modes.

Table 18-1. MODE0 - Mode Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0	MATCHCLR				MODE[1:0]		ENABLE	SWRST
0x01		15:8					PRESCALER[3:0]			
0x02	READREQ	7:0			ADDR[5:0]					
0x03		15:8	RREQ	RCONT						
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
0x05		15:8	OVFEO							CMPEO0
0x06	INTENCLR	7:0	OVF	SYNCRDY						CMP0
0x07	INTENSET	7:0	OVF	SYNCRDY						CMP0
0x08	INTFLAG	7:0	OVF	SYNCRDY						CMP0
0x09	Reserved									
0x0A	STATUS	7:0	SYNCBUSY							
0x0B	DBGCTRL	7:0								DBGRUN
0x0C	FREQCORR	7:0	SIGN	VALUE[6:0]						
0x0D ... 0x0F	Reserved									
0x10	COUNT	7:0	COUNT[7:0]							
0x11		15:8	COUNT[15:8]							
0x12		23:16	COUNT[23:16]							
0x13		31:24	COUNT[31:24]							
0x14 ... 0x17	Reserved									
0x18	COMP0	7:0	COMP[7:0]							
0x19		15:8	COMP[15:8]							
0x1A		23:16	COMP[23:16]							
0x1B		31:24	COMP[31:24]							

Table 18-2. MODE1 - Mode Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0					MODE[1:0]		ENABLE	SWRST
0x01		15:8					PRESCALER[3:0]			
0x02	READREQ	7:0			ADDR[5:0]					
0x03		15:8	RREQ	RCONT						
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
0x05		15:8	OVFEO						CMPEO1	CMPEO0
0x06	INTENCLR	7:0	OVF	SYNCRDY					CMP1	CMP0
0x07	INTENSET	7:0	OVF	SYNCRDY					CMP1	CMP0

Offset	Name	Bit Pos.								
0x08	INTFLAG	7:0	OVF	SYNCRDY					CMP1	CMP0
0x09	Reserved									
0x0A	STATUS	7:0	SYNCBUSY							
0x0B	DBGCTRL	7:0								DBGRUN
0x0C	FREQCORR	7:0	SIGN	VALUE[6:0]						
0x0D ... 0x0F	Reserved									
0x10	COUNT	7:0	COUNT[7:0]							
0x11		15:8	COUNT[15:8]							
0x12	Reserved									
0x13	Reserved									
0x14	PER	7:0	PER[7:0]							
0x15		15:8	PER[15:8]							
0x16	Reserved									
0x17	Reserved									
0x18	COMP0	7:0	COMP[7:0]							
0x19		15:8	COMP[15:8]							
0x1A	COMP1	7:0	COMP[7:0]							
0x1B		15:8	COMP[15:8]							

Table 18-3. MODE2 - Mode Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0	MATCHCLR	CLKREP			MODE[1:0]		ENABLE	SWRST
0x01		15:8					PRESCALER[3:0]			
0x02	READREQ	7:0			ADDR[5:0]					
0x03		15:8	RREQ	RCONT						
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
0x05		15:8	OVFEO							ALARMEO0
0x06	INTENCLR	7:0	OVF	SYNCRDY						ALARM0
0x07	INTENSET	7:0	OVF	SYNCRDY						ALARM0
0x08	INTFLAG	7:0	OVF	SYNCRDY						ALARM0
0x09	Reserved									
0x0A	STATUS	7:0	SYNCBUSY							
0x0B	DBGCTRL	7:0								DBGRUN
0x0C	FREQCORR	7:0	SIGN	VALUE[6:0]						
0x0D ... 0x0F	Reserved									
0x10	CLOCK	7:0	MINUTE[1:0]		SECOND[5:0]					
0x11		15:8	HOUR[3:0]				MINUTE[5:2]			
0x12		23:16	MONTH[1:0]		DAY[4:0]					HOUR[4]
0x13		31:24	YEAR[5:0]						MONTH[3:2]	

Offset	Name	Bit Pos.								
0x14 ... 0x17	Reserved									
0x18	ALARM	7:0	MINUTE[1:0]		SECOND[5:0]					
0x19		15:8	HOUR[3:0]				MINUTE[5:2]			
0x1A		23:16	MONTH[1:0]		DAY[4:0]					HOUR[4]
0x1B		31:24	YEAR[5:0]						MONTH[3:2]	
0x1C	MASK	7:0						SEL[2:0]		

18.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Please refer to "Register Access Protection" on page 226 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Write-Synchronized or the Read-Synchronized property in each individual register description. Please refer to "Synchronization" on page 231 for details.

Some registers are enable-protected, meaning they can only be written when the RTC is disabled. Enable-protection is denoted by the Enable-Protected property in each individual register description.

18.8.1 Control - MODE0

Name: CTRL

Offset: 0x00

Reset: 0x0000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
					PRESCALER[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MATCHCLR				MODE[1:0]		ENABLE	SWRST
Access	R/W	R	R	R	R/W	R/W	R/W	W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:8 – PRESCALER[3:0]: Prescaler**
  These bits define the prescaling factor for the RTC clock source (GCLK_RTC) to generate the counter clock (CLK_RTC_CNT).
  These bits are not synchronized.

Table 18-4. Prescaler

PRESCALER[3:0]	Name	Description
0x0	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x2	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x3	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x4	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x5	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x6	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x7	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x8	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0x9	DIV512	CLK_RTC_CNT = GCLK_RTC/512
0xA	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xb-0xf		Reserved

- **Bit 7 – MATCHCLR: Clear on Match**
  This bit is valid only in Mode 0 and Mode 2.
  0: The counter is not cleared on a Compare/Alarm 0 match.
  1: The counter is cleared on a Compare/Alarm 0 match.
  This bit is not synchronized.

- **Bits 6:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 3:2 – MODE[1:0]: Operating Mode**
  These bits define the operating mode of the RTC.
  These bits are not synchronized.

Table 18-5. Operating Mode

MODE[1:0]	Name	Description
0x0	COUNT32	Mode 0: 32-bit Counter
0x1	COUNT16	Mode 1: 16-bit Counter
0x2	CLOCK	Mode 2: Clock/Calendar
0x3		Reserved

- **Bit 1 – ENABLE: Enable**
  0: The peripheral is disabled or being disabled.
  1: The peripheral is enabled or being enabled.
  Due to synchronization, there is delay from writing CTRL.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately, and the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set. STATUS.SYNCBUSY will be cleared when the operation is complete.
  This bit is not enable-protected.

- **Bit 0 – SWRST: Software Reset**
  0: There is no reset operation ongoing.
  1: The reset operation is ongoing.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all registers in the RTC, except DBGCTRL, to their initial state, and the RTC will be disabled.
  Writing a one to CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.
  Due to synchronization, there is a delay from writing CTRL.SWRST until the reset is complete. CTRL.SWRST and STATUS.SYNCBUSY will both be cleared when the reset is complete.
  This bit is not enable-protected.

18.8.2 Control - MODE1

Name: CTRL

Offset: 0x00

Reset: 0x0000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
					PRESCALER[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					MODE[1:0]		ENABLE	SWRST
Access	R	R	R	R	R/W	R/W	R/W	W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:8 – PRESCALER[3:0]: Prescaler**
  These bits define the prescaling factor for the RTC clock source (GCLK_RTC) to generate the counter clock (CLK_RTC_CNT).

  These bits are not synchronized.

Table 18-6. Prescaler

PRESCALER[3:0]	Name	Description
0x0	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x2	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x3	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x4	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x5	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x6	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x7	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x8	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0x9	DIV512	CLK_RTC_CNT = GCLK_RTC/512
0xA	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xb-0xf		Reserved

- **Bits 7:4 – Reserved**
These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 3:2 – MODE[1:0]: Operating Mode**
These bits define the operating mode of the RTC.
These bits are not synchronized.

Table 18-7. Operating Mode

MODE[1:0]	Name	Description
0x0	COUNT32	Mode 0: 32-bit Counter
0x1	COUNT16	Mode 1: 16-bit Counter
0x2	CLOCK	Mode 2: Clock/Calendar
0x3		Reserved

- **Bit 1 – ENABLE: Enable**
0: The peripheral is disabled or being disabled.
1: The peripheral is enabled or being enabled.
Due to synchronization, there is delay from writing CTRL.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately, and the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set. STATUS.SYNCBUSY will be cleared when the operation is complete.
This bit is not enable-protected.

- **Bit 0 – SWRST: Software Reset**
0: There is no reset operation ongoing.
1: The reset operation is ongoing.
Writing a zero to this bit has no effect.
Writing a one to this bit resets all registers in the RTC, except DBGCTRL, to their initial state, and the RTC will be disabled.
Writing a one to CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.
Due to synchronization, there is a delay from writing CTRL.SWRST until the reset is complete. CTRL.SWRST and STATUS.SYNCBUSY will both be cleared when the reset is complete.
This bit is not enable-protected.

18.8.3 Control - MODE2

Name: CTRL

Offset: 0x00

Reset: 0x0000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
					PRESCALER[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MATCHCLR	CLKREP			MODE[1:0]		ENABLE	SWRST
Access	R/W	R/W	R	R	R/W	R/W	R/W	W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:8 – PRESCALER[3:0]: Prescaler**
  These bits define the prescaling factor for the RTC clock source (GCLK_RTC) to generate the counter clock (CLK_RTC_CNT).
  These bits are not synchronized.

Table 18-8. Prescaler

PRESCALER[3:0]	Name	Description
0x0	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x2	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x3	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x4	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x5	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x6	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x7	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x8	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0x9	DIV512	CLK_RTC_CNT = GCLK_RTC/512
0xA	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xb-0xf		Reserved

- **Bit 7 – MATCHCLR: Clear on Match**

  This bit is valid only in Mode 0 and Mode 2. This bit can be written only when the peripheral is disabled.

  0: The counter is not cleared on a Compare/Alarm 0 match.

  1: The counter is cleared on a Compare/Alarm 0 match.

  This bit is not synchronized.

- **Bit 6 – CLKREP: Clock Representation**

  This bit is valid only in Mode 2 and determines how the hours are represented in the Clock Value (CLOCK) register. This bit can be written only when the peripheral is disabled.

  0: 24 Hour

  1: 12 Hour (AM/PM)

  This bit is not synchronized.

- **Bits 5:4 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 3:2 – MODE[1:0]: Operating Mode**

  These bits define the operating mode of the RTC.

  These bits are not synchronized.

Table 18-9. Operating Mode

MODE[1:0]	Name	Description
0x0	COUNT32	Mode 0: 32-bit Counter
0x1	COUNT16	Mode 1: 16-bit Counter
0x2	CLOCK	Mode 2: Clock/Calendar
0x3		Reserved

- **Bit 1 – ENABLE: Enable**

  0: The peripheral is disabled or being disabled.

  1: The peripheral is enabled or being enabled.

  Due to synchronization, there is delay from writing CTRL.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately, and the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set. STATUS.SYNCBUSY will be cleared when the operation is complete.

  This bit is not enable-protected.

- **Bit 0 – SWRST: Software Reset**

  0: There is no reset operation ongoing.

  1: The reset operation is ongoing.

  Writing a zero to this bit has no effect.

  Writing a one to this bit resets all registers in the RTC, except DBGCTRL, to their initial state, and the RTC will be disabled.

  Writing a one to CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.

  Due to synchronization, there is a delay from writing CTRL.SWRST until the reset is complete. CTRL.SWRST and STATUS.SYNCBUSY will both be cleared when the reset is complete.

  This bit is not enable-protected.

18.8.4 Read Request

Name: READREQ

Offset: 0x02

Reset: 0x0010

Access: Read-Write

Property: -

Bit	15	14	13	12	11	10	9	8
	RREQ	RCONT						
Access	W	R/W	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			ADDR[5:0]					
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	1	0	0	0	0

- **Bit 15 – RREQ: Read Request**

  Writing a zero to this bit has no effect.

  Writing a one to this bit requests synchronization of the register pointed to by the Address bit group (READREQ.ADDR) and sets the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY).

- **Bit 14 – RCONT: Read Continuously**

  Writing a zero to this bit disables continuous synchronization.

  Writing a one to this bit enables continuous synchronization of the register pointed to by READREQ.ADDR. The register value will be synchronized automatically every time the register is updated. READREQ.RCONT prevents READREQ.RREQ from clearing automatically.

  This bit is cleared when the register pointed to by READREQ.ADDR is written.

- **Bits 13:6 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – ADDR[5:0]: Address**

  These bits select the offset of the register that needs read synchronization. In the RTC only COUNT and CLOCK, which share the same address, are available for read synchronization. Therefore, ADDR is a read-only constant of 0x10.

18.8.5 Event Control - MODE0

Name: EVCTRL

Offset: 0x04

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
	OVFEO							CMPEO0
Access	R/W	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 15 – OVFEO: Overflow Event Output Enable**
  0: Overflow event is disabled and will not be generated.
  1: Overflow event is enabled and will be generated for every overflow.

- **Bits 14:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 8 – CMPEO: Compare x Event Output Enable**
  0: Compare 0 event is disabled and will not be generated.
  1: Compare 0 event is enabled and will be generated for every compare match.

- **Bits 7:0 – PEREOx [x=7..0]: Periodic Interval x Event Output Enable**
  0: Periodic Interval x event is disabled and will not be generated.
  1: Periodic Interval x event is enabled and will be generated.

18.8.6 Event Control - MODE1

Name: EVCTRL

Offset: 0x04

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
	OVFEO						CMPEO1	CMPEO0
Access	R/W	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 15 – OVFEO: Overflow Event Output Enable**
  0: Overflow event is disabled and will not be generated.
  1: Overflow event is enabled and will be generated for every overflow.
- **Bits 14:10 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 9:8 – CMPEOx [x=1..0]: Compare x Event Output Enable**
  0: Compare x event is disabled and will not be generated.
  1: Compare x event is enabled and will be generated for every compare match.
- **Bits 7:0 – PEREOx [x=7..0]: Periodic Interval x Event Output Enable**
  0: Periodic Interval x event is disabled and will not be generated.
  1: Periodic Interval x event is enabled and will be generated.

18.8.7 Event Control - MODE2

Name: EVCTRL

Offset: 0x04

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
	OVFEO							ALARMEO0
Access	R/W	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 15 – OVFEO: Overflow Event Output Enable**
  0: Overflow event is disabled and will not be generated.
  1: Overflow event is enabled and will be generated for every overflow.
- **Bits 14:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 8 – ALARMEO: Alarm x Event Output Enable**
  0: Alarm 0 event is disabled and will not be generated.
  1: Alarm 0 event is enabled and will be generated for every alarm.
- **Bits 7:0 – PEREOx [x=7..0]: Periodic Interval x Event Output Enable**
  0: Periodic Interval x event is disabled and will not be generated.
  1: Periodic Interval x event is enabled and will be generated.

18.8.8 Interrupt Enable Clear - MODE0

Name: INTENCLR

Offset: 0x06

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						CMP0
Access	R/W	R/W	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

- **Bit 7 – OVF: Overflow Interrupt Enable**

  0: The Overflow interrupt is disabled.

  1: The Overflow interrupt is enabled, and an interrupt request will be generated when the Overflow interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Overflow Interrupt Enable bit and disable the corresponding interrupt.

- **Bit 6 – SYNCRDY: Synchronization Ready Interrupt Enable**

  0: The Synchronization Ready interrupt is disabled.

  1: The Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the Synchronization Ready interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Synchronization Ready Interrupt Enable bit and disable the corresponding interrupt.

- **Bits 5:1 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – CMP: Compare x Interrupt Enable**

  0: The Compare 0 interrupt is disabled.

  1: The Compare 0 interrupt is enabled, and an interrupt request will be generated when the Compare x interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Compare 0 Interrupt Enable bit and disable the corresponding interrupt.

18.8.9 Interrupt Enable Clear - MODE1

Name: INTENCLR

Offset: 0x06

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY					CMP1	CMP0
Access	R/W	R/W	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – OVF: Overflow Interrupt Enable**
  0: The Overflow interrupt is disabled.
  1: The Overflow interrupt is enabled, and an interrupt request will be generated when the Overflow interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Overflow Interrupt Enable bit and disable the corresponding interrupt.

- **Bit 6 – SYNCRDY: Synchronization Ready Interrupt Enable**
  0: The Synchronization Ready interrupt is disabled.
  1: The Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the Synchronization Ready interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Synchronization Ready Interrupt Enable bit and disable the corresponding interrupt.

- **Bits 5:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 1:0 – CMPx [x=1..0]: Compare x Interrupt Enable**
  0: The Compare x interrupt is disabled.
  1: The Compare x interrupt is enabled, and an interrupt request will be generated when the Compare x interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Compare x Interrupt Enable bit and disable the corresponding interrupt.

18.8.10 Interrupt Enable Clear - MODE2

Name: INTENCLR

Offset: 0x06

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						ALARM0
Access	R/W	R/W	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – OVF: Overflow Interrupt Enable**
  0: The Overflow interrupt is disabled.
  1: The Overflow interrupt is enabled, and an interrupt request will be generated when the Overflow interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Overflow Interrupt Enable bit and disable the corresponding interrupt.

- **Bit 6 – SYNCRDY: Synchronization Ready Interrupt Enable**
  0: The synchronization ready interrupt is disabled.
  1: The synchronization ready interrupt is enabled, and an interrupt request will be generated when the Synchronization Ready interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Synchronization Ready Interrupt Enable bit and disable the corresponding interrupt.

- **Bits 5:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – ALARM: Alarm x Interrupt Enable**
  0: The Alarm 0 interrupt is disabled.
  1: The Alarm 0 interrupt is enabled, and an interrupt request will be generated when the Alarm 0 interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the Alarm 0 interrupt.

18.8.11 Interrupt Enable Set - MODE0

Name: INTENSET

Offset: 0x07

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						CMP0
Access	R/W	R/W	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

- **Bit 7 – OVF: Overflow Interrupt Enable**
  0: The overflow interrupt is disabled.
  1: The overflow interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Overflow Interrupt Enable bit and enable the Overflow interrupt.

- **Bit 6 – SYNCRDY: Synchronization Ready Interrupt Enable**
  0: The synchronization ready interrupt is disabled.
  1: The synchronization ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Synchronization Ready Interrupt Enable bit and enable the Synchronization Ready interrupt.

- **Bits 5:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – CMP: Compare x Interrupt Enable**
  0: The compare 0 interrupt is disabled.
  1: The compare 0 interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Compare 0 Interrupt Enable bit and enable the Compare 0 interrupt.

18.8.12 Interrupt Enable Set - MODE1

Name: INTENSET

Offset: 0x07

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY					CMP1	CMP0
Access	R/W	R/W	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – OVF: Overflow Interrupt Enable**

  0: The overflow interrupt is disabled.

  1: The overflow interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the Overflow interrupt bit and enable the Overflow interrupt.

- **Bit 6 – SYNCRDY: Synchronization Ready Interrupt Enable**

  0: The synchronization ready interrupt is disabled.

  1: The synchronization ready interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the Synchronization Ready Interrupt Enable bit and enable the Synchronization Ready interrupt.

- **Bits 5:2 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 1:0 – CMPx [x=1..0]: Compare x Interrupt Enable**

  0: The compare x interrupt is disabled.

  1: The compare x interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the Compare x Interrupt Enable bit and enable the Compare x interrupt.

18.8.13 Interrupt Enable Set - MODE2

Name: INTENSET

Offset: 0x07

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						ALARM0
Access	R/W	R/W	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – OVF: Overflow Interrupt Enable**
  0: The overflow interrupt is disabled.
  1: The overflow interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Overflow Interrupt Enable bit and enable the Overflow interrupt.

- **Bit 6 – SYNCRDY: Synchronization Ready Interrupt Enable**
  0: The synchronization ready interrupt is disabled.
  1: The synchronization ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Synchronization Ready Interrupt bit and enable the Synchronization Ready interrupt.

- **Bits 5:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – ALARM: Alarm x Interrupt Enable**
  0: The alarm 0 interrupt is disabled.
  1: The alarm 0 interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Alarm 0 Interrupt Enable bit and enable the Alarm 0 interrupt.

18.8.14 Interrupt Flag Status and Clear - MODE0

Name: INTFLAG

Offset: 0x08

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						CMP0
Access	R/W	R/W	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – OVF: Overflow**

  This flag is cleared by writing a one to the flag.

  This flag is set on the next CLK_RTC_CNT cycle after an overflow condition occurs, and an interrupt request will be generated if INTENCLR/SET.OVF is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the Overflow interrupt flag.

- **Bit 6 – SYNCRDY: Synchronization Ready**

  This flag is cleared by writing a one to the flag.

  This flag is set on a 1-to-0 transition of the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY), except when caused by enable or software reset, and an interrupt request will be generated if INTEN-CLR/SET.SYNCRDY is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the Synchronization Ready interrupt flag.

- **Bits 5:1 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – CMP: Compare x**

  This flag is cleared by writing a one to the flag.

  This flag is set on the next CLK_RTC_CNT cycle after a match with the compare condition, and an interrupt request will be generated if INTENCLR/SET.CMP0 is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the Compare 0 interrupt flag.

18.8.15 Interrupt Flag Status and Clear - MODE1

Name: INTFLAG

Offset: 0x08

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY					CMP1	CMP0
Access	R/W	R/W	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – OVF: Overflow**
  This flag is cleared by writing a one to the flag.
  This flag is set on the next CLK_RTC_CNT cycle after an overflow condition occurs, and an interrupt request will be generated if INTENCLR/SET.OVF is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Overflow interrupt flag.

- **Bit 6 – SYNCRDY: Synchronization Ready**
  This flag is cleared by writing a one to the flag.
  This flag is set on a 1-to-0 transition of the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY), except when caused by enable or software reset, and an interrupt request will be generated if INTEN-CLR/SET.SYNCRDY is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Synchronization Ready interrupt flag.

- **Bits 5:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 1:0 – CMPx [x=1..0]: Compare x**
  This flag is cleared by writing a one to the flag.
  This flag is set on the next CLK_RTC_CNT cycle after a match with the compare condition , and an interrupt request will be generated if INTENCLR/SET.CMPx is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Compare x interrupt flag.

18.8.16 Interrupt Flag Status and Clear - MODE2

Name: INTFLAG

Offset: 0x08

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						ALARM0
Access	R/W	R/W	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – OVF: Overflow**
  This flag is cleared by writing a one to the flag.
  This flag is set on the next CLK_RTC_CNT cycle after an overflow condition occurs, and an interrupt request will be generated if INTENCLR/SET.OVF is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Overflow interrupt flag.

- **Bit 6 – SYNCRDY: Synchronization Ready**
  This flag is cleared by writing a one to the flag.
  This flag is set on a 1-to-0 transition of the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY), except when caused by enable or software reset, and an interrupt request will be generated if INTEN-CLR/SET.SYNCRDY is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Synchronization Ready interrupt flag.

- **Bits 5:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – ALARM: Alarm x**
  This flag is cleared by writing a one to the flag.
  This flag is set on the next CLK_RTC_CNT cycle after a match with ALARM0 condition occurs , and an interrupt request will be generated if INTENCLR/SET.ALARM0 is also one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Alarm 0 interrupt flag.

18.8.17 Status

Name: STATUS

Offset: 0x0A

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCBUSY							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – SYNCBUSY: Synchronization Busy**
  This bit is cleared when the synchronization of registers between the clock domains is complete.
  This bit is set when the synchronization of registers between clock domains is started.

- **Bits 6:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

18.8.18 Debug Control

Name: DBGCTRL

Offset: 0x0B

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – DBGRUN: Run During Debug**
  This bit is not reset by a software reset.
  Writing a zero to this bit causes the RTC to halt during debug mode.
  Writing a one to this bit allows the RTC to continue normal operation during debug mode.

18.8.19 Frequency Correction

Name: FREQCORR

Offset: 0x0C

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	SIGN	VALUE[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – SIGN: Correction Sign**
  0: The correction value is positive, i.e., frequency will be increased.
  1: The correction value is negative, i.e., frequency will be decreased.

- **Bits 6:0 – VALUE[6:0]: Correction Value**
  These bits define the amount of correction applied to the RTC prescaler.
  0: Correction is disabled and the RTC frequency is unchanged.
  1–127: The RTC frequency is adjusted according to the value.

18.8.20 Counter Value - MODE0

Name: COUNT

Offset: 0x10

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
	COUNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COUNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – COUNT[31:0]: Counter Value**
  These bits define the value of the 32-bit RTC counter.

18.8.21 Counter Value - MODE1

Name: COUNT

Offset: 0x10

Reset: 0x0000

Access: Read-Write

Property: -

Bit	15	14	13	12	11	10	9	8
	COUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – COUNT[15:0]: Counter Value**
  These bits define the value of the 16-bit RTC counter.

18.8.22 Clock Value - MODE2

Name: CLOCK

Offset: 0x10

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
	YEAR[5:0]						MONTH[3:2]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MONTH[1:0]		DAY[4:0]					HOUR[4]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HOUR[3:0]				MINUTE[5:2]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MINUTE[1:0]		SECOND[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:26 – YEAR[5:0]: Year**
  The year offset with respect to the reference year (defined in software).
  The year is considered a leap year if YEAR[1:0] is zero.

- **Bits 25:22 – MONTH[3:0]: Month**
  1 – January
  2 – February
  ...
  12 – December

- **Bits 21:17 – DAY[4:0]: Day**
  Day starts at 1 and ends at 28, 29, 30 or 31, depending on the month and year.

- **Bits 16:12 – HOUR[4:0]: Hour**
  When CTRL.CLKREP is zero, the Hour bit group is in 24-hour format, with values 0-23. When CTRL.CLKREP is one, HOUR[3:0] has values 1-12 and HOUR[4] represents AM (0) or PM (1).

Table 18-10. Hour

HOUR[4:0]	Name	Description
0x0-0xf		Reserved
0x10	PM	Afternoon Hour
0x11-0x1f		Reserved

- **Bits 11:6 – MINUTE[5:0]: Minute**
  0 – 59.
- **Bits 5:0 – SECOND[5:0]: Second**
  0– 59.

18.8.23 Counter Period - MODE1

Name: PER

Offset: 0x14

Reset: 0x0000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	PER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – PER[15:0]: Counter Period**
  These bits define the value of the 16-bit RTC period.

18.8.24 Compare n Value - MODE0

Name: COMP

Offset: 0x18

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	COMP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COMP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – COMP[31:0]: Compare Value**
  The 32-bit value of COMPn is continuously compared with the 32-bit COUNT value. When a match occurs, the Compare n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn) is set on the next counter cycle, and the counter value is cleared if CTRL.MATCHCLR is one.

18.8.25 Compare n Value - MODE1

Name: COMPn

Offset: 0x18+n*0x2 [n=0..1]

Reset: 0x0000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	COMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – COMP[15:0]: Compare Value**
  The 16-bit value of COMPn is continuously compared with the 16-bit COUNT value. When a match occurs, the Compare n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn) is set on the next counter cycle.

18.8.26 Alarm n Value - MODE2

Name: ALARM

Offset: 0x18

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	YEAR[5:0]						MONTH[3:2]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MONTH[1:0]		DAY[4:0]					HOUR[4]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HOUR[3:0]				MINUTE[5:2]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MINUTE[1:0]		SECOND[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

The 32-bit value of ALARM0 is continuously compared with the 32-bit CLOCK value, based on the masking set by MASKn.SEL. When a match occurs, the Alarm 0 interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.ALARMn) is set on the next counter cycle, and the counter is cleared if CTRL.MATCHCLR is one.

- **Bits 31:26 – YEAR[5:0]: Year**
  The alarm year. Years are only matched if MASKn.SEL is 6.
- **Bits 25:22 – MONTH[3:0]: Month**
  The alarm month. Months are matched only if MASKn.SEL is greater than 4.
- **Bits 21:17 – DAY[4:0]: Day**
  The alarm day. Days are matched only if MASKn.SEL is greater than 3.
- **Bits 16:12 – HOUR[4:0]: Hour**
  The alarm hour. Hours are matched only if MASKn.SEL is greater than 2.
- **Bits 11:6 – MINUTE[5:0]: Minute**
  The alarm minute. Minutes are matched only if MASKn.SEL is greater than 1.

- **Bits 5:0 – SECOND[5:0]: Second**
  The alarm second. Seconds are matched only if MASKn.SEL is greater than 0.

18.8.27 Alarm n Mask - MODE2

Name: MASK

Offset: 0x1C

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
						SEL[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – SEL[2:0]: Alarm Mask Selection**
  These bits define which bit groups of Alarm n are valid.

Table 18-11. Alarm Mask Selection

SEL[2:0]	Name	Description
0x0	OFF	Alarm Disabled
0x1	SS	Match seconds only
0x2	MMSS	Match seconds and minutes only
0x3	HHMMSS	Match seconds, minutes, and hours only
0x4	DDHHMMSS	Match seconds, minutes, hours, and days only
0x5	MMDDHHMMSS	Match seconds, minutes, hours, days, and months only
0x6	YYMMDDHHMMSS	Match seconds, minutes, hours, days, months, and years
0x7		Reserved

19. DMAC – Direct Memory Access Controller

19.1 Overview

The Direct Memory Access Controller (DMAC) contains both a Direct Memory Access engine and a Cyclic Redundancy Check (CRC) engine. The DMAC can transfer data between memories and peripherals, and thus off-load these tasks from the CPU. It enables high data transfer rates with minimum CPU intervention, and frees up CPU time. With access to all peripherals, the DMAC can handle automatic transfer of data between communication modules.

For the DMA part of the DMAC, it has several DMA channels which all can receive different types of transfer triggers, which will result in transfer requests from the DMA channels to the arbiter. Refer to Figure 19-1. The arbiter will grant one DMA channel at a time to act as the active channel. When the active channel has been granted, the fetch engine of the DMAC will fetch a transfer descriptor from SRAM into the internal memory of the active channel, before the active channel starts its data transmission. A DMA channel's data transfer can be interrupted by a higher prioritized channel. The DMAC will write back the updated transfer descriptor from the internal memory of the active channel to SRAM, before the higher prioritized channel gets to start its transfer. Once a DMA channel is done with its transfer optionally interrupts and events can be generated.

As one can see from Figure 19-1, the DMAC has four bus interfaces. The data transfer bus, which is used for performing the actual DMA transfer is an AHB master interface. The AHB/APB Bridge bus is an APB slave interface and is the bus used when writing and reading the I/O registers of the DMAC. The descriptor fetch bus is an AHB master interface and is used by the fetch engine, to fetch transfer descriptors from SRAM before a transfer can be started or continued. At last there is the write-back bus, which is an AHB master interface and it is used to write the transfer descriptor back to SRAM.

As mentioned, the DMAC also has a CRC module available. This can be used by software to detect an accidental error in the transferred data and to take corrective action, such as requesting the data to be sent again or simply not using the incorrect data.

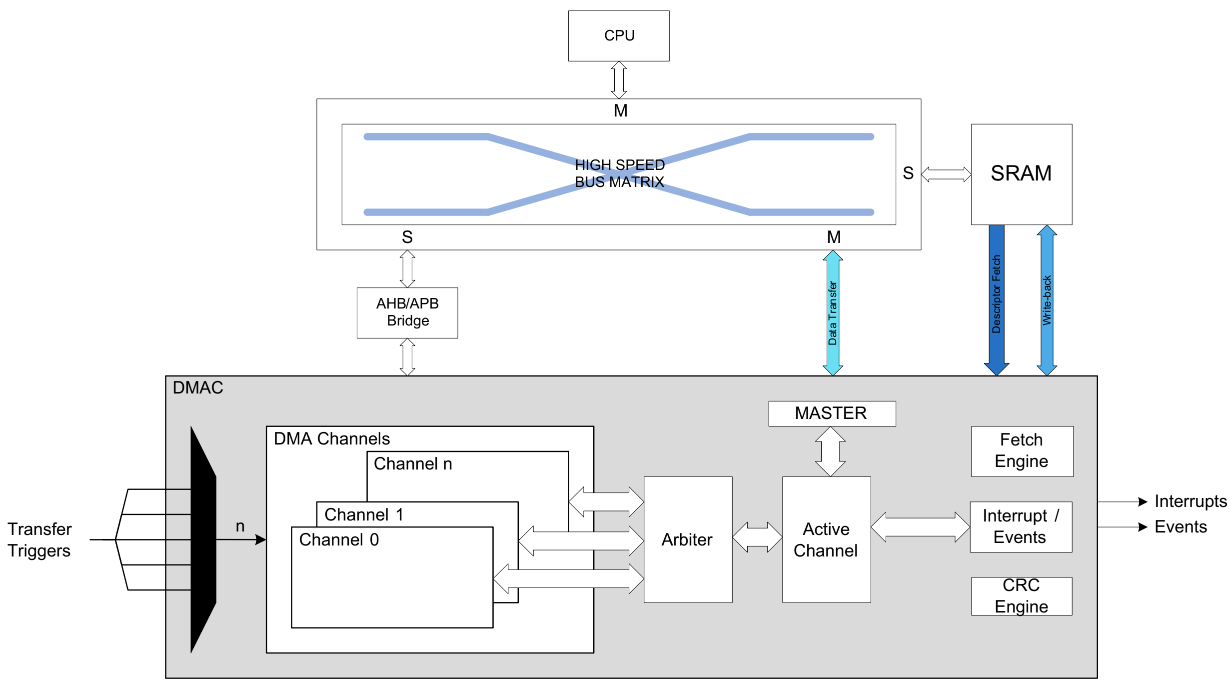
19.2 Features

- Data transfer between
  - Peripheral to peripheral
  - Peripheral to memory
  - Memory to peripheral
  - Memory to memory
- Transfer trigger sources
  - Software
  - Events from Event System
  - Dedicated requests from peripherals
- SRAM based transfer descriptors
  - Single transfer using one descriptor
  - Multi-buffer or circular buffer modes by linking multiple descriptors
- 6 channels
  - Enable 6 independent transfers
  - Automatic descriptor fetch for each channel
  - Suspend/resume operation support for each channel
- Flexible arbitration scheme
  - 4 configurable priority levels for each channel
  - Fixed or round-robin priority scheme within each priority level
- From 1 to 256kB data transfer in a single block transfer
- Multiple addressing modes
  - Static
  - Configurable increment scheme

- Optional interrupt generation
  - On block transfer complete
  - On error detection
  - On channel suspend
- 4 event inputs
  - One event input for each of the 4 least significant DMA channels
  - Can be selected to trigger normal transfers, periodic transfers or conditional transfers
  - Can be selected to suspend or resume channel operation
- 4 event outputs
  - One output event for each of the 4 least significant DMA channels
  - Selectable generation on AHB, burst, block or transaction transfer complete
- Error management supported by write-back function
  - Dedicated Write-Back memory section for each channel to store ongoing descriptor transfer
- CRC polynomial software selectable to
  - CRC-16 (CRC-CCITT)
  - CRC-32 (IEEE 802.3)

19.3 Block Diagram

Figure 19-1. DMAC Block Diagram



19.4 Signal Description

Not applicable.

19.5 Product Dependencies

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

19.5.1 I/O Lines

Not applicable.

19.5.2 Power Management

The DMAC will continue to operate in any sleep mode where the selected source clock is running. The DMAC's interrupts can be used to wake up the device from sleep modes. Events connected to the event system can trigger other operations in the system without exiting sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes. On hardware or software reset, all registers are set to their reset value.

19.5.3 Clocks

The DMAC bus clock (CLK_DMAC_APB) can be enabled and disabled in the power manager, and the default state of CLK_DMAC_APB can be found in "Peripheral Clock Masking" on page 113.

An AHB clock (CLK_DMAC_AHB) is required to clock the DMAC. This clock must be configured and enabled in the power manager before using the DMAC, and the default state of CLK_DMAC_AHB can be found in "Peripheral Clock Masking" on page 113.

This bus clock (CLK_DMAC_APB) is always synchronous to the module clock (CLK_DMAC_AHB), but can be divided by a prescaler and may run even when the module clock is turned off.

19.5.4 DMA

Not applicable.

19.5.5 Interrupts

The interrupt request line is connected to the interrupt controller. Using the DMAC interrupts requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

19.5.6 Events

The events are connected to the event system. Refer to "EVSYS – Event System" on page 402 for details on how to configure the Event System.

19.5.7 Debug Operation

When the CPU is halted in debug mode the DMAC will halt normal operation. The DMAC can be forced to continue operation during debugging. Refer to DBGCTRL for details.

19.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the peripheral access controller (PAC), except the following registers:

- Interrupt Pending (INTPEND) register
- Channel ID (CHID) register
- Channel Interrupt Flag Status and Clear (CHINTFLAG) register

Write-protection is denoted by the Write-Protected property in the register description.

Write-protection does not apply to accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

19.5.9 Analog Connections

Not applicable.

19.6 Functional Description

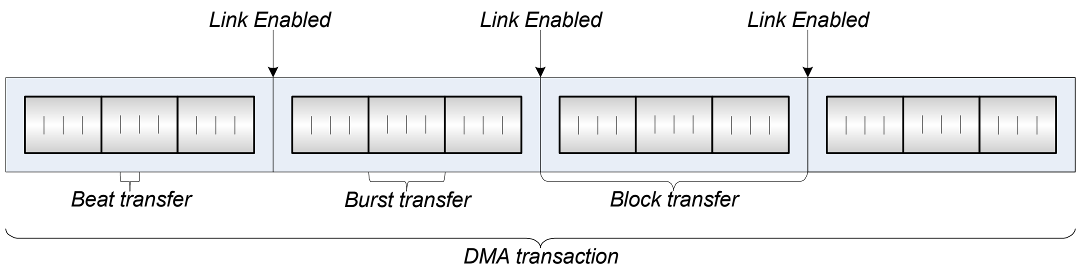
19.6.1 Principle of Operation

The DMAC consists of a DMA module and a CRC module.

19.6.1.1 DMA

The DMAC can, without interaction from the CPU, transfer data between peripherals and memories. The data transferred by the DMAC are called transactions, and these transactions can be split into smaller data transfers. Figure 19-2 shows the relationship between the different transfer sizes.

Figure 19-2. DMA Transfer Sizes



- Beat transfer: Defined as the size of one data transfer bus access, and the size is selected by writing the Beat Size bit group in the Block Transfer Control register (BTCTRL.BEATSIZE)
- Burst transfer: Defined as n beat transfers, where n will differ from one device family to another. For this device family, n is 1. A burst transfer is atomic, and cannot be interrupted.
- Block transfer: The amount of data one transfer descriptor can transfer, and the amount can range from 1 to 64k beats. In contrast to the burst transfer, a block transfer can be interrupted.
- Transaction: The DMAC can link several transfer descriptors by having the first descriptor pointing to the second and so forth, as shown in Figure 19-2. A DMA transaction is defined as all block transfers within a linked list, being completed.

A transfer descriptor describes how a block transfer should be carried out by the DMAC, and it must remain in SRAM. For further details on the transfer descriptor refer to "Transfer Descriptors" on page 273.

Figure 19-2 shows several block transfers linked together, which are called linked descriptors. For further information about linked descriptors, refer to "Linked Descriptors" on page 280.

A DMA transfer is initiated by an incoming transfer trigger on one of the DMA channels. This trigger can be configured to be either a software trigger, an event trigger or one of the dedicated peripheral triggers. The transfer trigger will result in a DMA transfer request from the specific channel to the arbiter, and if there are several DMA channels with pending transfer requests, the arbiter has to choose which channel to grant access to become the active channel. The DMA channel granted access as the active channel will carry out the transaction as configured in the transfer descriptor. The DMA channel can be interrupted by a higher prioritized channel after each burst transfer, but will resume its block transfer when it is granted access as the active channel again.

For each beat transfer an optional output event can be generated, and for each block transfer optional interrupts and an optional output event can be generated. When a transaction is completed, dependent of the configuration, the DMA channel will either be suspended or disabled.

19.6.1.2 CRC

The internal CRC supports two commonly used CRC polynomials; CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3). It can be used with selectable DMA channel or independently, with I/O interface.

19.6.2 Basic Operation

19.6.2.1 Initialization

The following DMAC registers are enable-protected, meaning that they can only be written when the DMAC is disabled (CTRL.DMAENABLE is zero):

- Descriptor Base Memory Address (BASEADDR) register
- Write-Back Memory Base Address (WRBADDR) register

The following DMAC bit is enable-protected, meaning that it can only be written when both the DMAC and CRC are disabled (CTRL.DMAENABLE and CTRL.CRCENABLE is zero):

- Software Reset bit in Control register (CTRL.SWRST)

The following DMA channel register is enable-protected, meaning that it can only be written when the corresponding DMA channel is disabled (CHCTRLA.ENABLE is zero):

- Channel Control B (CHCTRLB) register, except the Command (CHCTRLB.CMD) and Channel Arbitration Level (CHCTRLB.LVL) bits

The following DMA channel bit is enable-protected, meaning that it can only be written when the corresponding DMA channel is disabled:

- Channel Software Reset bit in Channel Control A register (CHCTRLA.SWRST)

The following CRC registers are enable-protected, meaning that they can only be written when the CRC is disabled (CTRL.CRCENABLE is zero):

- CRC Control (CRCCTRL) register
- CRC Checksum (CRCCHKSUM) register

Enable-protection is denoted by the Enable-Protected property in the register description.

Before the DMAC is enabled, it must be configured, as outlined by the following steps:

- The SRAM address of where the descriptor memory section is located must be written to the Description Base Address (BASEADDR) register
- The SRAM address of where the write-back section should be located must be written to the Write-Back Memory Base Address (WRBADDR) register
- Priority level x of the arbiter can be enabled by writing a one to the Priority Level x Enable bit in the Control register(CTRL.LVLENx)

Before a DMA channel is enabled, the DMA channel and the corresponding first transfer descriptor must be configured, as outlined by the following steps:

- DMA channel configurations
  - The channel number of the DMA channel to configure must be written to the Channel ID (CHID) register
  - Trigger action must be selected by writing the Trigger Action bit group in the Channel Control B register (CHCTRLB.TRIGACT)
  - Trigger source must be selected by writing the Trigger Source bit group in the Channel Control B register (CHCTRLB.TRIGSRC)
- Transfer Descriptor
  - The size of each access of the data transfer bus must be selected by writing the Beat Size bit group in the Block Transfer Control register (BTCTRL.BEATSIZE)
  - The transfer descriptor must be made valid by writing a one to the Valid bit in the Block Transfer Control register (BTCTRL.VALID)
  - Number of beats in the block transfer must be selected by writing the Block Transfer Count (BTCNT) register
  - Source address for the block transfer must be selected by writing the Block Transfer Source Address (SRCADDR) register
  - Destination address for the block transfer must be selected by writing the Block Transfer Destination Address (DSTADDR) register

If CRC calculation is needed the CRC module must be configured before it is enabled, as outlined by the following steps:

- CRC input source must selected by writing the CRC Input Source bit group in the CRC Control register (CRCCTRL.CRCSRC)

- Type of CRC calculation must be selected by writing the CRC Polynomial Type bit group in the CRC Control register (CRCCTRL.CRCPOLY)
- If I/O is chosen as input source, the beat size must be selected by writing the CRC Beat Size bit group in the CRC Control register (CRCCTRL.CRCBEATSIZE)

19.6.2.2 Enabling, Disabling and Resetting

The DMAC is enabled by writing a one to the DMA Enable bit in the Control register (CTRL.DMAENABLE). The DMAC is disabled by writing a zero to CTRL.DMAENABLE.

A DMA channel is enabled by writing a one to Enable bit in the Channel Control A register (CHCTRLA.ENABLE), after writing the corresponding channel id to the Channel ID bit group in the Channel ID register (CHID.ID). A DMA channel is disabled by writing a zero to CHCTRLA.ENABLE.

The CRC is enabled by writing a one to the CRC Enable bit in the Control register (CTRL.CRCENABLE). The CRC is disabled by writing a zero to CTRL.CRCENABLE.

The DMAC is reset by writing a one to the Software Reset bit in the Control register (CTRL.SWRST), when the DMAC and CRC are disabled. All registers in the DMAC, except DBGCTRL, will be reset to their initial state.

A DMA channel is reset by writing a one to the Software Reset bit in the Channel Control A register (CHCTRLA.SWRST), after writing the corresponding channel id to the Channel ID bit group in the Channel ID register (CHID.ID). The channel registers will be reset to their initial state. The corresponding DMA channel must be disabled in order for the reset to take effect.

19.6.2.3 Transfer Descriptors

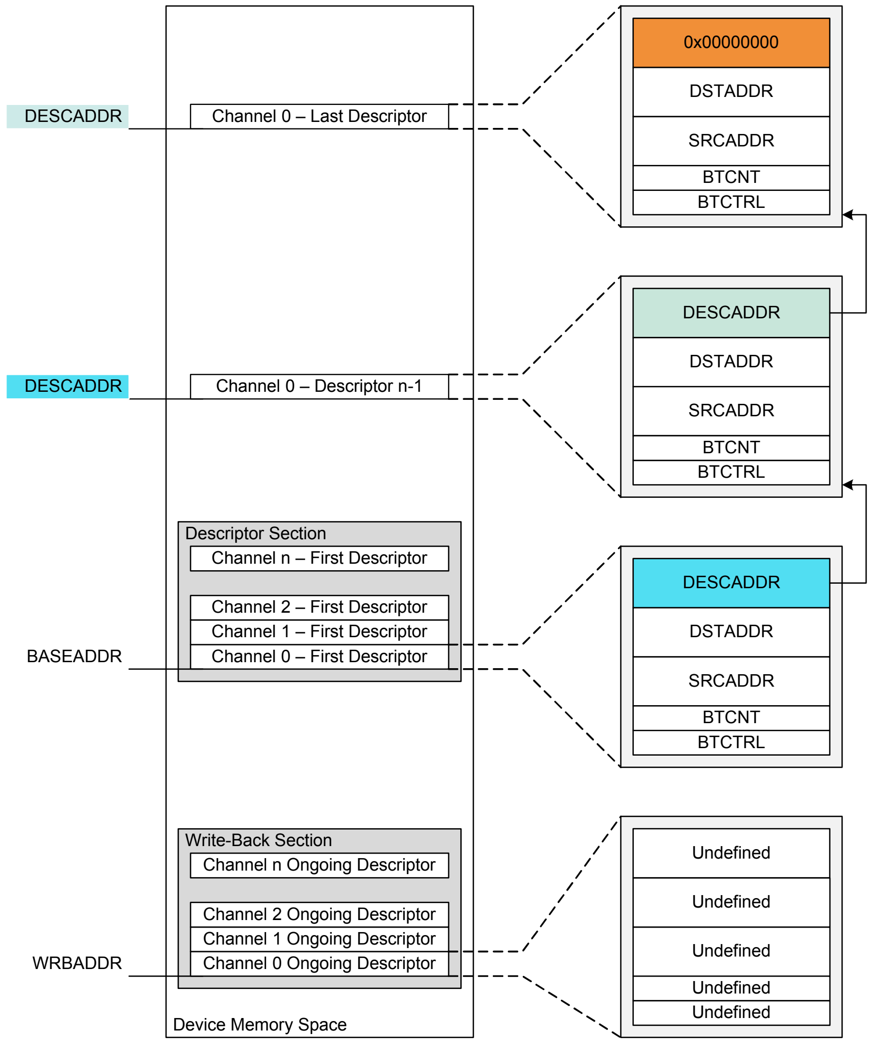
Together with the channel configurations the transfer descriptors decides how a block transfer should be executed. Before a DMA channel is enabled (CHCTRLA.ENABLE is written to one), and receives a transfer trigger, its first transfer descriptor has to be initialized and valid (BTCTRL.VALID). The first transfer descriptor describes the first block transfer of a transaction. For further details on the content of a transfer descriptor, refer to "Block Transfer Control" on page 325.

All transfer descriptors must reside in SRAM and the addresses stored in the Descriptor Memory Section Base Address (BASEADDR) and Write-Back Memory Section Base Address (WRBADDR) registers tells the DMAC where to find the descriptor memory section and the write-back memory section.

The descriptor memory section is where the DMAC expects to find the first transfer descriptors for all DMA channels. As BASEADDR points only to the first transfer descriptor of channel 0, refer to Figure 19-3, all first transfer descriptors must be stored in a contiguous memory section, where the transfer descriptors must be ordered according to their channel number. Figure 19-3 shows an example of linked descriptors on DMA channel 0. For further details on linked descriptors, refer to "Linked Descriptors" on page 280.

The write-back memory section is the section where the DMAC stores the transfer descriptors for the ongoing block transfers. WRBADDR points to the ongoing transfer descriptor of channel 0. All ongoing transfer descriptors will be stored in a contiguous memory section where the transfer descriptors are ordered according to their channel number. Figure 19-3 shows an example of linked descriptors on DMA channel 0. For further details on linked descriptors, refer to "Linked Descriptors" on page 280.

Figure 19-3. Memory Sections



The size of the descriptor and write-back memory sections is dependant on most significant enabled DMA channel, as shown below:

$$Size = 128bits \cdot (MostSignificantEnabledChannelNumber + 1)$$

For memory optimization, it is recommended to always use the less significant DMA channels if not all channels are required.

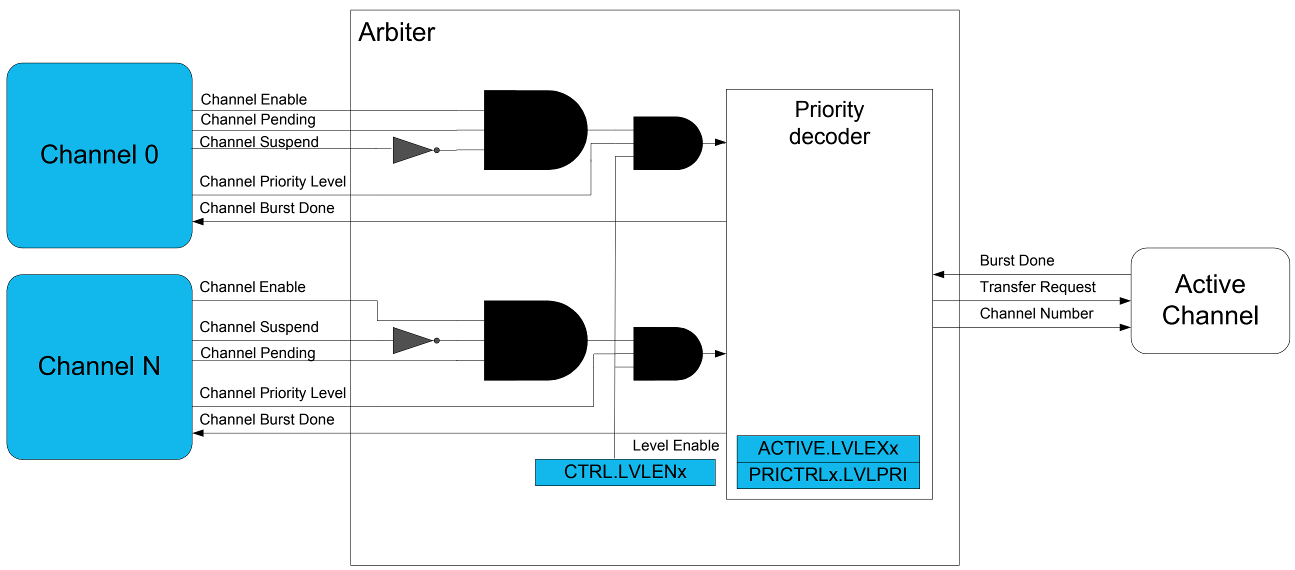
The descriptor and write-back memory sections can either be two separate memory sections, or they can share memory section (BASEADDR=WRBADDR). The benefit of having them in two separate sections, is that the same transaction for a channel can be repeated without having to modify the first transfer descriptor. The benefit of having descriptor memory and write-back memory in the same section is that it requires less SRAM. In addition, the latency from fetching the first descriptor of a transaction to the first burst transfer is executed, is reduced.

19.6.2.4 Arbitration

If a DMA channel is enabled and not suspended when it receives a transfer trigger, it will send a transfer request to the arbiter. When the arbiter receives the transfer request it will include the DMA channel in the queue of channels having pending transfers, and the corresponding Pending Channel x bit in the Pending Channels registers (PENDCH.PENDCHx) will be set. Dependent of the arbitration scheme, the arbiter will choose which DMA channel will be the next active channel. Refer to Figure 19-4. The active channel is the DMA channel being granted access to perform its next burst transfer. When the arbiter has granted a DMA channel access to the DMAC, the corresponding PENDCH.PENDCHx will be cleared. Depending on if the upcoming burst transfer is the first for the transfer request or not, the corresponding Busy Channel x bit in the Busy Channels register (BUSYCH.BUSYCHx) will either be set or remain one. When the channel has performed its granted burst transfer(s) it will either be fed into the queue of channels with pending transfers, set to be waiting for a new transfer trigger, it will be suspended or it will be disabled. This depends on the channel and block transfer configuration. If the DMA channel is fed into the queue of channels with pending transfers, the corresponding BUSYCH.BUSYCHx will remain one. If the DMA channel is set to wait for a new transfer trigger, suspended or disabled, the corresponding BUSYCH.BUSYCHx will be cleared.

If a DMA channel is suspended while it has a pending transfer, it will be removed from the queue of pending channels, but the corresponding PENDCH.PENDCHx will remain set. When the same DMA channel is resumed, it will be added to the queue of pending channels again. If a DMA channel gets disabled(CHCTRLA.ENABLE is zero) while it has a pending transfer, it will be removed from the queue of pending channels, and the corresponding PENDCH.PENDCHx will be cleared.

Figure 19-4. Arbiter overview

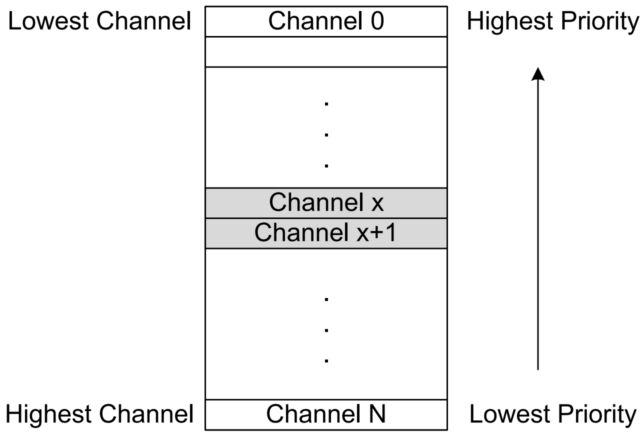


When a channel level is pending or the channel is transferring data, the corresponding Level Executing bit is set in the Active Channel and Levels register (ACTIVE.LVLEXx).

Each DMA channel supports a 4-level priority scheme. The priority level for a channel is configured by writing to the Channel Arbitration Level bit group in the Channel Control B register(CHCTRLB.LVL). As long as all priority levels are enabled, a channel with lower priority level number will have priority over a channel with higher priority level number. A priority level is enabled by writing the Priority Level x Enable bit in the Control register(CTRL.LVLENx) to one, for the corresponding level.

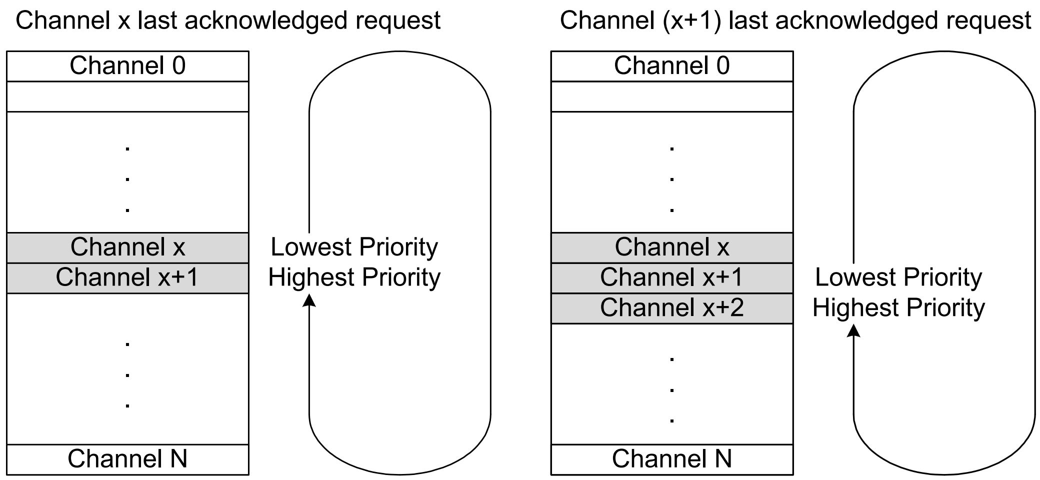
Within each priority level the DMAC's arbiter can be configured to prioritize statically or dynamically. For the arbiter to perform static arbitration within a priority level, the Level x Round-Robin Scheduling Enable bit in the Priority Control 0 register (PRICTRL0.RRLVLENx) has to be written to zero. When static arbitration is enabled (PRICTRL0.RRLVLENx is zero), the arbiter will prioritize a low channel number over a high channel number as shown in Figure 19-5. When using the static scheme there is a risk of high channel numbers never being granted access as the active channel. This can be avoided using a dynamic arbitration scheme.

Figure 19-5. Static priority



The dynamic arbitration scheme available in the DMAC is round-robin. Round-robin arbitration is enabled by writing PRICTRL0.RRLVLENx to one, for a given priority level x. With the round-robin scheme, the channel number of the last channel being granted access will have the lowest priority the next time the arbiter has to grant access to a channel within the same priority level, as shown in Figure 19-6. The channel number of the last channel being granted access as the active channel, will be stored in the Level x Channel Priority Number bit group in the Priority Control 0 register(PRICTRL0.LVLPRIx), for the corresponding priority level.

Figure 19-6. Round-robin scheduling



19.6.2.5 Data Transmission

Before the DMAC can perform a data transmission, a DMA channel has to be configured and enabled, its corresponding transfer descriptor has to be initialized and the arbiter has to grant the DMA channel access as the active channel.

Once the arbiter has granted a DMA channel access as the active channel (refer to Figure 19-1) the transfer descriptor for the DMA channel will be fetched from SRAM using the fetch bus, and stored in the internal memory for the active channel. Depending on if it is a new or ongoing block transfer, the transfer descriptor will either be fetched from the descriptor memory section (BASEADDR) or the write-back memory section (WRBADDR). By using the data transfer bus, the DMAC will read the data from the current source address and write it to the current destination address. For further details on how the current source and destination addresses are calculated, refer to "Addressing" on page 278.

The arbitration procedure is performed after each burst transfer. If the current DMA channel is granted access again, the block transfer counter (BTCNT) of the internal transfer descriptor will be decremented with the number of beats in a burst, and the active channel will perform a new burst transfer. If a different DMA channel than the current active channel is granted access, the BTCNT of the internal transfer descriptor will be decremented with the number of beats in a burst. The block transfer counter value will be written to the write-back section before the transfer descriptor of the newly granted DMA channel is fetched into the internal memory of the active channel. The optional output event, Beat, will be generated if configured and enabled.

When a block transfer has come to its end, BTCNT has reached zero, the Valid bit in the Block Transfer Control register will be written to zero in the internal transfer descriptor for the active channel before the entire transfer descriptor is written to the write-back memory. The optional interrupts, Channel Transfer Complete and Channel Suspend, and the optional output event, Block, will be generated if configured and enabled. If it was the last block transfer in a transaction, Next Address (DESCADDR) register will hold the value 0x00000000, and the DMA channel will either be suspended or disabled, depending on the configuration in the Block Action bit group in the Block Transfer Control register(BTCTRL.BLOCKACT). If the transaction has further block transfers pending, DESCADDR will hold the SRAM address to the next transfer descriptor to be fetched. The DMAC will fetch the next descriptor into the internal memory of the active channel and write its content to the write-back section for the channel, before the arbiter gets to choose the next active channel.

19.6.2.6 Transfer Triggers and Actions

A DMA transfer can be started only when a DMA transfer request is detected. A transfer request can be triggered from software, from peripheral, or from an event. There are dedicated Trigger Source selections for each DMA Channel Control B (CHCTRLB.TRIGSRC).

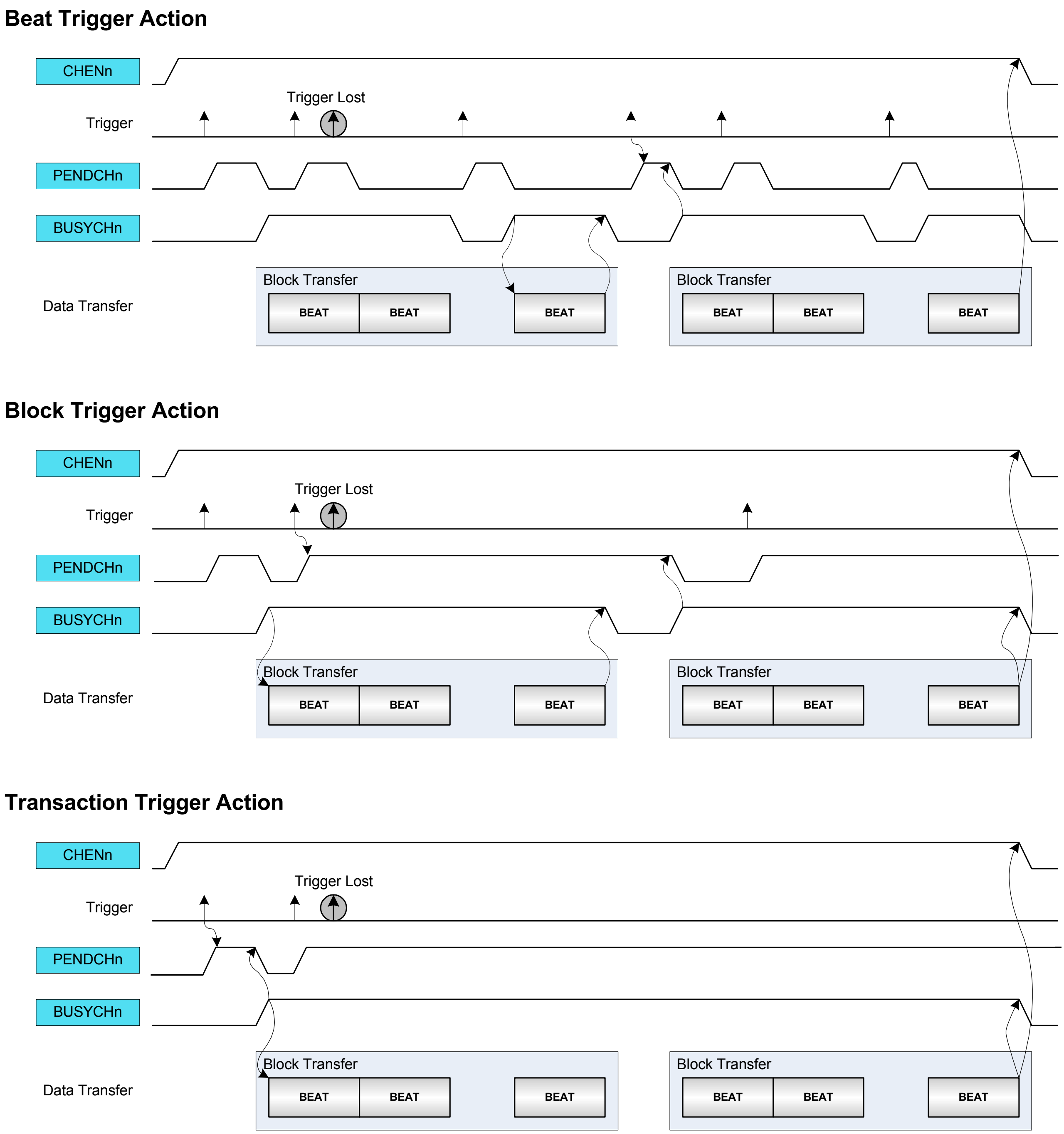
The trigger actions are available in the Trigger Action bit group in the Channel Control B register (CHCTRLB.TRIGACT). By default, a trigger starts a block transfer operation. If a single descriptor is defined for a channel, the channel is automatically disabled when a block transfer is complete. If a list of linked descriptors is defined for a channel, the channel is automatically disabled if the last descriptor in the list is executed or the channel will be waiting for the next block transfer trigger if the list still has descriptors to execute. When enabled again, the channel will wait for the next block transfer trigger. It is also possible to select the trigger to start beat or transaction transfers instead of a block transfer.

If the trigger source generates a transfer request during an ongoing transfer, this will be kept pending (CHSTATUS.PEND is one), and the transfer can start when the ongoing one is done. Only one pending transfer can be kept, and so if the trigger source generates more transfer requests when one is already pending, these will be lost. All channels pending status flags are also available in the Pending Channels register (PENDCH).

When the transfer starts, the corresponding Channel Busy status flag is set in Channel Status register (CHSTATUS.BUSY). When the trigger action is complete, the Channel Busy status flag is cleared. All channels busy status flags are also available in the Busy Channels register (BUSYCH) in DMAC.

Figure 19-7 on page 278 shows an example where triggers are used with two linked block descriptors.

Figure 19-7. Trigger action and transfers



19.6.2.7 Addressing

For the DMAC to know from where to where it should transfer the data, each block transfer needs to have a source and destination address defined. The source address can be set by writing the Transfer Source Address (SRCADDR) register, and the destination address can be set by writing the Transfer Destination Address (SRCADDR) register.

The addressing of this DMAC module can be static or incremental, for either source or destination of a block transfer, or both.

Incrementation for the source address of a block transfer is enabled by writing the Source Address Incrementation Enable bit in the Block Transfer Control register (BTCTRL.SRCINC) to one. The step size of the incrementation is configurable and can be chosen by writing the Step Selection bit in the Block Transfer Control register(BTCTRL.STEPSEL) to one, and the Address Increment Step Size bit group in the Block Transfer Control register (BTCTRL.STEPSIZE), to the desired step size. If BTCTRL.STEPSEL is zero, the step size for the source incrementation will be the size of one beat.

When source address incrementation is configured (BTCTRL.SRCINC is one), SRCADDR must be set to the source address of the last beat transfer in the block transfer. The source address should be calculated as follows:

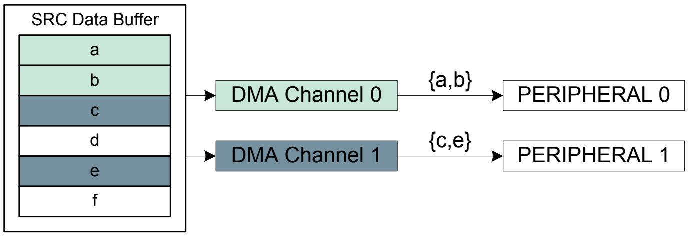
$$SRCADDR = SRCADDR_{START} + BTCNT \cdot (BEATSIZE + 1) \cdot 2^{STEPSIZE}$$, where BTCTRL.STEPSEL is one

$$SRCADDR = SRCADDR_{START} + BTCNT \cdot (BEATSIZE + 1)$$, where BTCTRL.STEPSEL is zero

- SRCADDRSTART is the source address of the first beat transfer in the block transfer
- BTCNT is the initial number of beats remaining in the block transfer
- BEATSIZE is the configured number of bytes in a beat
- STEPSIZE is the configured number of beats for each incrementation

Figure 19-8 shows an example where DMA channel 0 is configured to increment the source address by one beat (BTCTRL.SRCINC is one) after each beat transfer, and DMA channel 1 is configured to increment source address by two beats (BTCTRL.SRCINC is one, BTCTRL.STEPSEL is one, and BTCTRL.STEPSIZE is 0x1). As the destination address for both channels are peripherals, destination incrementation is disabled(BTCTRL.DSTINC is zero).

Figure 19-8. Source address increment



Incrementation for the destination address of a block transfer is enabled by writing the Destination Address Incrementation Enable bit in the Block Transfer Control register (BTCTRL.DSTINC) to one. The step size of the incrementation is configurable and can be chosen by writing BTCTRL.STEPSEL to zero, and BTCTRL.STEPSIZE to the desired step size. If BTCTRL.STEPSEL is one, the step size for the destination incrementation will be the size of one beat.

When destination address incrementation is configured (BTCTRL.DSTINC is one), SRCADDR must be set to the destination address of the last beat transfer in the block transfer. The destination address should be calculated as follows:

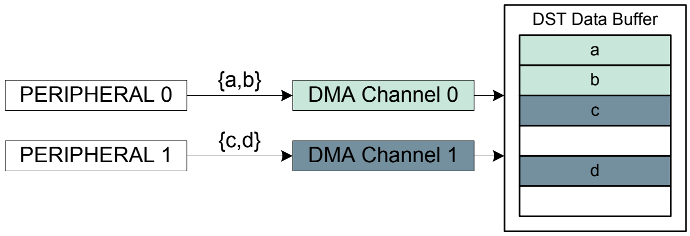
$$DSTADDR = DSTADDR_{START} + BTCNT \cdot (BEATSIZE + 1) \cdot 2^{STEPSIZE}$$, where BTCTRL.STEPSEL is zero

$$DSTADDR = DSTADDR_{START} + BTCNT \cdot (BEATSIZE + 1)$$, where BTCTRL.STEPSEL is one

- DSTADDRSTART is the destination address of the first beat transfer in the block transfer
- BTCNT is the initial number of beats remaining in the block transfer
- BEATSIZE is the configured number of bytes in a beat
- STEPSIZE is the configured number of beats for each incrementation

Figure 19-9 shows an example where DMA channel 0 is configured to increment destination address by one beat (BTCTRL.DSTINC is one) and DMA channel 1 is configured to increment destination address by two beats (BTCTRL.DSTINC is one, BTCTRL.STEPSEL is zero, and BTCTRL.STEPSIZE is 0x1). As the source address for both channels are peripherals, source incrementation is disabled(BTCTRL.SRCINC is zero).

Figure 19-9. Destination address increment



19.6.2.8 Error Handling

If a bus error is received from AHB slave during a DMA data transfer, the corresponding active channel is disabled and the corresponding Channel Transfer Error Interrupt flag in the Channel Interrupt Status and Clear register (CHINTFLAG.TERR) is set. If transfer error interrupt is enabled, optional error interrupt is generated. The transfer counter will not be decremented and its current value is written-back in the write-back memory section before the channel is disabled.

When the DMAC fetches an invalid descriptor (BTCTRL.VALID is zero) or when the channel is resumed and the DMA fetches the next descriptor with null address (DESCADDR is 0x00000000), the corresponding channel operation is suspended, the Channel Suspend Interrupt Flag in the Channel Interrupt Flag Status and Clear register (CHINTFLAG.SUSP) is set and the Channel Fetch Error bit in the Channel Status register (CHSTATUS.FERR) is set. If enabled, optional suspend interrupt is generated.

19.6.3 Additional Features

19.6.3.1 Linked Descriptors

A transaction can either consist of a single block transfer, or it can consist of several block transfers. When a transaction consist of several block transfers it is called linked descriptors.

Figure 19-3 shows how linked descriptors work. When the first block transfer is completed on DMA channel 0, the DMAC fetches the next transfer descriptor which is pointed to by the value stored in the Next Descriptor Address (DESCADDR) register, in the first transfer descriptor. Fetching the next transfer descriptor (DESCADDR) is continued until the last transfer descriptor. When the block transfer for the last transfer descriptor is executed and DESCADDR=0x00000000, the transaction is terminated. For further details on how the next descriptor is fetched from SRAM, refer to "Data Transmission" on page 276.

Adding Descriptor to the End of a List

To add a new descriptor at the end of the descriptor list, create the descriptor in SRAM, with DESCADDR=0x00000000 indicating it is the new last descriptor in the list, and modify the DESCADDR value of the current last descriptor to the address of the newly created descriptor.

Modifying a Descriptor in a List

In order to add descriptors to a list, the following actions must be performed:

1. Before enabling a channel, the Suspend interrupt must be enabled
2. Reserve memory space addresses to configure a new descriptor
3. Configure the new descriptor
   - Set the next descriptor address (DESCADDR)
   - Set the destination address (DSTADDR)
   - Set the source address (SRCADDR)
   - Configure the block transfer control (BTCTRL) including
     - Optionally enable the Suspend block action
     - Set the descriptor VALID bit
4. In the existing list and for the descriptor which has to be updated, set the VALID bit to zero
5. Read DESCADDR from the Write-Back memory

- If the DMA has not already fetched the descriptor which requires changes:
  - Update the DESCADDR location of the descriptor from the List
  - Optionally clear the Suspend block action
  - Set the descriptor VALID bit to one
  - Optionally enable the Resume software command
- If the DMA is executing the same descriptor as the one which requires changes:
  - Set the Channel Suspend software command and wait for the Suspend interrupt
  - Update the Write-Back next descriptor address (DESCRADDR)
  - Clear the interrupt sources and set the Resume software command
  - Update the DESCADDR location of the descriptor from the List
  - Optionally clear the Suspend block action
  - Set the descriptor VALID bit to one

6. Go to step 3 if needed

Adding a Descriptor Between Existing Descriptors

To insert a descriptor C between 2 existing descriptors (A & B), the descriptor currently executed by the DMA must be identified.

1. If DMA is executing descriptor B, descriptor C cannot be inserted.
2. If DMA has not started to execute descriptor A, follow the steps:
   a. Set the descriptor A VALID bit to 0
   b. Set the DESCADDR value of descriptor A to point descriptor C instead of descriptor B
   c. Set the DESCADDR value of descriptor C to point descriptor B
   d. Set the descriptor A VALID bit to 1.
3. If DMA is executing descriptor A,
   a. Apply the software suspend command to the channel and
   b. Perform steps 2a through 2d

Apply the software resume command to the channel.

19.6.3.2 Channel Suspend

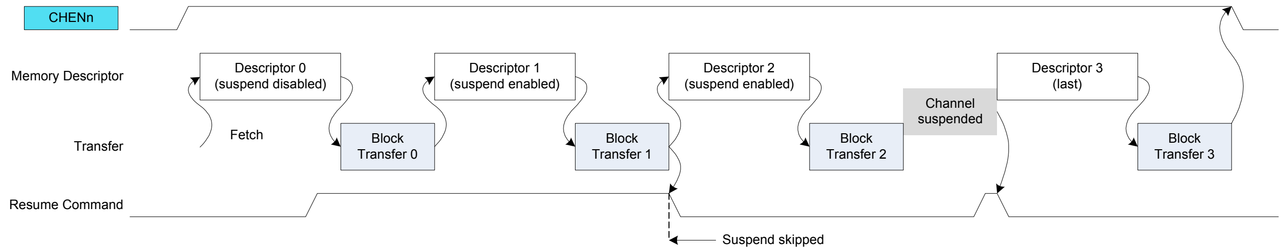
The channel operation can be suspended at anytime by software, by setting the Suspend command in Command bit field of Channel Control B register (CHCTRLB.CMD). When the ongoing burst transfer is completed, the channel operation is suspended and the suspend command is automatically cleared.
It is also possible to suspend a channel operation after a block transfer completes. The software must set the Suspend Block Action in the corresponding Block Transfer Control location (BTCTRL.BLOCKACT). When the block transfer is completed, the channel operation is suspended. The channel is kept enabled, can receive transfer triggers, but it will be removed from the arbitration scheme. The channel will automatically suspend the operation if an invalid transfer control descriptor is fetched from system memory (BTCTRL.VALID=0). The Channel Fetch Error bit in the Channel Status register (CHSTATUS.FERR) is set when an invalid descriptor is fetched. Only an enabled channel can be suspended. If the channel is disabled when suspended, the internal suspend command is cleared. When suspended, the Channel Suspend Interrupt flag in the Channel Interrupt Status and Clear register (CHINTFLAG.SUSP) is set and optional suspend interrupt is generated.

For more details on transfer descriptors, refer to "Transfer Descriptors" on page 273.

19.6.3.3 Channel Resume and Next Suspend Skip

A channel operation can be resumed by software by setting the Resume command in Command bitfield of Channel Control B register (CHCTRLB.CMD). If the channel is already suspended, the channel operation resumes from where it previously stopped when the Resume command is detected. When the Resume command is issued before the channel is suspended, the next suspend action is skipped and the channel continues the normal operation.

Figure 19-10.Channel suspend/resume operation



19.6.3.4 Event Input Actions

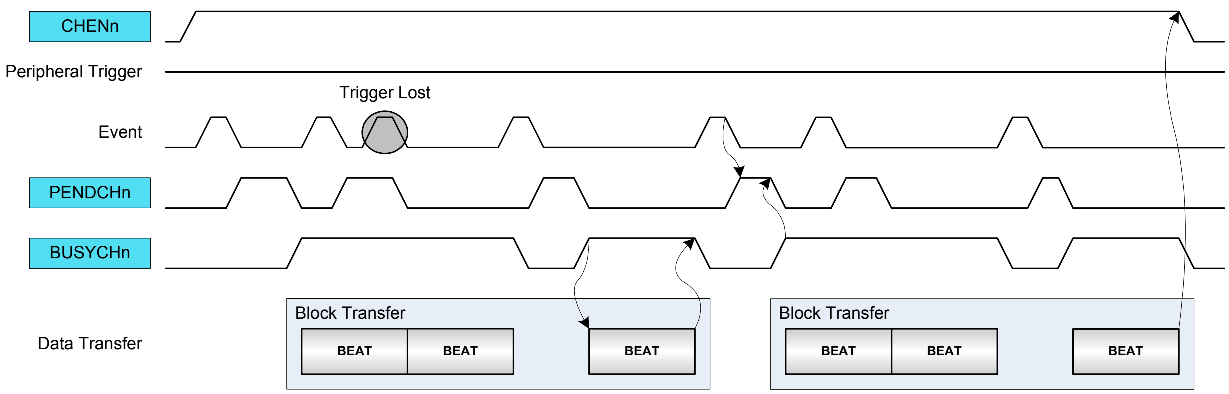
The event input actions are available only for channels supporting event inputs. For details on channels with event input support, refer to Table 23-6 and Table 23-4.

The Event Actions bits in the Channel Control B register (CHCTRLB.EVACT) specify the actions the DMA will take on an input event. Before using event actions, the event controller must be configured first and the corresponding Channel Event Input Enable bit (CHCTRLB.EVIE) must be set. The DMA supports only resynchronized events. For details on how to configure the resynchronized event path, refer to the Event System.

Normal transfer: When this event action is selected for a channel, the event input is used to trigger a beat or burst transfer on peripherals.

The transfer trigger is selected by setting the Trigger Source bits in Channel Control B register to zero (CHCTRLB.TRIGSRC). The event is acknowledged as soon as the event is received. When received, the Channel Pending status bit is set (CHSTATUS.PEND). If the event is received while the channel is pending, the event trigger is lost. Figure 19-11 shows an example where beat transfers are enabled by internal events.

Figure 19-11.Beat event trigger action

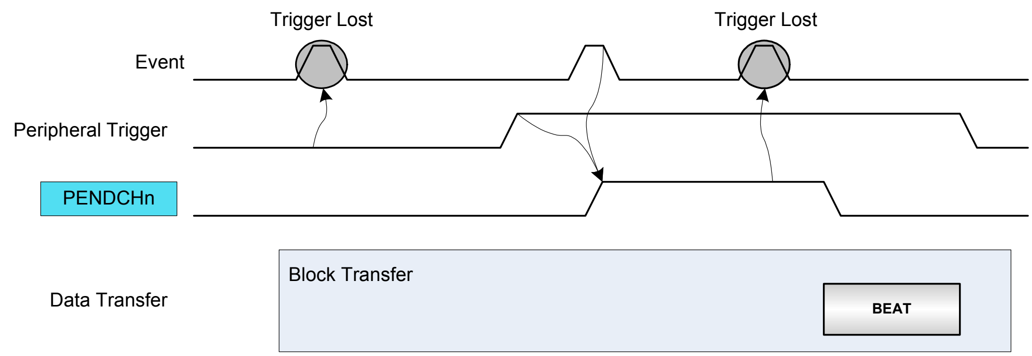


Periodic transfers: When this event action is selected for a channel, the event input is used to trigger a transfer on peripherals with pending transfer requests. This type of event is intended to be used with peripheral triggers for example, for timed communication protocols or periodic transfers between peripherals, as examples. The peripheral trigger is selected by the Trigger Source bits in the Channel Control B register (CHCTRLB.TRIGSRC).

The event is acknowledged as soon as the event is received. The peripheral trigger request is stored internally when the previous trigger action is completed (i.e. channel is not pending) and when an active event is received. If the peripheral trigger is active, the DMA will wait for an event before the peripheral trigger is internally registered. When both event and peripheral transfer trigger are active, the Channel Pending status bit is set (CHSTATUS.PEND). A software trigger will now trigger a transfer.

Figure 19-12 shows an example where the peripheral beat transfers are enabled by periodic events.

Figure 19-12.Periodic event with beat peripheral triggers

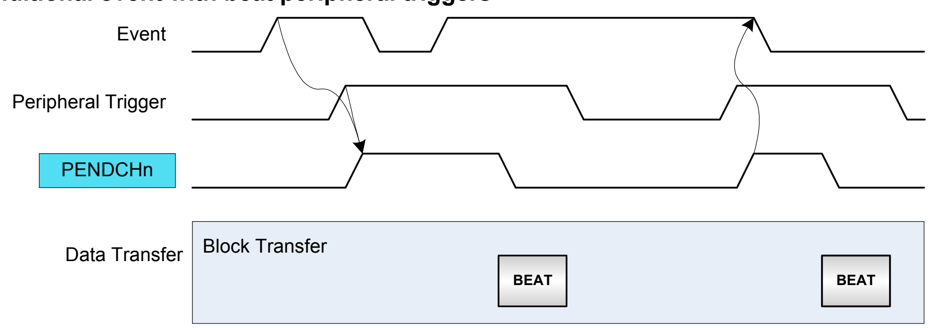


Conditional transfer: When the conditional transfer event action is selected, the event input is used to trigger a conditional transfer on peripherals with pending transfer requests. As example, this type of event can be used for peripheral to peripheral transfers, where one peripheral is source of event and the second peripheral is source of DMA trigger.

The peripheral Trigger Source must be set in Channel Control B register (CHCTRLB.TRIGSRC). Each peripheral trigger is stored internally when the event is received. When the peripheral trigger is stored internally, the Channel Pending status bit is set (CHSTATUS.PEND) and the event is acknowledged. A software trigger will now trigger a transfer.

Figure 19-13 shows an example where conditional event is enabled with peripheral beat trigger requests.

Figure 19-13.Conditional event with beat peripheral triggers

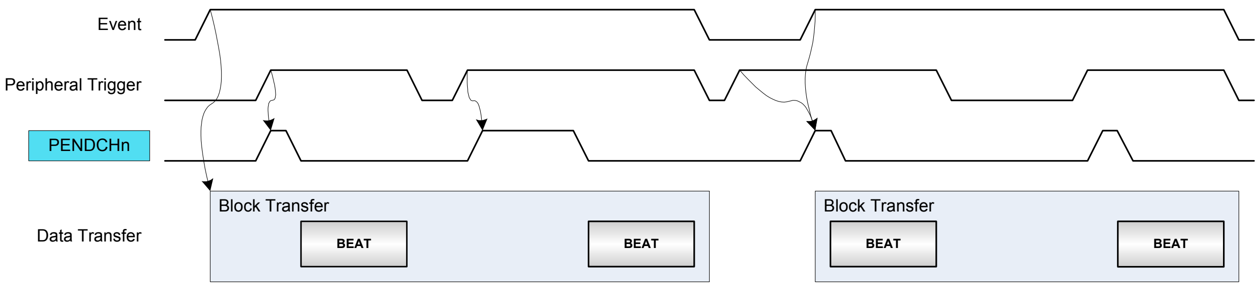


Conditional block transfer: When the conditional block event action is selected, the event input is used to trigger a conditional block transfer on peripherals. The peripheral Trigger Source must be set in Channel Control B register (CHCTRLB.TRIGSRC).

Before starting transfers within a block, an event must be received. When received, the event is acknowledged when the block transfer is completed. A software trigger will trigger a transfer.

Figure 19-14 shows an example where conditional event block transfer is enabled with peripheral beat trigger requests.

Figure 19-14.Conditional block transfer with beat peripheral triggers



Channel suspend: When the channel suspend event action is selected, the event input is used to suspend an ongoing channel operation. The event is acknowledged when the current AHB access is completed. For further details on channel suspend, refer to "Channel Suspend" on page 281.

Channel resume: When the channel resume event action is selected, the event input is used to resume a suspended channel operation. The event is acknowledged as soon as the event is received and the Channel Suspend Interrupt Flag (CHINTFLAG.SUSP) is cleared. For further details on channel suspend, refer to "Channel Suspend" on page 281.

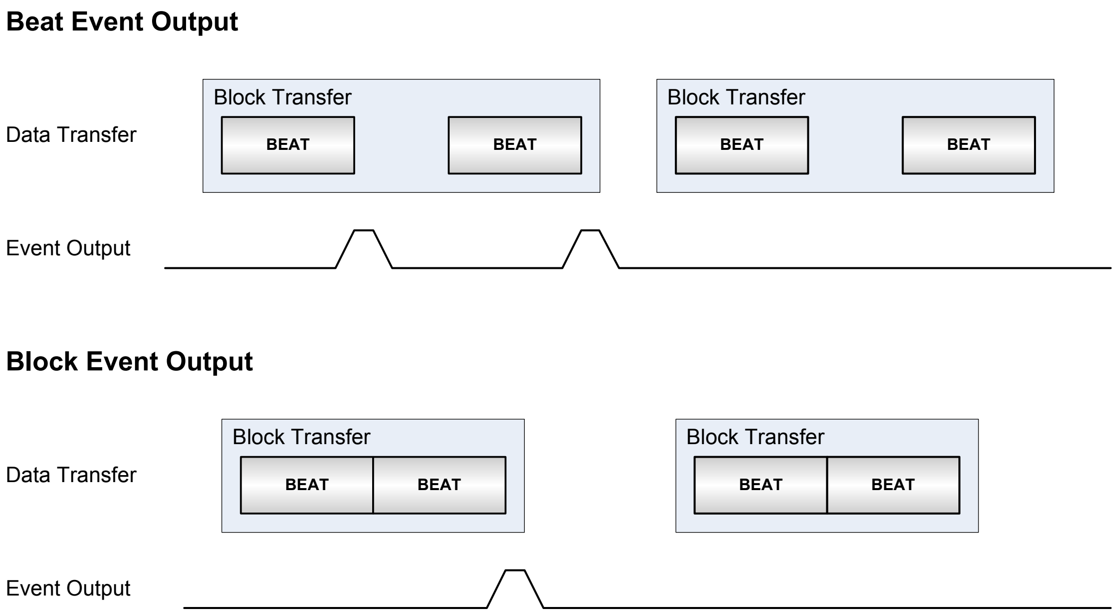
Skip next block suspend: This event can be used to skip the next block suspend action. If the channel is suspended before the event rises, the channel operation is resumed and the event is acknowledged. If the event rises before a suspend block action is detected, the event is kept until the next block suspend detection. When the block transfer is completed, the channel continues the operation (not suspended) and the event is acknowledged.

19.6.3.5 Event Output Selections

The event output selections are available only for channels supporting event outputs. The pulse width of an event output from a channel is one AHB clock cycle.

The Channel Event Output Enable can be set in Control B register (CHCTRLB.EVOE). The Event Output Selection is available in each Descriptor Block Control location (BTCTRL.EVOSEL). It is possible to generate events after each beat, burst or block transfer. To enable an event when the transaction is complete, the block event selection must be set in the last transfer descriptor only. Figure 19-15 shows an example where the event output generation is enabled in the first block transfer, and disabled in the second block.

Figure 19-15.Event output generation



19.6.3.6 Aborting Transfers

Transfers on any channel can be gracefully aborted by software, by disabling the corresponding DMA channel. It is also possible to abort all ongoing or pending transfers, by disabling the DMAC.

When DMAC disable request is detected:

- Active channel with ongoing transfers will be disabled when the ongoing beat access is completed and the Write-Back memory section is updated. This prevents transfer corruption before the channel is disabled.
- All other enabled channels will be disabled in the next clock cycle.

The corresponding Channel Enable bit in the Channel Control A register (CHCTRLA.ENABLE) is read as zero when the channel is disabled.

The corresponding DMAC Enable bit in the Control register (CTRL.DMAENABLE) is read as zero when the entire DMAC module is disabled.

19.6.3.7 CRC Operation

A cyclic redundancy check (CRC) is an error detection technique used to find accidental errors in data. It is commonly used to determine whether the data during a transmission, or data present in data and programme memories has been corrupted or not. A CRC takes a data stream or a block of data as input and generates a 16- or 32-bit output that can be appended to the data and used as a checksum. When the same data are later received or read, the device or application repeats the calculation. If the new CRC result does not match the one calculated earlier, the block contains a data error. The application will then detect this and may take a corrective action, such as requesting the data to be sent again or simply not using the incorrect data.

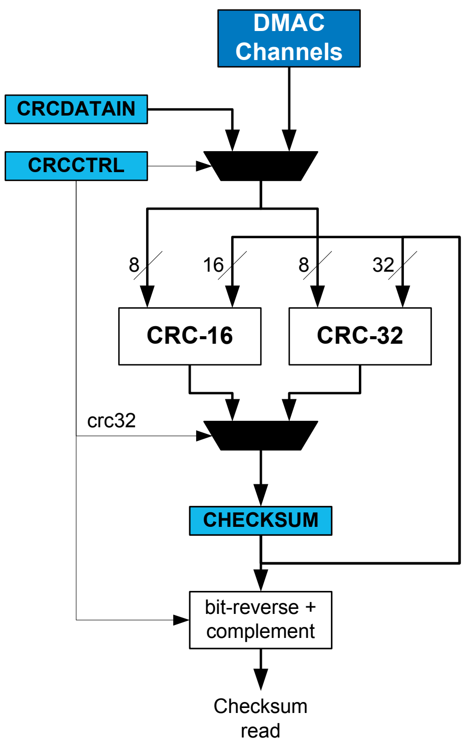
Typically, a CRC-n applied to a data block of arbitrary length will detect any single error burst not longer than n bits (any single alteration that spans no more than n bits of the data), and will detect the fraction $1-2^{-n}$ of all longer error bursts. The CRC module in DMAC supports two commonly used CRC polynomials: CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3).

- CRC-16:
  - Polynomial: $x^{16} + x^{12} + x^5+1$
  - Hex value: 0x1021
- CRC-32:
  - Polynomial: $x^{32} +x^{26}+x^{23} +x^{22} +x^{16} +x^{12} +x^{11} +x^{10} +x^8 +x^7 +x^5 +x^4 +x^2 +x+1$
  - Hex value: 0x04C11DB7

The data source for the CRC module must be selected in software as either the DMA channels or the APB bus interface. The CRC module then takes data input from the selected source and generates a checksum based on these data. The checksum is available in the CRC Checksum register (CRCCHKSUM). When CRC-32 polynomial is used, the final checksum read is bit reversed and complemented, as shown in Figure 19-16 on page 286.

The CRC polynomial to be used is configurable, and the default setting is CRC-16. The CRC module operates on byte only. When the DMA is used as data source for the CRC module, the DMA channel beat size setting will be used. When used with APB bus interface, the application must set the CRC Beat Size bit field of CRC Control register (CRCCTRL.CRCBEATSIZE). 8-, 16- or 32-bit bus transfer access type is supported. The corresponding number of bytes will be written in the CRCDATAIN register and the CRC module will operate on the input data in a byte by byte manner.

Figure 19-16.CRC generator block diagram



CRC on DMA data: CRC-16 or CRC-32 calculations can be performed on data passing through any DMA channel. Once a DMA channel is selected as the source, the CRC module will continuously generate the CRC on the data passing through the DMA channel. The checksum is available for readout once the DMA transaction is completed or aborted. A CRC can also be generated on SRAM, Flash or I/O memory by passing these data through a DMA channel. If the latter is done, the destination register for the DMA data can be the data input (CRCDATAIN) register in the CRC module.

CRC using the I/O interface: Before using the CRC module with the I/O interface, the application must set the CRC Beat Size bits in the CRC Control register (CRCCTRL.CRCBEATSIZE). 8/16/32-bit bus transfer type can be selected.

CRC can be performed on any data by loading them into the CRC module using the CPU and writing the data to the CRCDATAIN register. Using this method, an arbitrary number of bytes can be written to the register by the CPU, and CRC is done continuously for each byte. This means if a 32-bit data is written to the CRCDATAIN register the CRC module takes 4 cycles to calculate the CRC. The CRC complete is signaled by the CRCBUSY bit in the CRCSTATUS register. New data can be written only when CRCBUSY flag is not set.

19.6.4 DMA Operation

Not applicable.

19.6.5 Interrupts

The DMAC has the following interrupt sources:

- Transfer Complete (TCMPL): Indicates that a block transfer is completed on the corresponding channel. Refer to "Data Transmission" on page 276 for details.
- Transfer Error (TERR): Indicates that a bus error has occurred during a burst transfer, or that an invalid descriptor has been fetched. Refer to "Error Handling" on page 280 for details.
- Channel Suspend (SUSP): Indicates that the corresponding channel has been suspended. Refer to "Channel Suspend" on page 281 and "Data Transmission" on page 276 for details.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Channel Interrupt Flag Status and Clear (CHINTFLAG) register is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Channel Interrupt Enable Set (CHINTENSET) register, and disabled by writing a one to the corresponding bit in the Channel Interrupt Enable Clear (CHINTENCLR) register. An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled, the DMAC is reset or the corresponding DMA channel is reset. See CHINTFLAG for details on how to clear interrupt flags. All interrupt requests are ORed together on system level to generate one combined interrupt request to the NVIC. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

The user must read the Channel Interrupt Status (INTSTATUS) register to identify the channels with pending interrupts and must read the Channel Interrupt Flag Status and Clear (CHINTFLAG) register to determine which interrupt condition is present for the corresponding channel. It is also possible to read the Interrupt Pending register (INTPEND), which provides the lowest channel number with pending interrupt and the respective interrupt flags.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

19.6.6 Events

The DMAC can generate the following output events:

- Channel (CH): Generated when a block transfer for a given channel has been completed, or when a beat transfer within a block transfer for a given channel has been completed. Refer to "Event Output Selections" on page 285 for details.

Writing a one to the Channel Control B Event Output Enable bit (CHCTRLB.EVOE) enables the corresponding output event configured in the Event Output Selection bit group in the Block Transfer Control register (BTCTRL.EVOSEL). Writing a zero to CHCTRLB.EVOE disables the corresponding output event. Refer to "EVSYS – Event System" on page 402 for details on configuring the event system.

The DMAC can take the following actions on an input event:

- Transfer and Periodic Transfer Trigger (TRIG): normal transfer or periodic transfers on peripherals are enabled
- Conditional Transfer Trigger (CTRIG): conditional transfers on peripherals are enabled
- Conditional Block Transfer Trigger (CBLOCK): conditional block transfers on peripherals are enabled

- Channel Suspend Operation (SUSPEND): suspend a channel operation
- Channel Resume Operation (RESUME): resume a suspended channel operation
- Skip Next Block Suspend Action (SSKIP): skip the next block suspend transfer condition

Writing a one to the Channel Control B Event Input Enable bit (CHCTRLB.EVIE) enables the corresponding action on input event. Writing a zero to this bit disables the corresponding action on input event. Note that several actions can be enabled for incoming events. If several events are connected to the peripheral, any enabled action will be taken for any of the incoming events. For further details on event input actions, refer to "Event Input Actions" on page 282. Refer to the Event System chapter for details on configuring the event system.

19.6.7 Sleep Mode Operation

In standby sleep mode, the DMAC will be internally disabled, but maintains its current configuration.

19.6.8 Synchronization

Not applicable.

19.7 Register Summary

Table 19-1. DMAC Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0						CRCENABLE	DMAENABLE	SWRST
0x01		15:8					LVLEN3	LVLEN2	LVLEN1	LVLEN0
0x02	CRCCTRL	7:0					CRCPOLY[1:0]		CRCBEATSIZE[1:0]	
0x03		15:8			CRCSRC[5:0]					
0x04	CRCDATAIN	7:0	CRCDATAIN[7:0]							
0x05		15:8	CRCDATAIN[15:8]							
0x06		23:16	CRCDATAIN[23:16]							
0x07		31:24	CRCDATAIN[31:24]							
0x08	CRCCHKSUM	7:0	CRCCHKSUM[7:0]							
0x09		15:8	CRCCHKSUM[15:8]							
0x0A		23:16	CRCCHKSUM[23:16]							
0x0B		31:24	CRCCHKSUM[31:24]							
0x0C	CRCSTATUS	7:0							CRCZERO	CRCBUSY
0x0D	DBGCTRL	7:0								DBGRUN
0x0E	QOSCTRL	7:0			DQOS[1:0]		FQOS[1:0]		WRBQOS[1:0]	
0x0F	Reserved									
0x10	SWTRIGCTRL	7:0			SWTRIG5	SWTRIG4	SWTRIG3	SWTRIG2	SWTRIG1	SWTRIG0
0x11		15:8								
0x12		23:16								
0x13		31:24								
0x14	PRICTRL0	7:0	RRLVLEN0					LVLPRI0[2:0]		
0x15		15:8	RRLVLEN1					LVLPRI1[2:0]		
0x16		23:16	RRLVLEN2					LVLPRI2[2:0]		
0x17		31:24	RRLVLEN3					LVLPRI3[2:0]		
0x18 ... 0x1F	Reserved									
0x20	INTPEND	7:0						ID[2:0]		
0x21		15:8	PEND	BUSY	FERR			SUSP	TCMPL	TERR
0x22	Reserved									
0x23	Reserved									
0x24	INTSTATUS	7:0			CHINT5	CHINT4	CHINT3	CHINT2	CHINT1	CHINT0
0x25		15:8								
0x26		23:16								
0x27		31:24								
0x28	BUSYCH	7:0			BUSYCH5	BUSYCH4	BUSYCH3	BUSYCH2	BUSYCH1	BUSYCH0
0x29		15:8								
0x2A		23:16								
0x2B		31:24								

Offset	Name	Bit Pos.								
0x2C	PENDCH	7:0			PENDCH5	PENDCH4	PENDCH3	PENDCH2	PENDCH1	PENDCH0
0x2D		15:8								
0x2E		23:16								
0x2F		31:24								
0x30	ACTIVE	7:0					LVLEX3	LVLEX2	LVLEX1	LVLEX0
0x31		15:8	ABUSY			ID[4:0]				
0x32		23:16	BTCNT[7:0]							
0x33		31:24	BTCNT[15:8]							
0x34	BASEADDR	7:0	BASEADDR[7:0]							
0x35		15:8	BASEADDR[15:8]							
0x36		23:16	BASEADDR[23:16]							
0x37		31:24	BASEADDR[31:24]							
0x38	WRBADDR	7:0	WRBADDR[7:0]							
0x39		15:8	WRBADDR[15:8]							
0x3A		23:16	WRBADDR[23:16]							
0x3B		31:24	WRBADDR[31:24]							
0x3C ... 0x3E	Reserved									
0x3F	CHID	7:0						ID[2:0]		
0x40	CHCTRLA	7:0							ENABLE	SWRST
0x41 ... 0x43	Reserved									
0x44	CHCTRLB	7:0		LVL[1:0]		EVOE	EVIE	EVACT[2:0]		
0x45		15:8				TRIGSRC[4:0]				
0x46		23:16	TRIGACT[1:0]							
0x47		31:24							CMD[1:0]	
0x48 ... 0x4B	Reserved									
0x4C	CHINTENCLR	7:0						SUSP	TCMPL	TERR
0x4D	CHINTENSET	7:0						SUSP	TCMPL	TERR
0x4E	CHINTFLAG	7:0						SUSP	TCMPL	TERR
0x4F	CHSTATUS	7:0						FERR	BUSY	PEND

Table 19-2. DMAC SRAM Register Summary - Descriptor/Write-Back Memory Section

Offset	Name	Bit Pos.								
0x00	BTCTRL	7:0				BLOCKACT[1:0]		EVOSEL[1:0]		VALID
0x01		15:8	STEPSIZE[2:0]			STEPSEL	DSTINC	SRCINC	BEATSIZE[1:0]	
0x02	BTCNT	7:0	BTCNT[7:0]							
0x03		15:8	BTCNT[15:8]							

Offset	Name	Bit Pos.								
0x04	SRCADDR	7:0	SRCADDR[7:0]							
0x05		15:8	SRCADDR[15:8]							
0x06		23:16	SRCADDR[23:16]							
0x07		31:24	SRCADDR[31:24]							
0x08	DSTADDR	7:0	DSTADDR[7:0]							
0x09		15:8	DSTADDR[15:8]							
0x0A		23:16	DSTADDR[23:16]							
0x0B		31:24	DSTADDR[31:24]							
0x0C	DESCADDR	7:0	DESCADDR[7:0]							
0x0D		15:8	DESCADDR[15:8]							
0x0E		23:16	DESCADDR[23:16]							
0x0F		31:24	DESCADDR[31:24]							

19.8 Register Description

Registers can be 8, 16, or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Please refer to "Register Access Protection" on page 270 for details.

Some registers are enable-protected, meaning they can only be written when the DMAC is disabled. Enable-protection is denoted by the Enable-Protected property in each individual register description.

19.8.1 DMAC Registers

19.8.1.1 Control

Name: CTRL

Offset: 0x00

Reset: 0x0000

Access: Read-Write

Property: -

Bit	15	14	13	12	11	10	9	8
					LVLEN3	LVLEN2	LVLEN1	LVLEN0
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						CRCENABLE	DMAENABLE	SWRST
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:8 – LVLENx [x=3..0]: Priority Level x Enable**
  0: Transfer requests for Priority level x will not be handled.

  1: Transfer requests for Priority level x will be handled.

  When this bit is set, all requests with the corresponding level will be fed into the arbiter block. When cleared, all requests with the corresponding level will be ignored.

  For details on arbitration schemes, refer to "Arbitration" on page 275 section.

  These bits are not enable-protected.

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – CRCENABLE: CRC Enable**
  0: The CRC module is disabled.

  1: The CRC module is enabled.

  Writing a zero to this bit will disable the CRC module if the CRC Status Busy bit in the CRC Status register (CRC-STATUS.CRCBUSY) is zero. If the CRCSTATUS.CRCBUSY is one, the write will be ignored and the CRC module will not be disabled.

  Writing a one to this bit will enable the CRC module.

  This bit is not enable-protected.

- **Bit 1 – DMAENABLE: DMA Enable**
  0: The peripheral is disabled.

  1: The peripheral is enabled.

Writing a zero to this bit during an ongoing transfer, the bit will not be cleared until the internal data transfer buffer is empty and the DMA transfer is aborted. The internal data transfer buffer will be empty once the ongoing burst transfer is completed.

Writing a one to this bit will enable the DMA module.

This bit is not enable-protected.

- **Bit 0 – SWRST: Software Reset**

  0: There is no reset operation ongoing.

  1: The reset operation is ongoing.

  Writing a zero to this bit has no effect.

  Writing a one to this bit when both the DMAC and the CRC module are disabled (DMAENABLE and CRCENABLE is zero), resets all registers in the DMAC, except DBGCTRL, to their initial state If either the DMAC or CRC module is enabled, the reset request will be ignored and the DMAC will return an access error.

19.8.1.2 CRC Control

Name: CRCCTRL

Offset: 0x02

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
			CRCSRC[5:0]					
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					CRCPOLY[1:0]		CRCBEATSIZE[1:0]	
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:14 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 13:8 – CRCSRC[5:0]: CRC Input Source**
  These bits select the input source for generating the CRC, as shown in Table 19-3. The selected source is locked until either the CRC generation is completed or the CRC module is disabled. This means the CRCSRC cannot be modified when the CRC operation is ongoing. The lock is signaled by the CRCBUSY status bit. CRC generation complete is generated and signaled from the selected source when used with the DMA channel.

Table 19-3. CRC Input Source

CRCSRC[5:0]	Name	Description
0x0	NOACT	No action
0x1	IO	I/O interface
0x2-0x1f		Reserved
0x20-0x3F	CHN	DMA channel n
0x21-0x3f		Reserved

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 3:2 – CRCPOLY[1:0]: CRC Polynomial Type**
  These bits select the CRC polynomial type, as shown in Table 19-4.

Table 19-4. **CRC Polynomial Type**

CRCPOLY[1:0]	Name	Description
0x0	CRC16	CRC-16 (CRC-CCITT)
0x1	CRC32	CRC32 (IEEE 802.3)
0x2-0x3		Reserved

- **Bits 1:0 – CRCBEATSIZE[1:0]: CRC Beat Size**
  These bits define the size of the data transfer for each bus access when the CRC is used with I/O interface, as shown in Table 19-5.

Table 19-5. **CRC Beat Size**

CRCBEATSIZE[1:0]	Name	Description
0x0	BYTE	Byte bus access
0x1	HWORD	Half-word bus access
0x2	WORD	Word bus access
0x3		Reserved

19.8.1.3 CRC Data Input

Name: CRCDATAIN

Offset: 0x04

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	CRCDATAIN[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCDATAIN[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCDATAIN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCDATAIN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – CRCDATAIN[31:0]: CRC Data Input**
  These bits store the data for which the CRC checksum is computed. After the CRCDATAIN register has been written, the number of cycles for the new CRC checksum to be ready is dependent of the configuration of the CRC Beat Size bit group in the CRC Control register(CRCCTRL.CRCBEATSIZE). Each byte needs one clock cycle to be calculated.

19.8.1.4 CRC Checksum

Name: CRCCHKSUM

Offset: 0x08

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	CRCCHKSUM[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCCHKSUM[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCCHKSUM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCCHKSUM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

The CRCCHKSUM represents the 16- or 32-bit checksum value and the generated CRC.

- **Bits 31:0 – CRCCHKSUM[31:0]: CRC Checksum**

  These bits store the generated CRC result. The 16 MSB bits are always read zero when CRC-16 is enabled.

  These bits should only be read when CRC Module Busy bit in the CRC Status register (CRCSTATUS.BUSY) is zero.

  If CRC-16 is selected and CRCSTATUS.BUSY is zero (CRC generation is completed), this bit group will contain a valid checksum.

  If CRC-32 is selected and CRCSTATUS.BUSY is zero (CRC generation is completed), this bit group will contain a valid reversed checksum. Bit 31 is swapped with bit 0, bit 30 with bit 1, etc.

19.8.1.5 CRC Status

Name: CRCSTATUS

Offset: 0x0C

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
							CRCZERO	CRCBUSY
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – CRCZERO: CRC Zero**
  This bit is cleared when a new CRC source is selected.

  This bit is set when the CRC generation is complete and the CRC Checksum is zero.

- **Bit 0 – CRCBUSY: CRC Module Busy**
  When used with an I/O interface (CRCCTRL.CRCSRC is 0x1), this bit is cleared by writing a one to it.

  When used with an I/O interface (CRCCTRL.CRCSRC is 0x1), this bit is set when the CRC Data Input (CRC-DATAIN) register is written.

  When used with a DMA channel (CRCCTRL.CRCSRC is 0x20 to 0x3F), this bit is cleared when the corresponding DMA channel is disabled.

  When used with a DMA channel (CRCCTRL.CRCSRC is 0x20 to 0x3F), this bit is set when the corresponding DMA channel is enabled.

  Writing a zero to this bit has no effect.

  When used with an I/O interface(CRCCTRL.CRCSRC is 0x1), writing a one to this bit will clear the CRC Module Busy bit.

  When used with a DMA channel, writing a one to this bit has no effect.

19.8.1.6 Debug Control

Name: DBGCTRL

Offset: 0x0D

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – DBGRUN: Debug Run**
  This bit is not reset by a software reset.
  This bit controls the functionality when the CPU is halted by an external debugger.
  0: The DMAC is halted when the CPU is halted by an external debugger.
  1: The DMAC continues normal operation when the CPU is halted by an external debugger.

19.8.1.7 QOS Control

Name: QOSCTRL

Offset: 0x0E

Reset: 0x15

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
			DQOS[1:0]		FQOS[1:0]		WRBQOS[1:0]	
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	1	0	1

- **Bits 7:6 – Reserved**
These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:4 – DQOS[1:0]: Data Transfer Quality of Service**
These bits define the memory priority access during the data transfer operation, as shown in Table 19-6 on page 301

Table 19-6. Data Transfer Quality of Service

DQOS[1:0]	Name	Description
0x0	DISABLE	Background (no sensitive operation)
0x1	LOW	Sensitive Bandwidth
0x2	MEDIUM	Sensitive Latency
0x3	HIGH	Critical Latency

- **Bits 3:2 – FQOS[1:0]: Fetch Quality of Service**
These bits define the memory priority access during the fetch operation, as shown in Table 19-7 on page 301

Table 19-7. Fetch Quality of Service

FQOS[1:0]	Name	Description
0x0	DISABLE	Background (no sensitive operation)
0x1	LOW	Sensitive Bandwidth
0x2	MEDIUM	Sensitive Latency
0x3	HIGH	Critical Latency

- **Bits 1:0 – WRBQOS[1:0]: Write-Back Quality of Service**
These bits define the memory priority access during the write-back operation, as shown in Table 19-8 on page 302

Table 19-8. Write-Back Quality of Service

WRBQOS[1:0]	Name	Description
0x0	DISABLE	Background (no sensitive operation)
0x1	LOW	Sensitive Bandwidth
0x2	MEDIUM	Sensitive Latency
0x3	HIGH	Critical Latency

19.8.1.8 Software Trigger Control

Name: SWTRIGCTRL

Offset: 0x10

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			SWTRIG5	SWTRIG4	SWTRIG3	SWTRIG2	SWTRIG1	SWTRIG0
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – SWTRIGx [x=5..0]: Channel x Software Trigger**
  This bit is cleared when the Channel Pending bit in the Channel Status register (CHSTATUS.PEND) for the corresponding channel is set, or by writing a one to it.

  This bit is set if CHSTATUS.PEND is already one, when writing a one to this bit.

  Writing a zero to this bit will clear the bit.

  Writing a one to this bit will generate a DMA software trigger on channel x, if CHSTATUS.PEND is zero for channel x.

19.8.1.9 Priority Control 0

Name: PRICTRL0

Offset: 0x14

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	RRLVLEN3					LVLPRI3[2:0]		
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RRLVLEN2					LVLPRI2[2:0]		
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RRLVLEN1					LVLPRI1[2:0]		
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RRLVLEN0					LVLPRI0[2:0]		
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 31 – RRLVLEN3: Level 3 Round-Robin Scheduling Enable**

  0: Static scheduling scheme for channels with level 3 priority.

  1: Round-robin scheduling scheme for channels with level 3 priority.

  For details on scheduling schemes, refer to "Arbitration" on page 275.

- **Bits 30:27 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 26:24 – LVLPRI3[2:0]: Level 3 Channel Priority Number**

  When round-robin arbitration is enabled (PRICTRL0.RRLVLEN3 is one) for priority level 3, this register holds the channel number of the last DMA channel being granted access as the active channel with priority level 3.

  When static arbitration is enabled (PRICTRL0.RRLVLEN3 is zero) for priority level 3, and the value of this bit group is non-zero, it will affect the static priority scheme. If the value of this bit group is x, channel x will have the highest priority. The priority will decrease as the channel number increases from x to n, where n is the maximum number of channels. Channel n has higher priority than channel 0, and the priority will continue to decrease from channel 0 to channel (x-1).

This bit group is not reset when round-robin scheduling gets disabled (PRICTRL0.RRLVLEN3 written to zero).

- **Bit 23 – RRLVLEN2: Level 2 Round-Robin Scheduling Enable**

  0: Static scheduling scheme for channels with level 2 priority.

  1: Round-robin scheduling scheme for channels with level 2 priority.

  For details on scheduling schemes, refer to "Arbitration" on page 275.

- **Bits 22:19 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 18:16 – LVLPRI2[2:0]: Level 2 Channel Priority Number**

  When round-robin arbitration is enabled (PRICTRL0.RRLVLEN2 is one) for priority level 2, this register holds the channel number of the last DMA channel being granted access as the active channel with priority level 2.

  When static arbitration is enabled (PRICTRL0.RRLVLEN2 is zero) for priority level 2, and the value of this bit group is non-zero, it will affect the static priority scheme. If the value of this bit group is x, channel x will have the highest priority. The priority will decrease as the channel number increases from x to n, where n is the maximum number of channels. Channel n has higher priority than channel 0, and the priority will continue to decrease from channel 0 to channel (x-1).

  This bit group is not reset when round-robin scheduling gets disabled (PRICTRL0.RRLVLEN2 written to zero).

- **Bit 15 – RRLVLEN1: Level 1 Round-Robin Scheduling Enable**

  0: Static scheduling scheme for channels with level 1 priority.

  1: Round-robin scheduling scheme for channels with level 1 priority.

  For details on scheduling schemes, refer to "Arbitration" on page 275.

- **Bits 14:11 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 10:8 – LVLPRI1[2:0]: Level 1 Channel Priority Number**

  When round-robin arbitration is enabled (PRICTRL0.RRLVLEN1 is one) for priority level 1, this register holds the channel number of the last DMA channel being granted access as the active channel with priority level 1.

  When static arbitration is enabled (PRICTRL0.RRLVLEN1 is zero) for priority level 1, and the value of this bit group is non-zero, it will affect the static priority scheme. If the value of this bit group is x, channel x will have the highest priority. The priority will decrease as the channel number increases from x to n, where n is the maximum number of channels. Channel n has higher priority than channel 0, and the priority will continue to decrease from channel 0 to channel (x-1).

  This bit group is not reset when round-robin scheduling gets disabled (PRICTRL0.RRLVLEN1 written to zero).

- **Bit 7 – RRLVLEN0: Level 0 Round-Robin Scheduling Enable**

  0: Static scheduling scheme for channels with level 0 priority.

  1: Round-robin scheduling scheme for channels with level 0 priority.

  For details on scheduling schemes, refer to "Arbitration" on page 275.

- **Bits 6:3 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – LVLPRI0[2:0]: Level 0 Channel Priority Number**

  When round-robin arbitration is enabled (PRICTRL0.RRLVLEN0 is one) for priority level 0, this register holds the channel number of the last DMA channel being granted access as the active channel with priority level 0.

  When static arbitration is enabled (PRICTRL0.RRLVLEN0 is zero) for priority level 0, and the value of this bit group is non-zero, it will affect the static priority scheme. If the value of this bit group is x, channel x will have the highest priority. The priority will decrease as the channel number increases from x to n, where n is the maximum number of channels. Channel n has higher priority than channel 0, and the priority will continue to decrease from channel 0 to channel (x-1).

This bit group is not reset when round-robin scheduling gets disabled (PRICTRL0.RRLVLEN0 written to zero).

19.8.1.10 Interrupt Pending

Name: INTPEND

Offset: 0x20

Reset: 0x0000

Access: Read-Write

Property: -

Bit	15	14	13	12	11	10	9	8
	PEND	BUSY	FERR			SUSP	TCMPL	TERR
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						ID[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to identify the lowest DMA channel with pending interrupt.

- **Bit 15 – PEND: Pending**
  This bit is read one when the channel selected by Channel ID field (ID) is pending.
- **Bit 14 – BUSY: Busy**
  This bit is read one when the channel selected by Channel ID field (ID) is busy.
- **Bit 13 – FERR: Fetch Error**
  This bit is read one when the channel selected by Channel ID field (ID) fetched an invalid descriptor.
- **Bits 12:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 10 – SUSP: Channel Suspend**
  This bit is read one when the channel selected by Channel ID field (ID) has pending Suspend interrupt.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Channel ID (ID) Suspend interrupt flag.
- **Bit 9 – TCMPL: Transfer Complete**
  This bit is read one when the channel selected by Channel ID field (ID) has pending Transfer Complete interrupt.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Channel ID (ID) Transfer Complete interrupt flag.
- **Bit 8 – TERR: Transfer Error**
  This bit is read one when the channel selected by Channel ID field (ID) has pending Transfer Error interrupt.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Channel ID (ID) Transfer Error interrupt flag.
- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – ID[2:0]: Channel ID**
  These bits store the lowest channel number with pending interrupts. The number is valid if Suspend (SUSP), Transfer Complete (TCMPL) or Transfer Error (TERR) bits are set. The Channel ID field is refreshed when a new channel (with channel number less than the current one) with pending interrupts is detected, or when the application clears the corresponding channel interrupt sources. When no pending channels interrupts are available, these bits will always return zero value when read.

  When the bits are written, indirect access to the corresponding Channel Interrupt Flag register is enabled.

19.8.1.11 Interrupt Status

Name: INTSTATUS

Offset: 0x24

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			CHINT5	CHINT4	CHINT3	CHINT2	CHINT1	CHINT0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – CHINTx [x=5..0]: Channel x Pending Interrupt**
  This bit is set when Channel x has pending interrupt.

  This bit is cleared when the corresponding Channel x interrupts are disabled or the interrupts sources are cleared.

19.8.1.12 Busy Channels

Name: BUSYCH

Offset: 0x28

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			BUSYCH5	BUSYCH4	BUSYCH3	BUSYCH2	BUSYCH1	BUSYCH0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – BUSYCHx [x=5..0]: Busy Channel x**
  This bit is cleared when the channel trigger action for DMA channel x is complete, when a bus error for DMA channel x is detected, or when DMA channel x is disabled.

  This bit is set when DMA channel x starts a DMA transfer.

19.8.1.13 Pending Channels

Name: PENDCH
Offset: 0x2C
Reset: 0x00000000
Access: Read-Only
Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			PENDCH5	PENDCH4	PENDCH3	PENDCH2	PENDCH1	PENDCH0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – PENDCHx [x=5..0]: Pending Channel x**
  This bit is cleared when trigger execution defined by channel trigger action settings for DMA channel x is started, when a bus error for DMA channel x is detected or when DMA channel x is disabled. For details on trigger action settings, refer to Table 19-10.

  This bit is set when a transfer is pending on DMA channel x.

19.8.1.14 Active Channel and Levels

Name: ACTIVE

Offset: 0x30

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
	BTCNT[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BTCNT[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ABUSY			ID[4:0]				
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					LVLEX3	LVLEX2	LVLEX1	LVLEX0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:16 – BTCNT[15:0]: Active Channel Block Transfer Count**
  These bits hold the 16-bit block transfer count of the ongoing transfer. This value is stored in the active channel and written back in the corresponding Write-Back channel memory location when the arbiter grants a new channel access. The value is valid only when the active channel active busy flag (ABUSY) is set.

- **Bit 15 – ABUSY: Active Channel Busy**
  This bit is cleared when the active transfer count is written back in the Write-Back memory section.

  This flag is set when the next descriptor transfer count is read from the Write-Back memory section.

- **Bits 14:13 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 12:8 – ID[4:0]: Active Channel ID**
  These bits hold the channel index currently stored in the active channel registers. The value is updated each time the arbiter grants a new channel transfer access request.

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 3:0 – LVLEXx [x=3..0]: Level x Channel Trigger Request Executing**
  This bit is set when a level-x channel trigger request is executing or pending.

19.8.1.15 Descriptor Memory Section Base Address

Name: BASEADDR

Offset: 0x34

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	BASEADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BASEADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BASEADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BASEADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – BASEADDR[31:0]: Descriptor Memory Base Address**
  These bits store the Descriptor memory section base address. The value must be 128-bit aligned.

19.8.1.16 Write-Back Memory Section Base Address

Name: WRBADDR

Offset: 0x38

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	WRBADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WRBADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	WRBADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WRBADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – WRBADDR[31:0]: Write-Back Memory Base Address**
  These bits store the Write-Back memory base address. The value must be 128-bit aligned.

19.8.1.17 Channel ID

Name: CHID

Offset: 0x3F

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
						ID[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – ID[2:0]: Channel ID**
  These bits define the channel number that will be accessed. Before reading or writing a channel register, the channel ID bit group must be written first.

19.8.1.18 Channel Control A

Name: CHCTRLA

Offset: 0x40

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – ENABLE: Channel Enable**
  0: DMA channel is disabled.
  1: DMA channel is enabled.
  Writing a zero to this bit during an ongoing transfer, the bit will not be cleared until the internal data transfer buffer is empty and the DMA transfer is aborted. The internal data transfer buffer will be empty once the ongoing burst transfer is completed.
  Writing a one to this bit will enable the DMA channel.
  This bit is not enable-protected.

- **Bit 0 – SWRST: Channel Software Reset**
  0: There is no reset operation ongoing.
  1: The reset operation is ongoing.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets the channel registers to their initial state. The bit can be set when the channel is disabled (ENABLE = 0). Writing a one to this bit will be ignored as long as the channel is enabled (ENABLE = 1). This bit is automatically cleared when the reset is completed.
  Writing a one to this bit when the corresponding DMA channel is disabled (ENABLE is zero), resets all registers for the corresponding DMA channel to their initial state If the corresponding DMA channel is enabled, the reset request will be ignored.

19.8.1.19 Channel Control B

Name: CHCTRLB
Offset: 0x44
Reset: 0x00000000
Access: Read-Write
Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
							CMD[1:0]	
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRIGACT[1:0]							
Access	R/W	R/W	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				TRIGSRC[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		LVL[1:0]		EVOE	EVIE	EVACT[2:0]		
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:26 – Reserved**
These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 25:24 – CMD[1:0]: Software Command**
These bits define the software commands, as shown in Table 19-9.
These bits are not enable-protected.

Table 19-9. Software Command

CMD[1:0]	Name	Description
0x0	NOACT	No action
0x1	SUSPEND	Channel suspend operation
0x2	RESUME	Channel resume operation
0x3		Reserved

- **Bits 23:22 – TRIGACT[1:0]: Trigger Action**
  These bits define the trigger action used for a transfer, as shown in Table 19-10.

Table 19-10. Trigger Action

TRIGACT[1:0]	Name	Description
0x0	BLOCK	One trigger required for each block transfer
0x1		Reserved
0x2	BEAT	One trigger required for each beat transfer
0x3	TRANSACTION	One trigger required for each transaction

- **Bits 21:13 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 12:8 – TRIGSRC[4:0]: Peripheral Trigger Source**
  These bits define the peripheral trigger which is source of the transfer. For details on trigger selection and trigger modes, refer to "Transfer Triggers and Actions" on page 277 and Table 19-10.

Table 19-11. Peripheral trigger source

Value	Name	Description
0x00	DISABLE	Only software/event triggers
0x01	SERCOM0 RX	SERCOM0 RX Trigger
0x02	SERCOM0 TX	SERCOM0 TX Trigger
0x03	SERCOM1 RX	SERCOM1 RX Trigger
0x04	SERCOM1 TX	SERCOM1 TX Trigger
0x05-0x0B	-	Reserved
0x0C	TC1 OVF	TC1 Overflow Trigger
0x0D	TC1 MC0	TC1 Match/Compare 0 Trigger
0x0E	TC1 MC1	TC1 Match/Compare 1 Trigger
0x0F	TC2 OVF	TC2 Overflow Trigger
0x10	TC2 MC0	TC2 Match/Compare 0 Trigger
0x11	TC2 MC1	TC2 Match/Compare 1 Trigger
0x12	ADC RESRDY	ADC Result Ready Trigger

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 6:5 – LVL[1:0]: Channel Arbitration Level**
  These bits define the arbitration level used for the DMA channel. The available levels are shown in Table 19-12, where a high level has priority over a low level. For further details on arbitration schemes, refer to "Arbitration" on page 275.

These bits are not enable-protected.

Table 19-12. Channel Arbitration Level

LVL[1:0]	Name	Description
0x0	LVL0	Channel Priority Level 0
0x1	LVL1	Channel Priority Level 1
0x2	LVL2	Channel Priority Level 2
0x3	LVL3	Channel Priority Level 3
0x4-0x7		Reserved

- **Bit 4 – EVOE: Channel Event Output Enable**
  This bit indicates if the Channel event generation is enabled. The event will be generated for every condition defined in the descriptor Event Output Selection (BTCTRL.EVOSEL).
  0: Channel event generation is disabled.
  1: Channel event generation is enabled.
  This bit is available only on channels with event output support. Refer to Table 23-6 and Table 23-4 for details.
- **Bit 3 – EVIE: Channel Event Input Enable**
  0: Channel event action will not be executed on any incoming event.
  1: Channel event action will be executed on any incoming event.
  This bit is available only on channels with event input support. Refer to Table 23-6 and Table 23-4 for details.
- **Bits 2:0 – EVACT[2:0]: Event Input Action**
  These bits define the event input action, as shown in Table 19-13. The action is executed only if the corresponding EVIE bit in CHCTRLB register of the channel is set. For details on event actions, refer to "Event Input Actions" on page 282.
  These bits are available only for channels with event input support. Refer to Table 23-6 and Table 23-4 for details.

Table 19-13. Event Input Action

EVACT[2:0]	Name	Description
0x0	NOACT	No action
0x1	TRIG	Transfer and periodic transfer trigger
0x2	CTRIG	Conditional transfer trigger
0x3	CBLOCK	Conditional block transfer
0x4	SUSPEND	Channel suspend operation
0x5	RESUME	Channel resume operation
0x6	SSKIP	Skip next block suspend action
0x7		Reserved

19.8.1.20 Channel Interrupt Enable Clear

Name: CHINTENCLR

Offset: 0x4C

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
						SUSP	TCMPL	TERR
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Channel Interrupt Enable Set (CHINTENSET) register.

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – SUSP: Channel Suspend Interrupt Enable**
  0: The Channel Suspend interrupt is disabled.

  1: The Channel Suspend interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Channel Suspend Interrupt Enable bit, which disables the Channel Suspend interrupt.

- **Bit 1 – TCMPL: Transfer Complete Interrupt Enable**
  0: The Channel Transfer Complete interrupt is disabled. When block action is set to none, the TCMPL flag will not be set when a block transfer is completed.

  1: The Channel Transfer Complete interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Channel Transfer Complete Interrupt Enable bit, which disables the Channel Transfer Complete interrupt.

- **Bit 0 – TERR: Transfer Error Interrupt Enable**
  0: The Channel Transfer Error interrupt is disabled.

  1: The Channel Transfer Error interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Channel Transfer Error Interrupt Enable bit, which disables the Channel Transfer Error interrupt.

19.8.1.21 Channel Interrupt Enable Set

Name: CHINTENSET

Offset: 0x4D

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
						SUSP	TCMPL	TERR
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Channel Interrupt Enable Clear (CHINTENCLR) register.

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – SUSP: Channel Suspend Interrupt Enable**
  0: The Channel Suspend interrupt is disabled.
  1: The Channel Suspend interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Channel Suspend Interrupt Enable bit, which enables the Channel Suspend interrupt.

- **Bit 1 – TCMPL: Transfer Complete Interrupt Enable**
  0: The Channel Transfer Complete interrupt is disabled.
  1: The Channel Transfer Complete interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Channel Transfer Complete Interrupt Enable bit, which enables the Channel Transfer Complete interrupt.

- **Bit 0 – TERR: Transfer Error Interrupt Enable**
  0: The Channel Transfer Error interrupt is disabled.
  1: The Channel Transfer Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Channel Transfer Error Interrupt Enable bit, which enables the Channel Transfer Error interrupt.

19.8.1.22 Channel Interrupt Flag Status and Clear

Name: CHINTFLAG

Offset: 0x4E

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
						SUSP	TCMPL	TERR
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – SUSP: Channel Suspend**
  This flag is cleared by writing a one to it.
  This bit is set when a block transfer with suspend block action is completed, when a software suspend command is executed, when a suspend event is received or when an invalid descriptor is fetched by the DMA.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Channel Suspend interrupt flag for the corresponding channel.
  For details on available software commands, refer to Table 19-9.
  For details on available event input actions, refer to Table 19-13.
  For details on available block actions, refer to Table 19-17.

- **Bit 1 – TCMPL: Transfer Complete**
  This flag is cleared by writing a one to it.
  This flag is set when a block transfer is completed and the corresponding interrupt block action is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Transfer Complete interrupt flag for the corresponding channel.

- **Bit 0 – TERR: Transfer Error**
  This flag is cleared by writing a one to it.
  This flag is set when a bus error is detected during a beat transfer or when the DMAC fetches an invalid descriptor.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Transfer Error interrupt flag for the corresponding channel.

19.8.1.23 Channel Status

Name: CHSTATUS

Offset: 0x4F

Reset: 0x00

Access: Read-Only

Property: -

Bit	7	6	5	4	3	2	1	0
						FERR	BUSY	PEND
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – FERR: Fetch Error**
  This bit is cleared when the software resume command is executed.
  This bit is set when an invalid descriptor is fetched.

- **Bit 1 – BUSY: Channel Busy**
  This bit is cleared when the channel trigger action is complete, when a bus error is detected or when the channel is disabled.
  This bit is set when the DMA channel starts a DMA transfer.

- **Bit 0 – PEND: Channel Pending**
  This bit is cleared when trigger execution defined by channel trigger action settings is started, when a bus error is detected or when the channel is disabled. For details on trigger action settings, refer to Table 19-10.
  This bit is set when a transfer is pending on the DMA channel.

19.8.2 DMAC SRAM Registers

19.8.2.1 Block Transfer Control

Name: BTCTRL

Offset: 0x00

Bit	15	14	13	12	11	10	9	8
	STEPSIZE[2:0]			STEPSEL	DSTINC	SRCINC	BEATSIZE[1:0]	

Bit	7	6	5	4	3	2	1	0
				BLOCKACT[1:0]		EVOSEL[1:0]		VALID

The BTCTRL register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

- **Bits 15:13 – STEPSIZE[2:0]: Address Increment Step Size**
  These bits select the address increment step size, as shown in Table 19-14. The setting apply to source or destination address, depending on STEPSEL setting.

Table 19-14. Address Increment Step Size

STEPSIZE[2:0]	Name	Description
0x0	X1	Next ADDR <- ADDR + BEATSIZE * 1
0x1	X2	Next ADDR <- ADDR + BEATSIZE * 2
0x2	X4	Next ADDR <- ADDR + BEATSIZE * 4
0x3	X8	Next ADDR <- ADDR + BEATSIZE * 8
0x4	X16	Next ADDR <- ADDR + BEATSIZE * 16
0x5	X32	Next ADDR <- ADDR + BEATSIZE * 32
0x6	X64	Next ADDR <- ADDR + BEATSIZE * 64
0x7	X128	Next ADDR <- ADDR + BEATSIZE * 128

- **Bit 12 – STEPSEL: Step Selection**
  This bit selects if source or destination addresses are using the step size settings, according to Table 19-15.

Table 19-15. Step Selection

STEPSEL	Name	Description
0x0	DST	Step size settings apply to the destination address
0x1	SRC	Step size settings apply to the source address

- **Bit 11 – DSTINC: Destination Address Increment Enable**
  0: The Destination Address Increment is disabled.
  1: The Destination Address Increment is enabled.
  Writing a zero to this bit will disable the destination address incrementation. The address will be kept fixed during the data transfer.

Writing a one to this bit will enable the destination address incrementation. By default, the destination address is incremented by 1. If the STEPSEL bit is cleared, flexible step-size settings are available in the STEPSIZE register, as shown in Table 19-14.

- **Bit 10 – SRCINC: Source Address Increment Enable**
  0: The Source Address Increment is disabled.
  1: The Source Address Increment is enabled.
  Writing a zero to this bit will disable the source address incrementation. The address will be kept fixed during the data transfer.
  Writing a one to this bit will enable the source address incrementation. By default, the source address is incremented by 1. If the STEPSEL bit is set, flexible step-size settings are available in the STEPSIZE register, as shown in Table 19-14.

- **Bits 9:8 – BEATSIZE[1:0]: Beat Size**
  These bits define the size of one beat, as shown in Table 19-16. A beat is the size of one data transfer bus access, and the setting apply to both read and write accesses.

Table 19-16. Beat Size

BEATSIZE[1:0]	Name	Description
0x0	BYTE	8-bit access
0x1	HWORD	16-bit access
0x2	WORD	32-bit access
0x3		Reserved

- **Bits 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:3 – BLOCKACT[1:0]: Block Action**
  These bits define what actions the DMAC should take after a block transfer has completed. The available actions are listed in Table 19-17.

Table 19-17. Block Action

BLOCKACT[1:0]	Name	Description
0x0	NOACT	No action
0x1	INT	Channel in normal operation and block interrupt
0x2	SUSPEND	Channel suspend operation is completed
0x3	BOTH	Both channel suspend operation and block interrupt

- **Bits 2:1 – EVOSEL[1:0]: Event Output Selection**
  These bits define the event output selection, as shown in Table 19-18.

Table 19-18. Event Output Selection

EVOSEL[1:0]	Name	Description
0x0	DISABLE	Event generation disabled
0x1	BLOCK	Event strobe when block transfer complete
0x2		Reserved
0x3	BEAT	Event strobe when beat transfer complete

- **Bit 0 – VALID: Descriptor Valid**

  0: The descriptor is not valid.

  1: The descriptor is valid.

  Writing a zero to this bit in the Descriptor or Write-Back memory will suspend the DMA channel operation when fetching the corresponding descriptor.

  The bit is automatically cleared in the Write-Back memory section when channel is aborted, when an error is detected during the block transfer, or when the block transfer is completed.

19.8.2.2 Block Transfer Count

Name: BTCNT

Offset: 0x02

Bit	15	14	13	12	11	10	9	8
	BTCNT[15:8]							

Bit	7	6	5	4	3	2	1	0
	BTCNT[7:0]							

The BTCNT register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

- **Bits 15:0 – BTCNT[15:0]: Block Transfer Count**
  This bit group holds the 16-bit block transfer count.

  During a transfer, the internal counter value is decremented by one after each beat transfer. The internal counter is written to the corresponding write-back memory section for the DMA channel when the DMA channel loses priority, is suspended or gets disabled. The DMA channel can be disabled by a complete transfer, a transfer error or by software.

19.8.2.3 Transfer Source Address

Name: SRCADDR

Offset: 0x04

Bit	31	30	29	28	27	26	25	24
	SRCADDR[31:24]							

Bit	23	22	21	20	19	18	17	16
	SRCADDR[23:16]							

Bit	15	14	13	12	11	10	9	8
	SRCADDR[15:8]							

Bit	7	6	5	4	3	2	1	0
	SRCADDR[7:0]							

The SRCADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

- **Bits 31:0 – SRCADDR[31:0]: Transfer Source Address**
  This bit group holds the source address corresponding to the last beat transfer address in the block transfer.

19.8.2.4 Transfer Destination Address

Name: DSTADDR

Offset: 0x08

Bit	31	30	29	28	27	26	25	24
	DSTADDR[31:24]							

Bit	23	22	21	20	19	18	17	16
	DSTADDR[23:16]							

Bit	15	14	13	12	11	10	9	8
	DSTADDR[15:8]							

Bit	7	6	5	4	3	2	1	0
	DSTADDR[7:0]							

The DSTADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

- **Bits 31:0 – DSTADDR[31:0]: Transfer Destination Address**
  This bit group holds the destination address corresponding to the last beat transfer address in the block transfer.

19.8.2.5 Next Descriptor Address

Name: DESCADDR

Offset: 0x0C

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

The DESCADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

- **Bits 31:0 – DESCADDR[31:0]: Next Descriptor Address**
  This bit group holds the SRAM address of the next descriptor. The value must be 128-bit aligned. If the value of this SRAM register is 0x00000000, the transaction will be terminated when the DMAC tries to load the next transfer descriptor.

20. EIC – External Interrupt Controller

20.1 Overview

The External Interrupt Controller (EIC) allows external pins to be configured as interrupt lines. Each interrupt line can be individually masked and can generate an interrupt on rising, falling or both edges, or on high or low levels. Each external pin has a configurable filter to remove spikes. Each external pin can also be configured to be asynchronous in order to wake up the device from sleep modes where all clocks have been disabled. External pins can also generate an event.

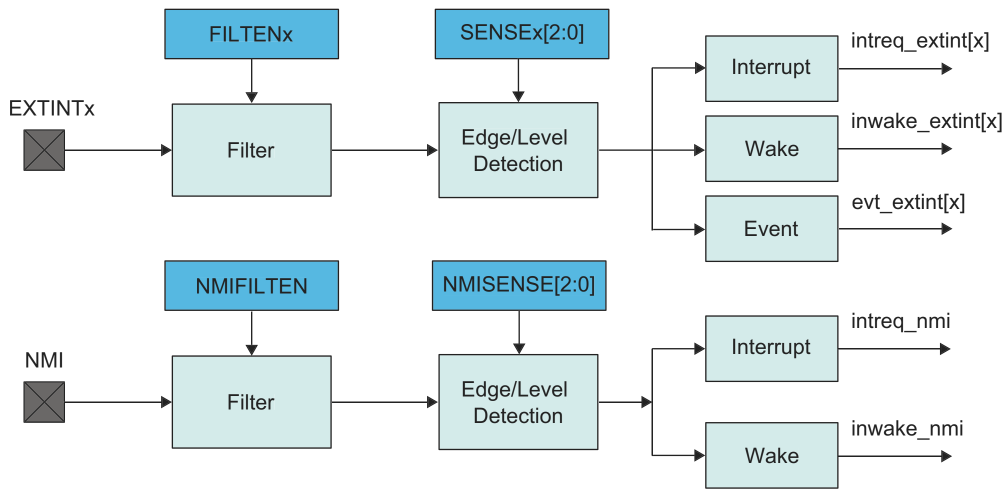
A separate non-maskable interrupt (NMI) is also supported. It has properties similar to the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other interrupt mode.

20.2 Features

- 8 external pins, plus one non-maskable pin
- Dedicated interrupt line for each pin
- Individually maskable interrupt lines
- Interrupt on rising, falling or both edges
- Interrupt on high or low levels
- Asynchronous interrupts for sleep modes without clock
- Filtering of external pins
- Event generation
- Configurable wake-up for sleep modes

20.3 Block Diagram

Figure 20-1. EIC Block Diagram



20.4 Signal Description

Signal Name	Type	Description
EXTINT[7..0]	Digital Input	External interrupt pin
NMI	Digital Input	Non-maskable interrupt pin

Refer to "I/O Multiplexing and Considerations" on page 10 for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

20.5 Product Dependencies

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

20.5.1 I/O Lines

Using the EIC's I/O lines requires the I/O pins to be configured. Refer to "PORT" on page 375 for details.

20.5.2 Power Management

All interrupts are available in all sleep modes, but the EIC can be configured to automatically mask some interrupts in order to prevent device wake-up.

The EIC will continue to operate in any sleep mode where the selected source clock is running. The EIC's interrupts can be used to wake up the device from sleep modes. Events connected to the Event System can trigger other operations in the system without exiting sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

20.5.3 Clocks

The EIC bus clock (CLK_EIC_APB) can be enabled and disabled in the Power Manager, and the default state of CLK_EIC_APB can be found in the Peripheral Clock Masking section in "PM – Power Manager" on page 107.

A generic clock (GCLK_EIC) is required to clock the peripheral. This clock must be configured and enabled in the Generic Clock Controller before using the peripheral. Refer to "GCLK – Generic Clock Controller" on page 85 for details.

This generic clock is asynchronous to the user interface clock (CLK_EIC_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 338 for further details.

20.5.4 Interrupts

There are two interrupt request lines, one for the external interrupts (EXTINT) and one for non-maskable interrupt (NMI).

The EXTINT interrupt request line is connected to the interrupt controller. Using the EIC interrupt requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

The NMI interrupt request line is also connected to the interrupt controller, but does not require the interrupt to be configured.

20.5.5 Events

The events are connected to the Event System. Using the events requires the Event System to be configured first. The External Interrupt Controller generates events as pulses.

Refer to "EVSYS – Event System" on page 402 for details.

20.5.6 Debug Operation

When the CPU is halted in 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.

20.5.7 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Interrupt Flag Status and Clear register (INTFLAG - refer to INTFLAG)
- Non-Maskable Interrupt Flag Status and Clear register (NMIFLAG - refer to NMIFLAG)

Write-protection is denoted by the Write-Protected property in the register description.

Write-protection does not apply to accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

20.5.8 Analog Connections

Not applicable.

20.6 Functional Description

20.6.1 Principle of Operation

The EIC detects edge or level condition to generate interrupts to the CPU Interrupt Controller or events to the Event System. Each external interrupt pin (EXTINT) can be filtered using majority vote filtering, clocked by generic clock GCLK_EIC.

20.6.2 Basic Operation

20.6.2.1 Initialization

The EIC must be initialized in the following order:

1. Enable CLK_EIC_APB
2. If edge detection or filtering is required, GCLK_EIC must be enabled
3. Write the EIC configuration registers (EVCTRL, WAKEUP, CONFIGy)
4. Enable the EIC

When NMI is used, GCLK_EIC must be enabled after EIC configuration (NMICTRL).

20.6.2.2 Enabling, Disabling and Resetting

The EIC is enabled by writing a one to the Enable bit in the Control register (CTRL.ENABLE). The EIC is disabled by writing a zero to CTRL.ENABLE.

The EIC is reset by writing a one to the Software Reset bit in the Control register (CTRL.SWRST). All registers in the EIC will be reset to their initial state, and the EIC will be disabled.

Refer to CTRL register for details.

20.6.3 External Pin Processing

Each external pin can be configured to generate an interrupt/event on edge detection (rising, falling or both edges) or level detection (high or low). The sense of external pins is configured by writing the Interrupt Sense x bits in the Config y register (CONFIGy.SENSEx). The corresponding interrupt flag (INTFLAG.EXTINT[x]) in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition is met (CONFIGy.SENSEx must be different from zero).

When the interrupt has been cleared in edge-sensitive mode, INTFLAG.EXTINT[x] will only be set if a new interrupt condition is met. In level-sensitive mode, when interrupt has been cleared, INTFLAG.EXTINT[x] will be set immediately if the EXTINTx pin still matches the interrupt condition.

Each external pin can be filtered by a majority vote filtering, clocked by GCLK_EIC. Filtering is enabled if bit Filter Enable x in the Configuration y register (CONFIGy.FILTENx) is written to one. The majority vote filter samples the external pin three times with GCLK_EIC and outputs the value when two or more samples are equal.

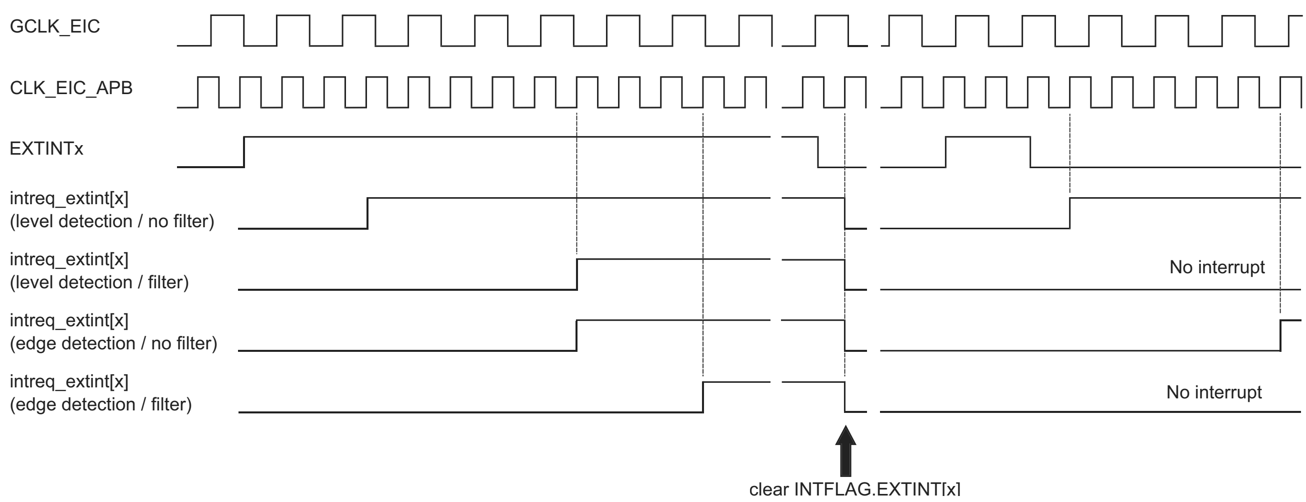
Table 20-1. Majority Vote Filter

Samples [0, 1, 2]	Filter Output
[0,0,0]	0
[0,0,1]	0
[0,1,0]	0
[0,1,1]	1
[1,0,0]	0
[1,0,1]	1
[1,1,0]	1
[1,1,1]	1

When an external interrupt is configured for level detection, or if filtering is disabled, detection is made asynchronously, and GCLK_EIC is not required.

If filtering or edge detection is enabled, the EIC automatically requests the GCLK_EIC to operate (GCLK_EIC must be enabled in the GCLK module, see "GCLK – Generic Clock Controller" on page 85 for details). If level detection is enabled, GCLK_EIC is not required, but interrupt and events can still be generated.

Figure 20-2. Interrupt detections



The detection delay depends on the detection mode.

Table 20-2. Interrupt Latency

Detection Mode	Latency (Worst Case)
Level without filter	3 CLK_EIC_APB periods
Level with filter	4 GCLK_EIC periods + 3 CLK_EIC_APB periods
Edge without filter	4 GCLK_EIC periods + 3 CLK_EIC_APB periods
Edge with filter	6 GCLK_EIC periods + 3 CLK_EIC_APB periods

20.6.4 Additional Features

The non-maskable interrupt pin can also generate an interrupt on edge or level detection, but it is configured with the dedicated NMI Control register (NMICTRL - refer to NMICTRL). To select the sense for NMI, write to the NMISENSE bit group in the NMI Control register (NMICTRL.NMISENSE). NMI filtering is enabled by writing a one to the NMI Filter Enable bit (NMICTRL.NMIFILTEN).

NMI detection is enabled only by the NMICTRL.NMISENSE value, and the EIC is not required to be enabled.

After reset, NMI is configured to no detection mode.

When an NMI is detected, the non-maskable interrupt flag in the NMI Flag Status and Clear register is set (NMIFLAG.NMI). NMI interrupt generation is always enabled, and NMIFLAG.NMI generates an interrupt request when set.

20.6.5 Interrupts

The EIC has the following interrupt sources:

- External interrupt pins (EXTINTx). See “Basic Operation” on page 335
- Non-maskable interrupt pin (NMI). See “Additional Features” on page 337

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when an interrupt condition occurs (NMIFLAG for NMI). Each interrupt, except NMI, can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the EIC is reset. See the INTFLAG register for details on how to clear interrupt flags. The EIC has one common interrupt request line for all the interrupt sources (except the NMI interrupt request line). Refer to “Processor And Architecture” on page 22 for details. The user must read the INTFLAG (or NMIFLAG) register to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to “Processor And Architecture” on page 22 for details.

20.6.6 Events

The EIC can generate the following output events:

- External event from pin (EXTINTx).

Writing a one to an Event Output Control register (EVCTRLEXTINTEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. Refer to “EVSYS – Event System” on page 402 for details on configuring the Event System.

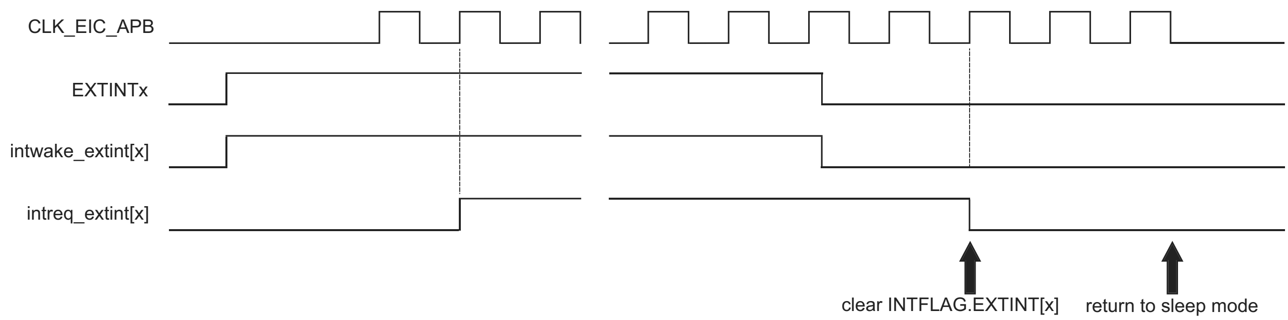
When the condition on pin EXTINTx matches the configuration in the CONFIGy register, the corresponding event is generated, if enabled.

20.6.7 Sleep Mode Operation

In sleep modes, an EXTINTx pin can wake up the device if the corresponding condition matches the configuration in CONFIGy register. Writing a one to a Wake-Up Enable bit (WAKEUP.WAKEUPEN[x]) enables the wake-up from pin EXTINTx. Writing a zero to a Wake-Up Enable bit (WAKEUP.WAKEUPEN[x]) disables the wake-up from pin EXTINTx.

Using WAKEUPEN[x]=1 with INTENSET=0 is not recommended.

Figure 20-3. Wake-Up Operation Example (High-Level Detection, No Filter, WAKEUPEN[x]=1)



20.6.8 Synchronization

Due to the asynchronicity between CLK_EIC_APB and GCLK_EIC, some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set immediately, and cleared when synchronization is complete.

If an operation that requires synchronization is executed while STATUS.SYNCBUSY is one, the bus will be stalled. All operations will complete successfully, but the CPU will be stalled, and interrupts will be pending as long as the bus is stalled.

The following bits need synchronization when written:

- Software Reset bit in the Control register (CTRL.SWRST)
- Enable bit in the Control register (CTRL.ENABLE)

No register needs synchronization when written.

No register needs synchronization when read.

20.7 Register Summary

Table 20-3. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0							ENABLE	SWRST
0x01	STATUS	7:0	SYNCBUSY							
0x02	NMICTRL	7:0					NMIFILTEN	NMISENSE[2:0]		
0x03	NMIFLAG	7:0								NMI
0x04	EVCTRL	7:0	EXTINTEO7	EXTINTEO6	EXTINTEO5	EXTINTEO4	EXTINTEO3	EXTINTEO2	EXTINTEO1	EXTINTEO0
0x05		15:8								
0x06		23:16								
0x07		31:24								
0x08	INTENCLR	7:0	EXTINT7	EXTINT6	EXTINT5	EXTINT4	EXTINT3	EXTINT2	EXTINT1	EXTINT0
0x09		15:8								
0x0A		23:16								
0x0B		31:24								
0x0C	INTENSET	7:0	EXTINT7	EXTINT6	EXTINT5	EXTINT4	EXTINT3	EXTINT2	EXTINT1	EXTINT0
0x0D		15:8								
0x0E		23:16								
0x0F		31:24								
0x10	INTFLAG	7:0	EXTINT7	EXTINT6	EXTINT5	EXTINT4	EXTINT3	EXTINT2	EXTINT1	EXTINT0
0x11		15:8								
0x12		23:16								
0x13		31:24								
0x14	WAKEUP	7:0	WAKEUPEN7	WAKEUPEN6	WAKEUPEN5	WAKEUPEN4	WAKEUPEN3	WAKEUPEN2	WAKEUPEN1	WAKEUPEN0
0x15		15:8								
0x16		23:16								
0x17		31:24								
0x18	CONFIG0	7:0	FILTEN1	SENSE1[2:0]			FILTEN0	SENSE0[2:0]		
0x19		15:8	FILTEN3	SENSE3[2:0]			FILTEN2	SENSE2[2:0]		
0x1A		23:16	FILTEN5	SENSE5[2:0]			FILTEN4	SENSE4[2:0]		
0x1B		31:24	FILTEN7	SENSE7[2:0]			FILTEN6	SENSE6[2:0]		

20.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-protected property in each individual register description. Refer to "Register Access Protection" on page 335 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Synchronized property in each individual register description. Refer to "Synchronization" on page 338 for details.

Some registers are enable-protected, meaning they can be written only when the EIC is disabled. Enable-protection is denoted by the Enabled-Protected property in each individual register description.

20.8.1 Control

Name: CTRL

Offset: 0x00

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – ENABLE: Enable**
  0: The EIC is disabled.
  1: The EIC is enabled.
  Due to synchronization, there is delay from writing CTRL.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately, and the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set. STATUS.SYNCBUSY will be cleared when the operation is complete.

- **Bit 0 – SWRST: Software Reset**
  0: There is no ongoing reset operation.
  1: The reset operation is ongoing.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all registers in the EIC to their initial state, and the EIC will be disabled.
  Writing a one to CTRL.SWRST will always take precedence, meaning that all other writes in the same write operation will be discarded.
  Due to synchronization, there is a delay from writing CTRL.SWRST until the reset is complete. CTRL.SWRST and STATUS.SYNCBUSY will both be cleared when the reset is complete.

20.8.2 Status

Name: STATUS

Offset: 0x01

Reset: 0x00

Access: Read-Only

Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCBUSY							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – SYNCBUSY: Synchronization Busy**
  This bit is cleared when the synchronization of registers between the clock domains is complete.
  This bit is set when the synchronization of registers between clock domains is started.

- **Bits 6:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

20.8.3 Non-Maskable Interrupt Control

Name: NMICTRL

Offset: 0x02

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
					NMIFILTEN	NMISENSE[2:0]		
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – NMIFILTEN: Non-Maskable Interrupt Filter Enable**
  0: NMI filter is disabled.
  1: NMI filter is enabled.

- **Bits 2:0 – NMISENSE[2:0]: Non-Maskable Interrupt Sense**
  These bits define on which edge or level the NMI triggers.

Table 20-4. Non-Maskable Interrupt Sense

NMISENSE[2:0]	Name	Description
0x0	NONE	No detection
0x1	RISE	Rising-edge detection
0x2	FALL	Falling-edge detection
0x3	BOTH	Both-edges detection
0x4	HIGH	High-level detection
0x5	LOW	Low-level detection
0x6-0x7		Reserved

20.8.4 Non-Maskable Interrupt Flag Status and Clear

Name: NMIFLAG

Offset: 0x03

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
								NMI
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – NMI: Non-Maskable Interrupt**
  This flag is cleared by writing a one to it.

  This flag is set when the NMI pin matches the NMI sense configuration, and will generate an interrupt request.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the non-maskable interrupt flag.

20.8.5 Event Control

Name: EVCTRL

Offset: 0x04

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTINTEO7	EXTINTEO6	EXTINTEO5	EXTINTEO4	EXTINTEO3	EXTINTEO2	EXTINTEO1	EXTINTEO0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – EXTINTEOx [x=7..0]: External Interrupt x Event Output Enable**
  These bits indicate whether the event associated with the EXTINTx pin is enabled or not to generated for every detection.

  0: Event from pin EXTINTx is disabled.

  1: Event from pin EXTINTx is enabled.

20.8.6 Interrupt Enable Clear

Name: INTENCLR

Offset: 0x08

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTINT7	EXTINT6	EXTINT5	EXTINT4	EXTINT3	EXTINT2	EXTINT1	EXTINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – EXTINTx [x=7..0]: External Interrupt x Enable**
  0: The external interrupt x is disabled.
  1: The external interrupt x is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the External Interrupt x Enable bit, which enables the external interrupt.

20.8.7 Interrupt Enable Set

Name: INTENSET

Offset: 0x0C

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTINT7	EXTINT6	EXTINT5	EXTINT4	EXTINT3	EXTINT2	EXTINT1	EXTINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – EXTINTx [x=7..0]: External Interrupt x Enable**
  0: The external interrupt x is disabled.
  1: The external interrupt x is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the External Interrupt x Enable bit, which enables the external interrupt.

20.8.8 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x10

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTINT7	EXTINT6	EXTINT5	EXTINT4	EXTINT3	EXTINT2	EXTINT1	EXTINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – EXTINTx [x=7..0]: External Interrupt x**
  This flag is cleared by writing a one to it.

  This flag is set when EXTINTx pin matches the external interrupt sense configuration and will generate an interrupt request if INTENCLR/SET.EXTINT[x] is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the External Interrupt x flag.

20.8.9 Wake-Up Enable

Name: WAKEUP

Offset: 0x14

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WAKEUPEN7	WAKEUPEN6	WAKEUPEN5	WAKEUPEN4	WAKEUPEN3	WAKEUPEN2	WAKEUPEN1	WAKEUPEN0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – WAKEUPENx [x=7..0]: External Interrupt x Wake-up Enable**
  This bit enables or disables wake-up from sleep modes when the EXTINTx pin matches the external interrupt sense configuration.

  0: Wake-up from the EXTINTx pin is disabled.

  1: Wake-up from the EXTINTx pin is enabled.

20.8.10 Configuration n

Name: CONFIG
Offset: 0x18
Reset: 0x00000000
Access: Read-Write
Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	FILTEN7	SENSE7[2:0]			FILTEN6	SENSE6[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FILTEN5	SENSE5[2:0]			FILTEN4	SENSE4[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FILTEN3	SENSE3[2:0]			FILTEN2	SENSE2[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FILTEN1	SENSE1[2:0]			FILTEN0	SENSE0[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31, 27, 23, 19, 15, 11, 7 – FILTENx: Filter 0 Enable**
  0: Filter is disabled for EXTINT[n*8+x] input.
  1: Filter is enabled for EXTINT[n*8+x] input.
- **Bits 30:28, 26:24, 22:20, 18:16, 14:12, 10:8, 6:4 – SENSEx: Input Sense 0 Configuration**
  These bits define on which edge or level the interrupt or event for EXTINT[n*8+x] will be generated.

Table 20-5. Input Sense 0 Configuration

SENSE0[2:0]	Name	Description
0x0	NONE	No detection
0x1	RISE	Rising-edge detection
0x2	FALL	Falling-edge detection
0x3	BOTH	Both-edges detection

SENSE0[2:0]	Name	Description
0x4	HIGH	High-level detection
0x5	LOW	Low-level detection
0x6-0x7		Reserved

21. NVMCTRL – Non-Volatile Memory Controller

21.1 Overview

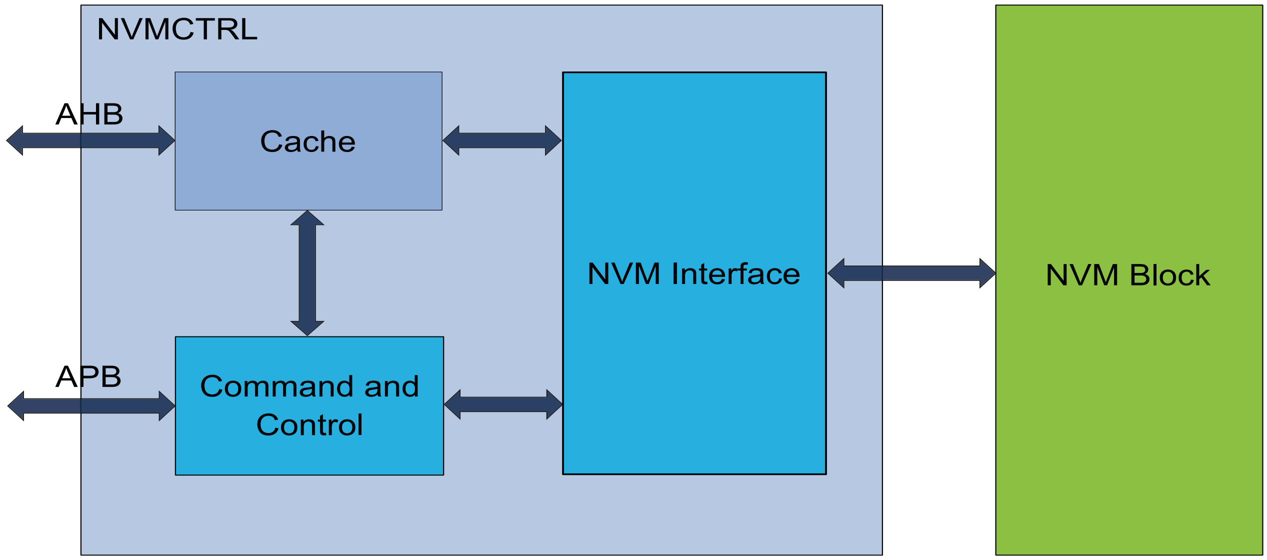
Non-volatile memory (NVM) is a reprogrammable flash memory that retains program and data storage even with power off. The NVM Controller (NVMCTRL) connects to the AHB and APB bus interfaces for system access to the NVM block. The AHB interface is used for reads and writes to the NVM block, while the APB interface is used for commands and configuration.

21.2 Features

- 32-bit AHB interface for reads and writes
- All NVM sections are memory mapped to the AHB, including calibration and system configuration
- 32-bit APB interface for commands and control
- Programmable wait states for read optimization
- 16 regions can be individually protected or unprotected
- Additional protection for boot loader
- Supports device protection through a security bit
- Interface to Power Manager for power-down of flash blocks in sleep modes
- Can optionally wake up on exit from sleep or on first access
- Direct-mapped cache

21.3 Block Diagram

Figure 21-1. Block Diagram



21.4 Signal Description

Not applicable

21.5 Product Dependencies

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

21.5.1 Power Management

The NVMCTRL will continue to operate in any sleep mode where the selected source clock is running. The NVMCTRL's interrupts can be used to wake up the device from sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

The Power Manager will automatically put the NVM block into a low-power state when entering sleep mode. This is based on the Control B register (CTRLB - refer to CTRLB) SLEEPPRM bit setting. Read the CTRLB register description for more details.

21.5.2 Clocks

Two synchronous clocks are used by the NVMCTRL. One is provided by the AHB bus (CLK_NVMCTRL_AHB) and the other is provided by the APB bus (CLK_NVMCTRL_APB). For higher system frequencies, a programmable number of wait states can be used to optimize performance. When changing the AHB bus frequency, the user must ensure that the NVM Controller is configured with the proper number of wait states. Refer to the "Electrical Characteristics" on page 648 for the exact number of wait states to be used for a particular frequency range.

21.5.3 Interrupts

The NVM Controller interrupt request line is connected to the interrupt controller. Using the NVMCTRL interrupt requires the interrupt controller to be programmed first.

Refer to "Nested Vector Interrupt Controller" on page 23 for details.

21.5.4 Debug Operation

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

Access to the NVM block can be protected by the security bit. In this case, the NVM block will not be accessible. See "Security Bit" on page 358 for details.

21.5.5 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Interrupt Flag Status and Clear register (INTFLAG - refer to INTFLAG)
- Status register (STATUS - refer to STATUS)

Write-protection is denoted by the Write-Protected property in the register description. Write-protection does not apply for accesses through an external debugger.

When the CPU is halted in debug mode, all write-protection is automatically disabled. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

21.5.6 Analog Connections

Not applicable.

21.6 Functional Description

21.6.1 Principle of Operation

The NVM Controller is a slave on the AHB and APB buses. It responds to commands, read requests and write requests, based on user configuration.

21.6.2 Basic Operations

21.6.2.1 Initialization

After power up, the NVM Controller goes through a power-up sequence. During this time, access to the NVM Controller from the AHB bus is halted. Upon power-up completion, the NVM Controller is operational without any need for user configuration.

21.6.2.2 Enabling, Disabling and Resetting

Not applicable.

21.6.3 Memory Organization

Refer to "Physical Memory Map" on page 18 for memory sizes and addresses for each device.

The NVM is organized into rows, where each row contains four pages, as shown in Figure 21-2. The NVM has a row-erase granularity, while the write granularity is by page. In other words, a single row erase will erase all four pages in the row, while four write operations are used to write the complete row.

Figure 21-2. Row Organization

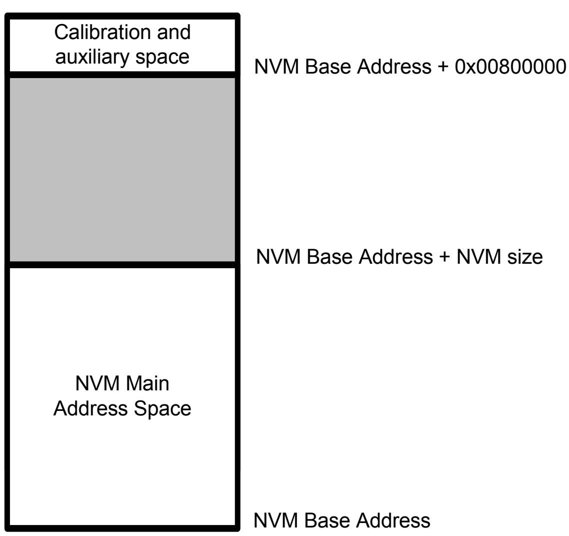
Row n	Page (n * 4) + 3	Page (n * 4) + 2	Page (n * 4) + 1	Page (n * 4) + 0

The NVM block contains a calibration and auxiliary space that is memory mapped. Refer to Figure 21-3 for details.

The calibration and auxiliary space contains factory calibration and system configuration information. This space can be read from the AHB bus in the same way as the main NVM main address space.

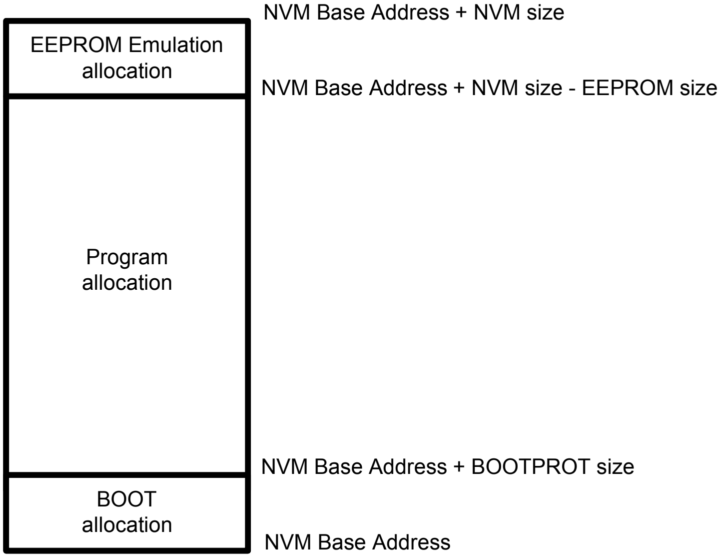
In addition, a boot loader section can be allocated at the beginning of the main array, and an EEPROM emulation area can be allocated at the end of the NVM main address space.

Figure 21-3. NVM Memory Organization



The lower rows in the NVM main address space can be allocated as a boot loader section by using the BOOTPROT fuses, and the upper rows can be allocated to EEPROM emulation, as shown in Figure 21-4. The boot loader section is protected by the lock bit(s) corresponding to this address space and by the BOOTPROT[2:0] fuse. The EEPROM rows can be written regardless of the region lock status. The number of rows protected by BOOTPROT and the number of rows allocated to EEPROM emulation are given in Table 21-2 and Table 21-3, respectively.

Figure 21-4. EEPROM Emulation and Boot Loader Allocation



21.6.4 Region Lock Bits

The NVM block is grouped into 16 equally sized regions. The region size is dependent on the flash memory size, and is given in the table below. Each region has a dedicated lock bit preventing writing and erasing pages in the region. After production, all regions will be unlocked.

Table 21-1. Region Size

Memory Size [KB]	Region Size [KB]
256	16
128	8
64	4
32	2

To lock or unlock a region, the Lock Region and Unlock Region commands are provided. Writing one of these commands will temporarily lock/unlock the region containing the address loaded in the ADDR register. ADDR can be written by software, or the automatically loaded value from a write operation can be used. The new setting will stay in effect until the next reset, or the setting can be changed again using the lock and unlock commands. The current status of the lock can be determined by reading the LOCK register.

To change the default lock/unlock setting for a region, the user configuration section of the auxiliary space must be written using the Write Auxiliary Page command. Writing to the auxiliary space will take effect after the next reset. Therefore, a boot of the device is needed for changes in the lock/unlock setting to take effect. See "Physical Memory Map" on page 18 for calibration and auxiliary space address mapping.

21.6.5 Command and Data Interface

The NVM Controller is addressable from the APB bus, while the NVM main address space is addressable from the AHB bus. Read and automatic page write operations are performed by addressing the NVM main address space directly, while other operations such as manual page writes and row erase must be performed by issuing commands through the NVM Controller.

To issue a command, the CTRLA.CMD bits must be written along with the CTRLA.CMDEX value. When a command is issued, INTFLAG.READY will be cleared until the command has completed. Any commands written while INTFLAG.READY is low will be ignored. Read the CTRLA register description for more details.

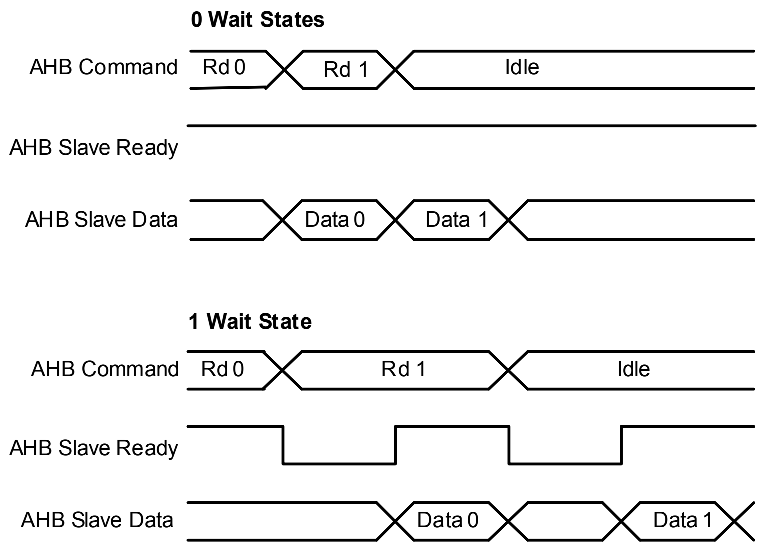
The CTRLB register must be used to control the power reduction mode, read wait states and the write mode.

21.6.5.1 NVM Read

Reading from the NVM main address space is performed via the AHB bus by addressing the NVM main address space or auxiliary address space directly. Read data is available after the configured number of read wait states (CTRLB.RWS) set in the NVM Controller, has passed.

The number of cycles data are delayed to the AHB bus is determined by the read wait states. Examples of using zero and one wait states are shown in Figure 21-5.

Figure 21-5. Read Wait State Examples



21.6.5.2 NVM Write

The NVM Controller requires that an erase must be done before programming. The entire NVM main address space can be erased by a debugger Chip Erase command. Alternatively, rows can be individually erased by the Erase Row command.

After programming, the region that the page resides in can be locked to prevent spurious write or erase sequences. Locking is performed on a per-region basis, and so locking a region locks all pages inside the region.

Data to be written to the NVM block are first written and stored in an internal buffer called the page buffer. The page buffer contains the same number of bytes as an NVM page. Writes to the page buffer must be 16 or 32 bits. 8-bit writes to the page buffer is not allowed, and will cause a system exception.

Writing to the NVM block via the AHB bus is performed by a load operation to the page buffer. For each AHB bus write, the address is stored in the ADDR register. After the page buffer has been loaded with the required number of bytes, the page can be written to the addressed location by setting CMD to Write Page and setting the key value to CMDEX. The LOAD bit in the STATUS register indicates whether the page buffer has been loaded or not. Before writing the page to memory, the accessed row must be erased.

By default, automatic page writes are enabled (MANW=0). This will trigger a write operation to the page addressed by ADDR when the last location of the page is written.

Because the address is automatically stored in ADDR during the I/O bus write operation, the last given address will be present in the ADDR register. There is no need to load the ADDR register manually, unless a different page in memory is to be written.

Procedure for Manual Page Writes (MANW=1)

The row to be written must be erased before the write command is given.

- Write to the page buffer by addressing the NVM main address space directly
- Write the page buffer to memory: CMD=Write Page and CMDEX
- The READY bit in the INTFLAG register will be low while programming is in progress, and access through the AHB will be stalled

Procedure for Automatic Page Writes (MANW=0)

The row to be written must be erased before the last write to the page buffer is performed.

Note that partially written pages must be written with a manual write.

- Write to the page buffer by addressing the NVM main address space directly.
  - When the last location in the page buffer is written, the page is automatically written to NVM main address space.
- INTFLAG.READY will be zero while programming is in progress and access through the AHB will be stalled.

21.6.5.3 Page Buffer Clear

The page buffer is automatically cleared to all ones after a page write is performed. If a partial page has been written and it is desired to clear the contents of the page buffer, the Page Buffer Clear command can be used.

21.6.5.4 Erase Row

Before a page can be written, the row that contains the page must be erased. The Erase Row command can be used to erase the desired row. Erasing the row sets all bits to one. If the row resides in a region that is locked, the erase will not be performed and the Lock Error bit in the Status register (STATUS.LOCKE) will be set.

Procedure for Erase Row

- Write the address of the row to erase ADDR. Any address within the row can be used.
- Issue an Erase Row command.

21.6.5.5 Lock and Unlock Region

These commands are used to lock and unlock regions as detailed in section "Region Lock Bits" on page 355.

21.6.5.6 Set and Clear Power Reduction Mode

The NVM Controller and block can be taken in and out of power reduction mode through the set and clear power reduction mode commands. When the NVM Controller and block are in power reduction mode, the Power Reduction Mode bit in the Status register (STATUS.PRM) is set.

21.6.6 NVM User Configuration

The NVM user configuration resides in the auxiliary space. See "Physical Memory Map" on page 18 for calibration and auxiliary space address mapping.

The bootloader resides in the main array starting at offset zero. The allocated boot loader section is protected against write.

Table 21-2. Boot Loader Size

BOOTPROT [2:0]	Rows Protected by BOOTPROT	Boot Loader Size in Bytes
7	None	0
6	2	512
5	4	1024
4	8	2048

BOOTPROT [2:0]	Rows Protected by BOOTPROT	Boot Loader Size in Bytes
3	16	4096
2	32	8192
1	64	16384
0	128	32768

The EEPROM bits indicates the Flash size reserved for EEPROM emulation according to the Table 21-3. EEPROM resides in the upper rows of the NVM main address space and are writable, regardless of the region lock status.

Table 21-3. Flash size for EEPROM emulation

EEPROM[2:0]	Rows Allocated to EEPROM	EEPROM Size in Bytes for EEPROM emulation[(1)]
7	None	0
6	1	256
5	2	512
4	4	1024
3	8	2048
2	16	4096
1	32	8192
0	64	16384

Note: 1. the actual size of the EEPROM depends on the emulation software. For more information see Application Note AT03265

21.6.7 Security Bit

The security bit allows the entire chip to be locked from external 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 a debugger Chip Erase command. After issuing the SSB command, the PROGE error bit can be checked. Refer to "DSU – Device Service Unit" on page 36 for details.

21.6.8 Cache

The NVM Controller cache reduces the device power consumption and improves system performance when wait states are required. It is a direct-mapped cache that implements 8 lines of 64 bits (i.e., 64 bytes). NVM Controller cache can be enabled by writing a zero in the CACHEDIS bit in the CTRLB register (CTRLB.CACHEDIS). Cache can be configured to three different modes using the READMODE bit group in the CTRLB register. Refer to CTRLB register description for more details. The INVALL command can be issued through the CTRLA register to invalidate all cache lines. Commands affecting NVM content automatically invalidate cache lines.

21.7 Register Summary

Table 21-4. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		CMD[6:0]						
0x01		15:8	CMDEX[7:0]							
0x02	Reserved									
0x03	Reserved									
0x04	CTRLB	7:0	MANW			RWS[3:0]				
0x05		15:8							SLEEPPRM[1:0]	
0x06		23:16						CACHEDIS	READMODE[1:0]	
0x07		31:24								
0x08	PARAM	7:0	NVMP[7:0]							
0x09		15:8	NVMP[15:8]							
0x0A		23:16						PSZ[2:0]		
0x0B		31:24								
0x0C	INTENCLR	7:0							ERROR	READY
0x0D ... 0x0F	Reserved									
0x10	INTENSET	7:0							ERROR	READY
0x11 ... 0x13	Reserved									
0x14	INTFLAG	7:0							ERROR	READY
0x15 ... 0x17	Reserved									
0x18	STATUS	7:0				NVME	LOCKE	PROGE	LOAD	PRM
0x19		15:8								SB
0x1A	Reserved									
0x1B	Reserved									
0x1C	ADDR	7:0	ADDR[7:0]							
0x1D		15:8	ADDR[15:8]							
0x1E		23:16			ADDR[21:16]					
0x1F		31:24								
0x20	LOCK	7:0	LOCK[7:0]							
0x21		15:8	LOCK[15:8]							

21.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to the "Register Access Protection" on page 353 and the "PAC – Peripheral Access Controller" on page 27 for details.

21.8.1 Control A

Name: CTRLA

Offset: 0x00

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
	CMDEX[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		CMD[6:0]						
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:8 – CMDEX[7:0]: Command Execution**
  This bit group should be written with the key value 0xA5 to enable the command written to CMD to be executed. If the bit group is written with a different key value, the write is not performed and the PROGE status bit is set. PROGE is also set if the a previously written command is not complete.

  The key value must be written at the same time as CMD. If a command is issued through the APB bus on the same cycle as an AHB bus access, the AHB bus access will be given priority. The command will then be executed when the NVM block and the AHB bus are idle.

  The READY status must be one when the command is issued.

  Bit 0 of the CMDEX bit group will read back as one until the command is issued.

Table 21-5. Command Execution

CMDEX[7:0]	Name	Description
0x0-0xa4		Reserved
0xA5	KEY	Execution Key
0xa6-0xff		Reserved

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 6:0 – CMD[6:0]: Command**
  These bits define the command to be executed when the CMDEX key is written, as shown in Table 21-6.

Table 21-6. Command

CMD[6:0]	Name	Description
0x0-0x1		Reserved
0x2	ER	Erase Row - Erases the row addressed by the ADDR register.
0x3		Reserved
0x4	WP	Write Page - Writes the contents of the page buffer to the page addressed by the ADDR register.
0x5	EAR	Erase Auxiliary Row - Erases the auxiliary row addressed by the ADDR register. This command can be given only when the security bit is not set and only to the user configuration row.
0x6	WAP	Write Auxiliary Page - Writes the contents of the page buffer to the page addressed by the ADDR register. This command can be given only when the security bit is not set and only to the user configuration row.
0x7-0x9		Reserved
0xA	SF	Security Flow Command
0xb-0xe		Reserved
0xF	WL	Write lockbits
0x10-0x3f		Reserved
0x40	LR	Lock Region - Locks the region containing the address location in the ADDR register.
0x41	UR	Unlock Region - Unlocks the region containing the address location in the ADDR register.
0x42	SPRM	Sets the power reduction mode.
0x43	CPRM	Clears the power reduction mode.
0x44	PBC	Page Buffer Clear - Clears the page buffer.
0x45	SSB	Set Security Bit - Sets the security bit by writing 0x00 to the first byte in the lockbit row.
0x46	INVALL	Invalidates all cache lines.
0x47-0x7f		Reserved

21.8.2 Control B

Name: CTRLB

Offset: 0x04

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
						CACHEDIS	READMODE[1:0]	
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
							SLEEPPRM[1:0]	
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MANW			RWS[3:0]				
Access	R/W	R	R	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:19 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 18 – CACHEDIS: Cache Disable**
  This bit is used to disable the cache.

  0: The cache is enabled.

  1: The cache is disabled.

- **Bits 17:16 – READMODE[1:0]: NVMCTRL Read Mode**

Table 21-7. NVMCTRL Read Mode

READMODE[1:0]	Name	Description
0x0	NO_MISS_PENALTY	The NVM Controller (cache system) does not insert wait states on a cache miss. Gives the best system performance.
0x1	LOW_POWER	Reduces power consumption of the cache system, but inserts a wait state each time there is a cache miss. This mode may not be relevant if CPU performance is required, as the application will be stalled and may lead to increase run time.
0x2	DETERMINISTIC	The cache system ensures that a cache hit or miss takes the same amount of time, determined by the number of programmed flash wait states. This mode can be used for real-time applications that require deterministic execution timings.
0x3		Reserved

- **Bits 15:10 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 9:8 – SLEEPPRM[1:0]: Power Reduction Mode during Sleep**
  Indicates the power reduction mode during sleep.

Table 21-8. Power Reduction Mode during Sleep

SLEEPPRM[1:0]	Name	Description
0x0	WAKEONACCESS	NVM block enters low-power mode when entering sleep.NVM block exits low-power mode upon first access.
0x1	WAKEUPINSTANT	NVM block enters low-power mode when entering sleep.NVM block exits low-power mode when exiting sleep.
0x2		Reserved
0x3	DISABLED	Auto power reduction disabled.

- **Bit 7 – MANW: Manual Write**
  0: Writing to the last word in the page buffer will initiate a write operation to the page addressed by the last write operation. This includes writes to memory and auxiliary rows.

  1: Write commands must be issued through the CMD register.

- **Bits 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:1 – RWS[3:0]: NVM Read Wait States**
  These bits give the number of wait states for a read operation. Zero indicates zero wait states, one indicates one wait state, etc., up to 15 wait states.

  This register is initialized to 0 wait states. Software can change this value based on the NVM access time and system frequency.

- **Bit 0 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

21.8.3 NVM Parameter

Name: PARAM

Offset: 0x08

Reset: 0X000000000000XXXXXXXXXXXXXXXXXXX

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
						PSZ[2:0]		
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	X	X	X

Bit	15	14	13	12	11	10	9	8
	NVMP[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	X	X	X	X	X	X	X	X

Bit	7	6	5	4	3	2	1	0
	NVMP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	X	X	X	X	X	X	X	X

- **Bits 31:19 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 18:16 – PSZ[2:0]: Page Size**
  Indicates the page size. Not all device families will provide all the page sizes indicated in the table.

Table 21-9. Page Size

PSZ[2:0]	Name	Description
0x0	8	8 bytes
0x1	16	16 bytes
0x2	32	32 bytes
0x3	64	64 bytes
0x4	128	128 bytes

PSZ[2:0]	Name	Description
0x5	256	256 bytes
0x6	512	512 bytes
0x7	1024	1024 bytes

- **Bits 15:0 – NVMP[15:0]: NVM Pages**
  Indicates the number of pages in the NVM main address space.

21.8.4 Interrupt Enable Clear

Name: INTENCLR

Offset: 0x0C

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
							ERROR	READY
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – ERROR: Error Interrupt Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the ERROR interrupt enable.
  This bit will read as the current value of the ERROR interrupt enable.

- **Bit 0 – READY: NVM Ready Interrupt Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the READY interrupt enable.
  This bit will read as the current value of the READY interrupt enable.

21.8.5 Interrupt Enable Set

Name: INTENSET

Offset: 0x10

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
							ERROR	READY
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – ERROR: Error Interrupt Enable**
  Writing a zero to this bit has no effect.

  Writing a one to this bit sets the ERROR interrupt enable.

  This bit will read as the current value of the ERROR interrupt enable.

- **Bit 0 – READY: NVM Ready Interrupt Enable**
  Writing a zero to this bit has no effect.

  Writing a one to this bit sets the READY interrupt enable.

  This bit will read as the current value of the READY interrupt enable.

21.8.6 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x14

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
							ERROR	READY
Access	R	R	R	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – ERROR: Error**
  This flag is set on the occurrence of an NVME, LOCKE or PROGE error.
  0: No errors have been received since the last clear.
  1: At least one error has occurred since the last clear.
  This bit can be cleared by writing a one to its bit location.

- **Bit 0 – READY: NVM Ready**
  0: The NVM controller is busy programming or erasing.
  1: The NVM controller is ready to accept a new command.

21.8.7 Status

Name: STATUS

Offset: 0x18

Reset: 0X0000000X00000000

Access: Read-Write

Property: -

Bit	15	14	13	12	11	10	9	8
								SB
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	X

Bit	7	6	5	4	3	2	1	0
				NVME	LOCKE	PROGE	LOAD	PRM
Access	R	R	R	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bits 15:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 8 – SB: Security Bit Status**
  0: The Security bit is inactive.
  1: The Security bit is active.

- **Bits 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 4 – NVME: NVM Error**
  0: No programming or erase errors have been received from the NVM controller since this bit was last cleared.
  1: At least one error has been registered from the NVM Controller since this bit was last cleared.
  This bit can be cleared by writing a one to its bit location.

- **Bit 3 – LOCKE: Lock Error Status**
  0: No programming of any locked lock region has happened since this bit was last cleared.
  1: Programming of at least one locked lock region has happened since this bit was last cleared.
  This bit can be cleared by writing a one to its bit location.

- **Bit 2 – PROGE: Programming Error Status**
  0: No invalid commands or bad keywords were written in the NVM Command register since this bit was last cleared.
  1: An invalid command and/or a bad keyword was/were written in the NVM Command register since this bit was last cleared.
  This bit can be cleared by writing a one to its bit location.

- **Bit 1 – LOAD: NVM Page Buffer Active Loading**
  This bit indicates that the NVM page buffer has been loaded with one or more words. Immediately after an NVM load has been performed, this flag is set, and it remains set until a page write or a page buffer clear (PBCLR) command is given.

  This bit can be cleared by writing a one to its bit location.

- **Bit 0 – PRM: Power Reduction Mode**
  This bit indicates the current NVM power reduction state. The NVM block can be set in power reduction mode in two ways: through the command interface or automatically when entering sleep with SLEEPPRM set accordingly.

  PRM can be cleared in three ways: through AHB access to the NVM block, through the command interface (SPRM and CPRM) or when exiting sleep with SLEEPPRM set accordingly.

  0: NVM is not in power reduction mode.

  1: NVM is in power reduction mode.

21.8.8 Address

Name: ADDR

Offset: 0x1C

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			ADDR[21:16]					
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:22 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 21:0 – ADDR[21:0]: NVM Address**
  ADDR drives the hardware (16-bit) address to the NVM when a command is executed using CMDEX. 8-bit addresses must be shifted one bit to the right before writing to this register.

  This register is automatically updated when writing to the page buffer, and can also be manually written. This register holds the address offset for the section addressed.

21.8.9 Lock Section

Name: LOCK

Offset: 0x20

Reset: -

Access: Read-Write

Property: -

Bit	15	14	13	12	11	10	9	8
	LOCK[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LOCK[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – LOCK[15:0]: Region Lock Bits**
  In order to set or clear these bits, the CMD register must be used.
  0: The corresponding lock region is locked.
  1: The corresponding lock region is not locked.

22. PORT

22.1 Overview

The Port (PORT) controls the I/O pins of the microcontroller. The I/O pins are organized in a series of groups, collectively referred to as a line bundle, and each group can have up to 32 pins that can be configured and controlled individually or as a group. Each pin may either be used for general-purpose I/O under direct application control or assigned to an embedded device peripheral. When used for general-purpose I/O, each pin can be configured as input or output, with highly configurable driver and pull settings.

All I/O pins have true read-modify-write functionality when used for general-purpose I/O; the direction or the output value of one or more pins may be changed (set, reset or toggled) without unintentionally changing the state of any other pins in the same line bundle via a single, atomic 8-, 16- or 32-bit write.

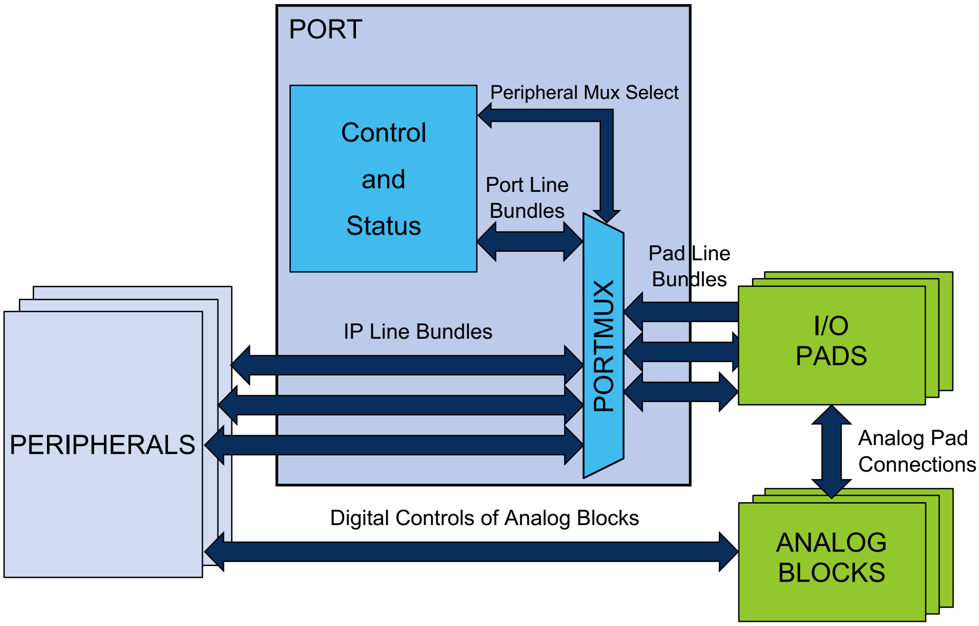
The PORT is connected to the high-speed bus matrix through an AHB/APB bridge. The Pin Direction, Data Output Value and Data Input Value registers may also be accessed using the low-latency CPU local bus (IOBUS; ARM® single-cycle I/O port).

22.2 Features

- Selectable input and output configuration individually for each pin
- Software-controlled multiplexing of peripheral functions on I/O pins
- Flexible pin configuration through a dedicated Pin Configuration register
- Configurable output driver and pull settings:
  - Totem-pole (push-pull)
  - Pull configuration
  - Driver strength
- Configurable input buffer and pull settings:
  - Internal pull-up or pull-down
  - Input sampling criteria
  - Input buffer can be disabled if not needed for lower power consumption
- Read-modify-write support for pin configuration, output value and pin direction

22.3 Block Diagram

Figure 22-1. PORT Block Diagram



22.4 Signal Description

Signal Name	Type	Description
Pxy	Digital I/O	General-purpose I/O pin y

Refer to "I/O Multiplexing and Considerations" on page 10 for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

22.5 Product Dependencies

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

22.5.1 I/O Lines

The I/O lines of the PORT are mapped to pins of the physical device package according to a simple naming scheme. Each line bundle of up to 32 pins is assigned a letter identifier, starting with A, that monotonically increases through the alphabet for each subsequent line bundle. Within each line bundle, each pin is assigned a numerical identifier according to its bit position.

The resulting PORT pins are mapped as Pxy, where x=A, B, C,… and y=00, 01, …, 31 to uniquely identify each pin in the device, e.g., PA24, PC03, etc.

Each pin may have one or more peripheral multiplexer settings, which allow the pad to be routed internally to a dedicated peripheral function. When enabled, the selected peripheral is given control over the output state of the pad, as well as the ability to read the current physical pad state. Refer to "I/O Multiplexing and Considerations" on page 10 for details.

Device-specific configurations may result in some pins (and the corresponding Pxy pin) not being implemented.

22.5.2 Power Management

During reset, all PORT lines are configured as inputs with input buffers, output buffers and pull disabled.

If the PORT peripheral is shut down, the latches contained in the pad will retain their current configuration, such as the output value and pull settings. However, the PORT configuration registers and input synchronizers will lose their contents, and these will not be restored when PORT is powered up again. The user must, therefore, reconfigure the PORT peripheral at power up to ensure it is in a well-defined state before use.

The PORT will continue to operate in any sleep mode where the selected module source clock is running.

22.5.3 Clocks

The PORT bus clock (CLK_PORT_APB) can be enabled and disabled in the Power Manager, and the default state of CLK_PORT_APB can be found in the Peripheral Clock Masking section in the "PM – Power Manager" on page 107.

The PORT is fed by two different clocks: a CPU main clock, which allows the CPU to access the PORT through the low-latency CPU local bus (IOBUS), and an APB clock, which is a divided clock of the CPU main clock and allows the CPU to access the PORT registers through the high-speed matrix and the AHB/APB bridge.

IOBUS accesses have priority over APB accesses. The latter must insert wait states in the event of concurrent PORT accesses.

The PORT input synchronizers use the CPU main clock so that the resynchronization delay is minimized with respect to the APB clock.

22.5.4 DMA

Not applicable.

22.5.5 Interrupts

Not applicable.

22.5.6 Events

Not applicable.

22.5.7 Debug Operation

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

22.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC).

Write-protection is denoted by the Write-Protected property in the register description.

When the CPU is halted in debug mode, all write-protection is automatically disabled.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

22.5.9 Analog Connections

Analog functions are connected directly between the analog blocks and the I/O pads using analog buses. However, selecting an analog peripheral function for a given pin will disable the corresponding digital features of the pad.

22.5.10 CPU Local Bus

The CPU local bus (IOBUS) is an interface that connects the CPU directly to the PORT. It is a single-cycle bus interface, and does not support wait states. It supports byte, half word and word sizes.

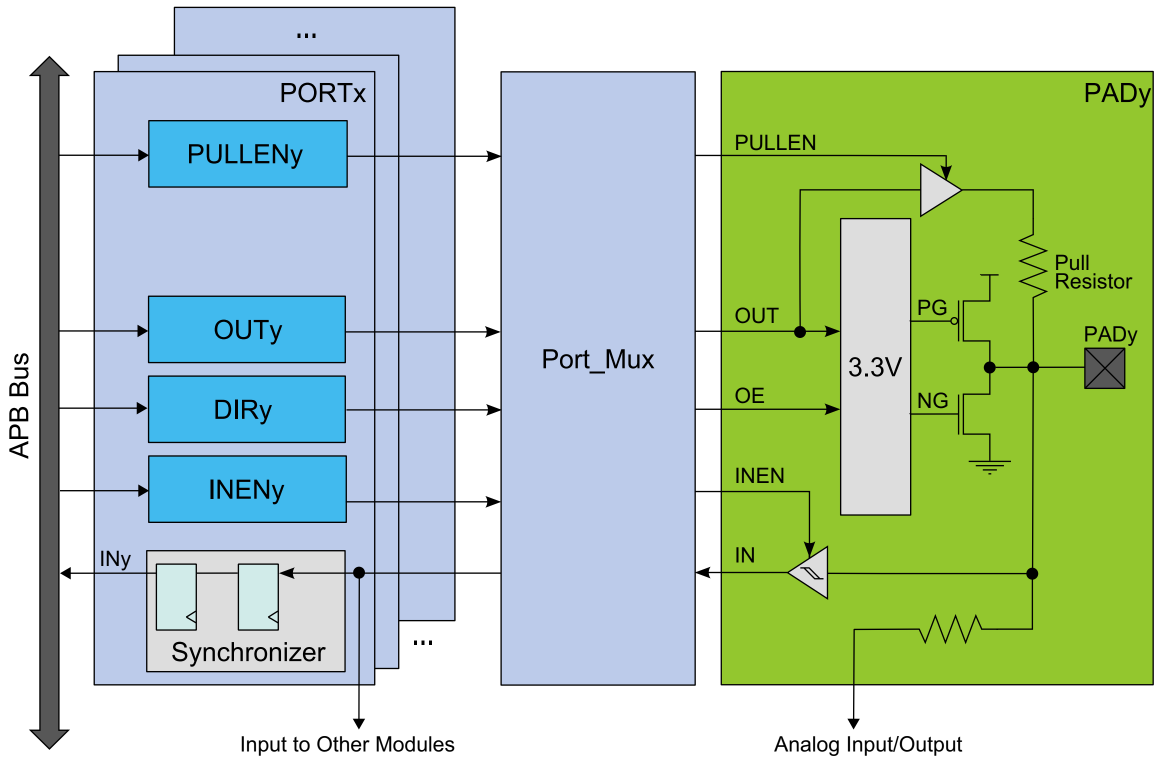
The CPU accesses the PORT module through the IOBUS when it performs read or write from address 0x60000000. The PORT register map is equivalent to the one described in the register description section.

This bus is generally used for low latency. The Data Direction (DIRn) and Data Output Value (OUTn) registers can be read, written, set, cleared or toggled using this bus, and the Data Input Value (INn) registers can be read.

Since the IOBUS cannot wait for IN register resynchronization, the Control register (CTRLn) must be configured to enable continuous sampling of all pins that will need to be read via the IOBUS to prevent stale data from being read.

22.6 Functional Description

Figure 22-2. Overview of the PORT



22.6.1 Principle of Operation

The I/O pins of the device are controlled by reads and writes of the PORT peripheral registers. For each port pin, a corresponding bit in the Data Direction (DIRn) and Data Output Value (OUTn) registers are used to enable that pin as an output and to define the output state.

The direction of each pin in a port group is configured via the DIR register. If a bit in DIR is written to one, the corresponding pin is configured as an output pin. If a bit in DIR is written to zero, the corresponding pin is configured as an input pin.

When the direction is set as output, the corresponding bit in the OUT register is used to set the level of the pin. If bit y of OUT is written to one, pin y is driven high. If bit y of OUT is written to zero, pin y is driven low.

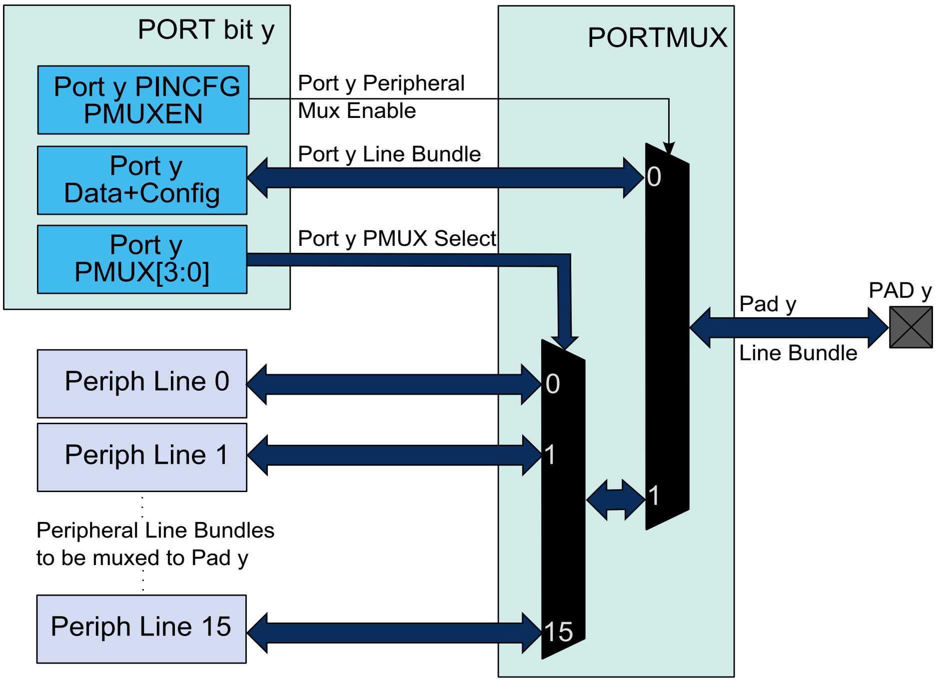
Additional pin configuration can be set by writing to the Pin Configuration (PINCFG0) registers.

The Data Input Value bit (INn) is used to read the port pin with resynchronization to the PORT clock. By default, these input synchronizers are clocked only when an input value read is requested in order to reduce power consumption. Input value can always be read, whether the pin is configured as input or output, except if digital input is disabled by writing a zero to the INEN bit in the Pin Configuration registers (PINCFGy).

The PORT also allows peripheral functions to be connected to individual I/O pins by writing a one to the corresponding PMUXEN bit in the PINCFGy registers and by writing the chosen selection to the Peripheral Multiplexing registers (PMUX0) for that pin. This will override the connection between the PORT and that I/O pin, and connect the selected peripheral line bundle to the pad instead of the PORT line bundle.

Each group of up to 32 pins is controlled by a set of registers, as described in Figure 22-3. This set of registers is duplicated for each group of pins, with increasing base addresses.

Figure 22-3. Overview of the Peripheral Functions Multiplexing



22.6.2 Basic Operation

22.6.2.1 Initialization

After reset, all standard-function device I/O pins are connected to the PORT with outputs tri-stated and input buffers disabled, even if no clocks are running. Specific pins, such as the ones used for connection to a debugger, may be configured differently, as required by their special function.

22.6.3 Basic Operation

Each I/O pin y can be configured and accessed by reading or writing PORT registers. Because PORT registers are grouped into sets of registers for each group of up to 32 pins, the base address of the register set for pin y is at byte address PORT + (y / 32) * 0x80. (y%32) will be used as the index within that register set.

To use pin y as an output, configure it as output by writing the (y%32) bit in the DIR register to one. To avoid disturbing the configuration of other pins in that group, this can also be done by writing the (y%32) bit in the DIRSET register to one. The desired output value can be set by writing the (y%32) bit to that value in register OUT.

Similarly, writing an OUTSET bit to one will set the corresponding bit in the OUT register to one, while writing an OUTCLR bit to one will set it to zero, and writing an OUTTGL bit to one will toggle that bit in OUT.

To use pin y as an input, configure it as input by writing the (y%32) bit in the DIR register to zero. To avoid disturbing the configuration of other pins in that group, this can also be done by writing the (y%32) bit in DIRCLR register to one. The desired input value can be read from the (y%32) bit in register IN as soon as the INEN bit in the Pin Configuration register (PINCFGy) is written to one. Refer to "I/O Multiplexing and Considerations" on page 10 for details on pin configuration.

By default, the input synchronizer is clocked only when an input read is requested, which will delay the read operation by two CLK_PORT cycles. To remove that delay, the input synchronizers for each group of eight pins can be configured to be always active, but this comes at the expense of higher power consumption. This is controlled by writing a one to the corresponding SAMPLINGn bit group of the CTRL register, where n = (y%32) / 8.

To use pin y as one of the available peripheral functions for that pin, configure it by writing a one to the corresponding PMUXEN bit of the PINCFGy register. The PINCFGy register for pin y is at byte offset (PINCFG0 + (y%32)).

The peripheral function can be selected by writing to the PMUXO or PMUXE bit group in the PMUXn register. The PMUXO/PMUXE bit group is at byte offset (PMUX0 + (y%32) / 2), in bits 3:0 if y is even and in bits 7:4 if y is odd.

The chosen peripheral must also be configured and enabled.

22.6.4 I/O Pin Configuration

The Pin Configuration register (PINCFGy) is used for additional I/O pin configuration. A pin can be set in a totem-pole, open-drain or pull configuration.

Because pull configuration is done through the Pin Configuration register, all intermediate PORT states during switching of pin direction and pin values are avoided.

The I/O pin configurations are described further in this chapter, and summarized in Table 22-1.

22.6.4.1 Pin Configurations Summary

Table 22-1. Pin Configurations Summary

DIR	INEN	PULLEN	OUT	Configuration
0	0	0	X	Reset or analog I/O; all digital disabled
0	0	1	0	Pull-down; input disabled
0	0	1	1	Pull-up; input disabled
0	1	0	X	Input
0	1	1	0	Input with pull-down
0	1	1	1	Input with pull-up
1	0	X	X	Output; input disabled
1	1	X	X	Output; input enabled

22.6.4.2 Input Configuration

Figure 22-4. I/O Configuration - Standard Input

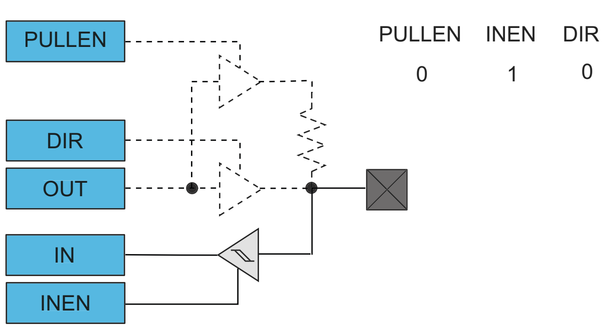
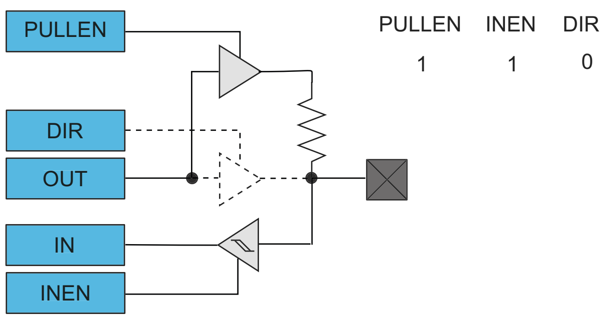


Figure 22-5. I/O Configuration - Input with Pull



Note that when pull is enabled, the pull value is defined by the OUTx value.

22.6.4.3 Totem-Pole Output

When configured for totem-pole (push-pull) output, the pin is driven low or high according to the corresponding bit setting in the OUT register. In this configuration, there is no current limitation for sink or source other than what the pin is capable of. If the pin is configured for input, the pin will float if no external pull is connected. Note, that enabling the output driver automatically disables pull.

Figure 22-6. I/O Configuration - Totem-Pole Output with Disabled Input

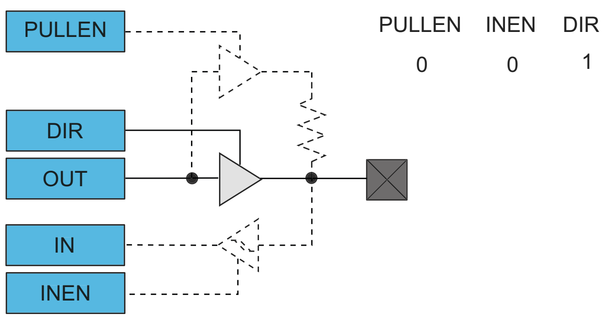


Figure 22-7. I/O Configuration - Totem-Pole Output with Enabled Input

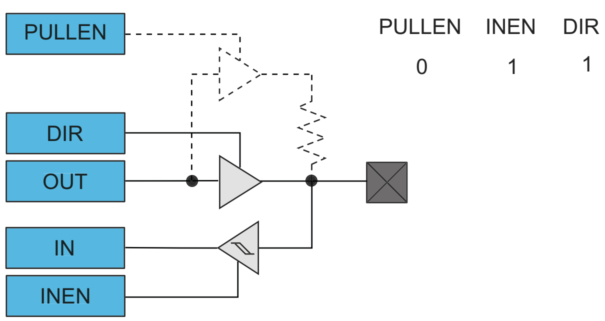
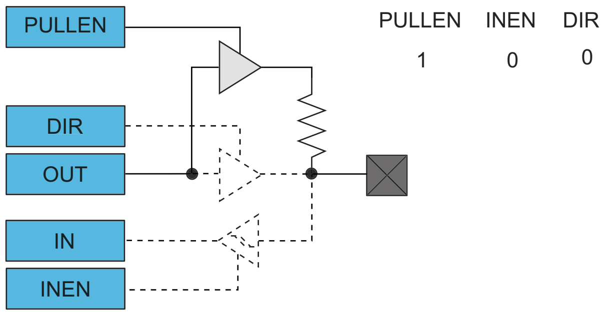
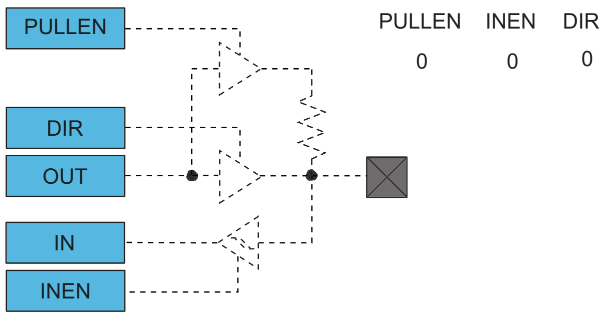


Figure 22-8. I/O Configuration - Output with Pull



22.6.4.4 Digital Functionality Disabled

Figure 22-9. I/O Configuration - Reset or Analog I/O: Digital Output, Input and Pull Disabled



22.7 Register Summary

The I/O pins are organized in groups with up to 32 pins. Group 0 consists of the PA pins, group 1 the PB pins, etc. Each group has its own set of registers. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, while the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

Table 22-2. Register Summary

Offset	Name	Bit Pos.								
Mode GROUP										
0x00	DIR	7:0	DIR[7:0]							
0x01		15:8	DIR[15:8]							
0x02		23:16	DIR[23:16]							
0x03		31:24	DIR[31:24]							
0x04	DIRCLR	7:0	DIRCLR[7:0]							
0x05		15:8	DIRCLR[15:8]							
0x06		23:16	DIRCLR[23:16]							
0x07		31:24	DIRCLR[31:24]							
0x08	DIRSET	7:0	DIRSET[7:0]							
0x09		15:8	DIRSET[15:8]							
0x0A		23:16	DIRSET[23:16]							
0x0B		31:24	DIRSET[31:24]							
0x0C	DIRTGL	7:0	DIRTGL[7:0]							
0x0D		15:8	DIRTGL[15:8]							
0x0E		23:16	DIRTGL[23:16]							
0x0F		31:24	DIRTGL[31:24]							
0x10	OUT	7:0	OUT[7:0]							
0x11		15:8	OUT[15:8]							
0x12		23:16	OUT[23:16]							
0x13		31:24	OUT[31:24]							
0x14	OUTCLR	7:0	OUTCLR[7:0]							
0x15		15:8	OUTCLR[15:8]							
0x16		23:16	OUTCLR[23:16]							
0x17		31:24	OUTCLR[31:24]							
0x18	OUTSET	7:0	OUTSET[7:0]							
0x19		15:8	OUTSET[15:8]							
0x1A		23:16	OUTSET[23:16]							
0x1B		31:24	OUTSET[31:24]							
0x1C	OUTTGL	7:0	OUTTGL[7:0]							
0x1D		15:8	OUTTGL[15:8]							
0x1E		23:16	OUTTGL[23:16]							
0x1F		31:24	OUTTGL[31:24]							
0x20	IN	7:0	IN[7:0]							
0x21		15:8	IN[15:8]							
0x22		23:16	IN[23:16]							
0x23		31:24	IN[31:24]							

Offset	Name	Bit Pos.								
0x24	CTRL	7:0	SAMPLING[7:0]							
0x25		15:8	SAMPLING[15:8]							
0x26		23:16	SAMPLING[23:16]							
0x27		31:24	SAMPLING[31:24]							
0x28	WRCONFIG	7:0	PINMASK[7:0]							
0x29		15:8	PINMASK[15:8]							
0x2A		23:16		DRVSTR				PULLEN	INEN	PMUXEN
0x2B		31:24	HWSEL	WRPINCFG		WRPMUX	PMUX[3:0]			
0x2C ... 0x2F	Reserved									
0x30	PMUX0	7:0	PMUXO[3:0]				PMUXE[3:0]			
0x31	PMUX1	7:0	PMUXO[3:0]				PMUXE[3:0]			
...	...	...								
0x3F	PMUX15	7:0	PMUXO[3:0]				PMUXE[3:0]			
0x40	PINCFG0	7:0		DRVSTR				PULLEN	INEN	PMUXEN
0x41	PINCFG1	7:0		DRVSTR				PULLEN	INEN	PMUXEN
...	...	...								
0x5F	PINCFG31	7:0		DRVSTR				PULLEN	INEN	PMUXEN
0x60 ... 0x7F	Reserved									

22.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to "Register Access Protection" on page 377 for details.

22.8.1 Data Direction

Name: DIRn

Offset: 0x00+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	DIR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DIR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – DIR[31:0]: Port Data Direction**
  These bits set the data direction for the individual I/O pins in the PORT group.
  0: The corresponding I/O pin in the group is configured as an input.
  1: The corresponding I/O pin in the group is configured as an output.

22.8.2 Data Direction Clear

Name: DIRCLRn

Offset: 0x04+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	DIRCLR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIRCLR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIRCLR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DIRCLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to set one or more I/O pins as an input, without doing a read-modify-write operation. Changes in this register will also be reflected in the Data Direction (DIR), Data Direction Toggle (DIRTGL) and Data Direction Set (DIRSET) registers.

- **Bits 31:0 – DIRCLR[31:0]: Port Data Direction Clear**

  0: The I/O pin direction is cleared.

  1: The I/O pin direction is set.

  Writing a zero to a bit has no effect.

  Writing a one to a bit will clear the corresponding bit in the DIR register, which configures the I/O pin as an input.

22.8.3 Data Direction Set

Name: DIRSETn

Offset: 0x08+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	DIRSET[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIRSET[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIRSET[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DIRSET[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to set one or more I/O pins as an output, without doing a read-modify-write operation. Changes in this register will also be reflected in the Data Direction (DIR), Data Direction Toggle (DIRTGL) and Data Direction Clear (DIRCLR) registers.

- **Bits 31:0 – DIRSET[31:0]: Port Data Direction Set**

  0: The I/O pin direction is cleared.

  1: The I/O pin direction is set.

  Writing a zero to a bit has no effect.

  Writing a one to a bit will set the corresponding bit in the DIR register, which configures the I/O pin as an output.

22.8.4 Data Direction Toggle

Name: DIRTGLn

Offset: 0x0C+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	DIRTGL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIRTGL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIRTGL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DIRTGL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to toggle the direction of one or more I/O pins, without doing a read-modify-write operation. Changes in this register will also be reflected in the Data Direction (DIR), Data Direction Set (DIRSET) and Data Direction Clear (DIRCLR) registers.

- **Bits 31:0 – DIRTGL[31:0]: Port Data Direction Toggle**

  0: The I/O pin direction is cleared.

  1: The I/O pin direction is set.

  Writing a zero to a bit has no effect.

  Writing a one to a bit will toggle the corresponding bit in the DIR register, which reverses the direction of the I/O pin.

22.8.5 Data Output Value

Name: OUTn

Offset: 0x10+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	OUT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OUT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	OUT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OUT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register sets the data output drive value for the individual I/O pins in the PORT.

- **Bits 31:0 – OUT[31:0]: Port Data Output Value**
  These bits set the logical output drive level of I/O pins configured as outputs via the Data Direction register (DIR). For pins configured as inputs via the Data Direction register (DIR) with pull enabled via the Pull Enable register (PULLEN), these bits will set the input pull direction.
  0: The I/O pin output is driven low, or the input is connected to an internal pull-down.
  1: The I/O pin output is driven high, or the input is connected to an internal pull-up.

22.8.6 Data Output Value Clear

Name: OUTCLRn

Offset: 0x14+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	OUTCLR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OUTCLR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	OUTCLR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OUTCLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to set one or more output I/O pin drive levels low, without doing a read-modify-write operation. Changes in this register will also be reflected in the Data Output Value (OUT), Data Output Value Toggle (OUTTGL) and Data Output Value Set (OUTSET) registers.

- **Bits 31:0 – OUTCLR[31:0]: Port Data Output Value Clear**

  0: The I/O pin output is driven low.

  1: The I/O pin output is driven high.

  Writing a zero to a bit has no effect.

  Writing a one to a bit will clear the corresponding bit in the OUT register, which sets the output drive level low for I/O pins configured as outputs via the Data Direction register (DIR). For pins configured as inputs via the Data Direction register (DIR) and with pull enabled via the Pull Enable register (PULLEN), these bits will set the input pull direction to an internal pull-down.

22.8.7 Data Output Value Set

Name: OUTSETn

Offset: 0x18+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	OUTSET[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OUTSET[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	OUTSET[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OUTSET[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to set one or more output I/O pin drive levels high, without doing a read-modify-write operation. Changes in this register will also be reflected in the Data Output Value (OUT), Data Output Value Toggle (OUTTGL) and Data Output Value Clear (OUTCLR) registers.

- **Bits 31:0 – OUTSET[31:0]: Port Data Output Value Set**

  0: The I/O pin output is driven low.

  1: The I/O pin output is driven high.

  Writing a zero to a bit has no effect.

  Writing a one to a bit will set the corresponding bit in the OUT register, which sets the output drive level high for I/O pins configured as outputs via the Data Direction register (DIR). For pins configured as inputs via the Data Direction register (DIR) and with pull enabled via the Pull Enable register (PULLEN), these bits will set the input pull direction to an internal pull-up.

22.8.8 Data Output Value Toggle

Name: OUTTGLn

Offset: 0x1C+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	OUTTGL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OUTTGL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	OUTTGL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OUTTGL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to toggle the drive level of one or more output I/O pins, without doing a read-modify-write operation. Changes in this register will also be reflected in the Data Output Value (OUT), Data Output Value Set (OUTSET) and Data Output Value Clear (OUTCLR) registers.

- **Bits 31:0 – OUTTGL[31:0]: Port Data Output Value Toggle**

  0: The I/O pin output is driven low.

  1: The I/O pin output is driven high.

  Writing a zero to a bit has no effect.

  Writing a one to a bit will toggle the corresponding bit in the OUT register, which inverts the output drive level for I/O pins configured as outputs via the Data Direction register (DIR). For pins configured as inputs via the Data Direction register (DIR) and with pull enabled via the Pull Enable register (PULLEN), these bits will toggle the input pull direction.

22.8.9 Data Input Value

Name: INn

Offset: 0x20+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
	IN[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	IN[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IN[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	IN[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – IN[31:0]: Port Data Input Value**
  These bits are cleared when the corresponding I/O pin input sampler detects a logical low level on the input pin.
  These bits are set when the corresponding I/O pin input sampler detects a logical high level on the input pin.

22.8.10 Control

Name: CTRLn

Offset: 0x24+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	SAMPLING[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SAMPLING[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SAMPLING[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SAMPLING[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – SAMPLING[31:0]: Input Sampling Mode**
  Configures the input sampling functionality of the I/O pin input samplers for pins configured as inputs via the Data Direction register (DIR).

  0: The I/O pin input synchronizer is disabled.

  1: The I/O pin input synchronizer is enabled.

  The input samplers are enabled and disabled in sub-groups of eight. Thus, if any pins within a byte request continuous sampling, all pins in that eight pin sub-group will be continuously sampled.

22.8.11 Write Configuration

Name: WRCONFIGn

Offset: 0x28+n*0x80 [n=0..0]

Reset: 0x00000000

Access: Write-Only

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	HWSEL	WRPINCFG		WRPMUX	PMUX[3:0]			
Access	W	W	R	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		DRVSTR				PULLEN	INEN	PMUXEN
Access	R	W	R	R	R	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PINMASK[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PINMASK[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

This write-only register is used to configure several pins simultaneously with the same configuration and/or peripheral multiplexing.

In order to avoid the side effect of non-atomic access, 8-bit or 16-bit writes to this register will have no effect. Reading this register always returns zero.

- **Bit 31 – HWSEL: Half-Word Select**
  This bit selects the half-word field of a 32-pin group to be reconfigured in the atomic write operation.
  0: The lower 16 pins of the PORT group will be configured.
  1: The upper 16 pins of the PORT group will be configured.
  This bit will always read as zero.

- **Bit 30 – WRPINCFG: Write PINCFG**
  This bit determines whether the atomic write operation will update the Pin Configuration register (PINCFGy) or not for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits.
  0: The PINCFGy registers of the selected pins will not be updated.
  1: The PINCFGy registers of the selected pins will be updated.
  Writing a zero to this bit has no effect.

Writing a one to this bit updates the configuration of the selected pins with the written WRCONFIG.DRVSTR, WRCONFIG.SLEWLIM, WRCONFIG.ODRAIN, WRCONFIG.PULLEN, WRCONFIG.INEN, WRCONFIG.PMUXEN and WRCONFIG.PINMASK values.

This bit will always read as zero.

- **Bit 29 – Reserved**

  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 28 – WRPMUX: Write PMUX**

  This bit determines whether the atomic write operation will update the Peripheral Multiplexing register (PMUXn) or not for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits.

  0: The PMUXn registers of the selected pins will not be updated.

  1: The PMUXn registers of the selected pins will be updated.

  Writing a zero to this bit has no effect.

  Writing a one to this bit updates the pin multiplexer configuration of the selected pins with the written WRCONFIG.PMUX value.

  This bit will always read as zero.

- **Bits 27:24 – PMUX[3:0]: Peripheral Multiplexing**

  These bits determine the new value written to the Peripheral Multiplexing register (PMUXn) for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits, when the WRCONFIG.WRPMUX bit is set.

  These bits will always read as zero.

- **Bit 23 – Reserved**

  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 22 – DRVSTR: Output Driver Strength Selection**

  This bit determines the new value written to PINCFGy.DRVSTR for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits the WRCONFIG.WRPINCFG bit is set.

  This bit will always read as zero.

- **Bits 21:19 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 18 – PULLEN: Pull Enable**

  This bit determines the new value written to PINCFGy.PULLEN for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits when the WRCONFIG.WRPINCFG bit is set.

  This bit will always read as zero.

- **Bit 17 – INEN: Input Enable**

  This bit determines the new value written to PINCFGy.INEN for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits when the WRCONFIG.WRPINCFG bit is set.

  This bit will always read as zero.

- **Bit 16 – PMUXEN: Peripheral Multiplexer Enable**

  This bit determines the new value written to PINCFGy.PMUXEN for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits when the WRCONFIG.WRPINCFG bit is set.

  This bit will always read as zero.

- **Bits 15:0 – PINMASK[15:0]: Pin Mask for Multiple Pin Configuration**

  These bits select the pins to be configured within the half-word group selected by the WRCONFIG.HWSEL bit.

  0: The configuration of the corresponding I/O pin in the half-word group will be left unchanged.

1: The configuration of the corresponding I/O pin in the half-word pin group will be updated.

These bits will always read as zero.

22.8.12 Peripheral Multiplexing n

Name: PMUXnm

Offset: 0x30+n*0x80 [n=0..0]+m*0x1 [m=0..15]

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	PMUXO[3:0]				PMUXE[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

There are up to 16 Peripheral Multiplexing registers in each group, one for every set of two subsequent I/O lines. The n denotes the number of the set of I/O lines, while the x denotes the number of the group.

- **Bits 7:4 – PMUXO[3:0]: Peripheral Multiplexing Odd**
  These bits select the peripheral function for odd-numbered pins (2*n + 1) of a PORT group, if the corresponding PINCFGy.PMUXEN bit is one.
  Not all possible values for this selection may be valid. For more details, refer to "I/O Multiplexing and Considerations" on page 10.

Table 22-3. Peripheral Multiplexing Odd

PMUXO[3:0]	Name	Description
0x0	A	Peripheral function A selected
0x1	B	Peripheral function B selected
0x2	C	Peripheral function C selected
0x3	D	Peripheral function D selected
0x4	E	Peripheral function E selected
0x5	F	Peripheral function F selected
0x6	G	Peripheral function G selected
0x7	H	Peripheral function H selected
0x8-0xf		Reserved

- **Bits 3:0 – PMUXE[3:0]: Peripheral Multiplexing Even**
  These bits select the peripheral function for even-numbered pins (2*n) of a PORT group, if the corresponding PINCFGy.PMUXEN bit is one.
  Not all possible values for this selection may be valid. For more details, refer to "I/O Multiplexing and Considerations" on page 10.

Table 22-4. Peripheral Multiplexing Even

PMUXE[3:0]	Name	Description
0x0	A	Peripheral function A selected
0x1	B	Peripheral function B selected
0x2	C	Peripheral function C selected
0x3	D	Peripheral function D selected
0x4	E	Peripheral function E selected
0x5	F	Peripheral function F selected
0x6	G	Peripheral function G selected
0x7	H	Peripheral function H selected
0x8-0xf		Reserved

22.8.13 Pin Configuration n

Name: PINCFGnm

Offset: 0x40+n*0x80 [n=0..0]+m*0x1 [m=0..31]

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
		DRVSTR				PULLEN	INEN	PMUXEN
Access	R	W	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

There are up to 32 Pin Configuration registers in each group, one for each I/O line. The n denotes the number of the I/O line, while the x denotes the number of the Port group.

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 6 – DRVSTR: Output Driver Strength Selection**
  This bit controls the output driver strength of an I/O pin configured as an output.
  0: Pin drive strength is set to normal drive strength.
  1: Pin drive strength is set to stronger drive strength.

- **Bits 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – PULLEN: Pull Enable**
  This bit enables the internal pull-up or pull-down resistor of an I/O pin configured as an input.
  0: Internal pull resistor is disabled, and the input is in a high-impedance configuration.
  1: Internal pull resistor is enabled, and the input is driven to a defined logic level in the absence of external input.

- **Bit 1 – INEN: Input Enable**
  This bit controls the input buffer of an I/O pin configured as either an input or output.
  0: Input buffer for the I/O pin is disabled, and the input value will not be sampled.
  1: Input buffer for the I/O pin is enabled, and the input value will be sampled when required.
  Writing a zero to this bit disables the input buffer completely, preventing read-back of the physical pin state when the pin is configured as either an input or output.

- **Bit 0 – PMUXEN: Peripheral Multiplexer Enable**
  This bit enables or disables the peripheral multiplexer selection set in the Peripheral Multiplexing register (PMUXn) to enable or disable alternative peripheral control over an I/O pin direction and output drive value.
  0: The peripheral multiplexer selection is disabled, and the PORT registers control the direction and output drive value.
  1: The peripheral multiplexer selection is enabled, and the selected peripheral controls the direction and output drive value.
  Writing a zero to this bit allows the PORT to control the pad direction via the Data Direction register (DIR) and output drive value via the Data Output Value register (OUT). The peripheral multiplexer value in PMUXn is ignored.
  Writing a one to this bit enables the peripheral selection in PMUXn to control the pad. In this configuration, the physical pin state may still be read from the Data Input Value register (IN) if PINCFGy.INEN is set.

23. EVSYS – Event System

23.1 Overview

The Event System (EVSYS) allows autonomous, low-latency and configurable communication between peripherals.

Several peripherals can be configured to emit and/or respond to signals known as events. The exact condition to generate an event, or the action taken upon receiving an event, is specific to each module. Peripherals that respond to events are called event users. Peripherals that emit events are called event generators. A peripheral can have one or more event generators and can have one or more event users.

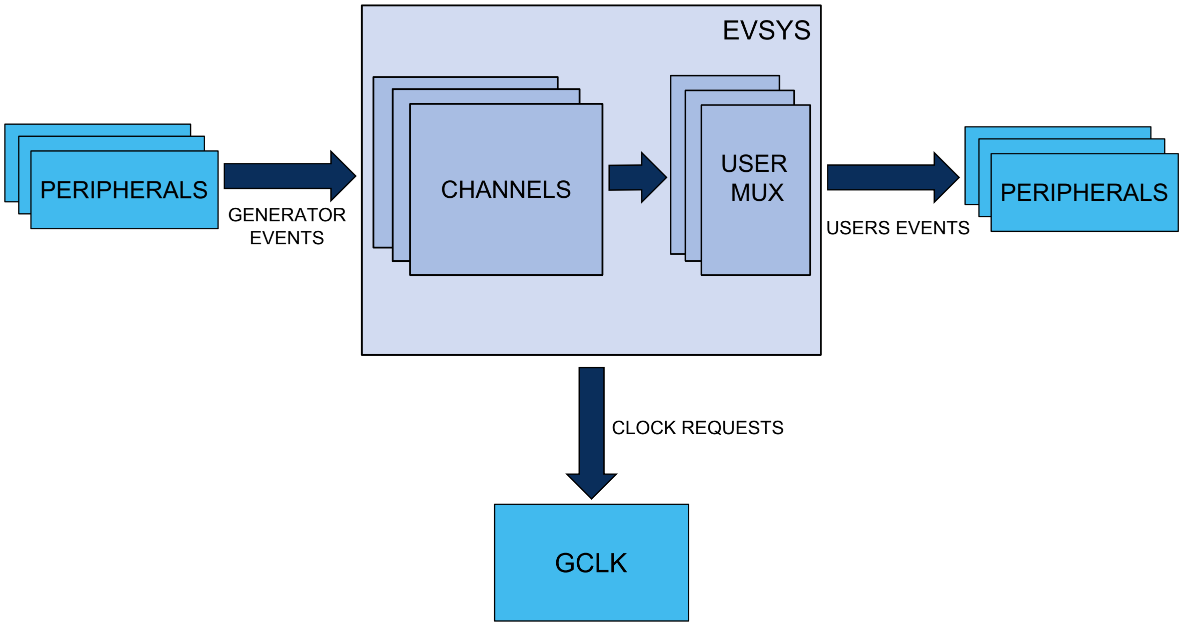
Communication is made without CPU intervention and without consuming system resources such as bus or RAM bandwidth. This reduces the load on the CPU and other system resources, compared to a traditional interrupt-based system.

23.2 Features

- System for direct peripheral-to-peripheral communication and signaling
- 6 configurable event channels, where each channel can:
  - Be connected to any event generator
  - Provide a pure asynchronous, resynchronized or synchronous path
- 18 event generators
- 44 event users
- Configurable edge detector
- Peripherals can be event generators, event users or both
- SleepWalking and interrupt for operation in low-power modes
- Software event generation
- Each event user can choose which event channel to listen to

23.3 Block Diagram

Figure 23-1. Event System Block Diagram



23.4 Signal Description

Not applicable.

23.5 Product Dependencies

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

23.5.1 I/O Lines

Not applicable.

23.5.2 Power Management

The EVSYS can be used to wake up the CPU from all sleep modes, even if the clock used by the EVSYS channel and the EVSYS bus clock are disabled. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

In all sleep modes where the clock for the EVSYS is stopped, the device can wake up the EVSYS clock.

Some event generators can generate an event when the system clock is stopped. The generic clock (GCLK_EVSYS_CHANNELx) for this channel will be restarted if the channel uses a synchronized path or a resynchronized path, without waking the system from sleep. The clock remains active only as long as necessary to handle the event. After the event has been handled, the clock will be turned off and the system will remain in the original sleep mode. This is known as SleepWalking. When an asynchronous path is used, there is no need for the clock to be activated for the event to be propagated to the user.

On a software reset, all registers are set to their reset values and any ongoing events are canceled.

23.5.3 Clocks

The EVSYS bus clock (CLK_EVSYS_APB) can be enabled and disabled in the Power Manager, and the default state of CLK_EVSYS_APB can be found in the Peripheral Clock Masking section in "PM – Power Manager" on page 107.

Each EVSYS channel has a dedicated generic clock (GCLK_EVSYS_CHANNELx). These are used for detection and propagation of events for each channel. These clocks must be configured and enabled in the generic clock controller before using the EVSYS. Refer to "Enabling a Generic Clock" on page 90 for details.

23.5.4 DMA

Not applicable.

23.5.5 Interrupts

The interrupt request line is connected to the interrupt controller. Using the EVSYS interrupts requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

23.5.6 Events

Not applicable.

23.5.7 Debug Operation

When the CPU is halted in debug mode, the EVSYS continues normal operation. If the EVSYS 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.

23.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following register:

- Interrupt Flag Status and Clear register (INTFLAG)

Write-protection is denoted by the Write-Protected property in the register description.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

23.5.9 Analog Connections

Not applicable.

23.6 Functional Description

23.6.1 Principle of Operation

The EVSYS allows for communication between peripherals via events. Peripherals that respond to events (event users) are connected to multiplexers which have all event channels as input. Each each event channel can be configured to route signals from any peripheral emitting events (event generator) to one or more event users.

23.6.2 Basic Operation

23.6.2.1 Initialization

The peripheral that is to act as event generator must be configured to be able to generate events. The peripheral to act as event user must be configured to handle incoming events.

When this has been done, the event system is ready to be configured. The configuration must follow this order:

1. Configure the event user by performing a single 16-bit write to the User Multiplexer register (USER) with:
   1.1. The channel to be connected to a user is written to the Channel bit group (USER.CHANNEL)

1.2. The user to connect the channel is written to the User bit group (USER.USER)

2. Configure the channel by performing a single 32-bit write to the Channel (CHANNEL) register with:
   2.1. The channel to be configured is written to the Channel Selection bit group (CHANNEL.CHANNEL)
   2.2. The path to be used is written to the Path Selection bit group (CHANNEL.PATH)
   2.3. The type of edge detection to use on the channel is written to the Edge Selection bit group (CHANNEL.EDGSEL)
   2.4. The event generator to be used is written to the Event Generator bit group (CHANNEL.EVGEN)

23.6.2.2 Enabling, Disabling and Resetting

The EVSYS is always enabled.

The EVSYS is reset by writing a one to the Software Reset bit in the Control register (CTRL.SWRST). All registers in the EVSYS will be reset to their initial state and any ongoing events will be canceled. Refer to the CTRL register for details.

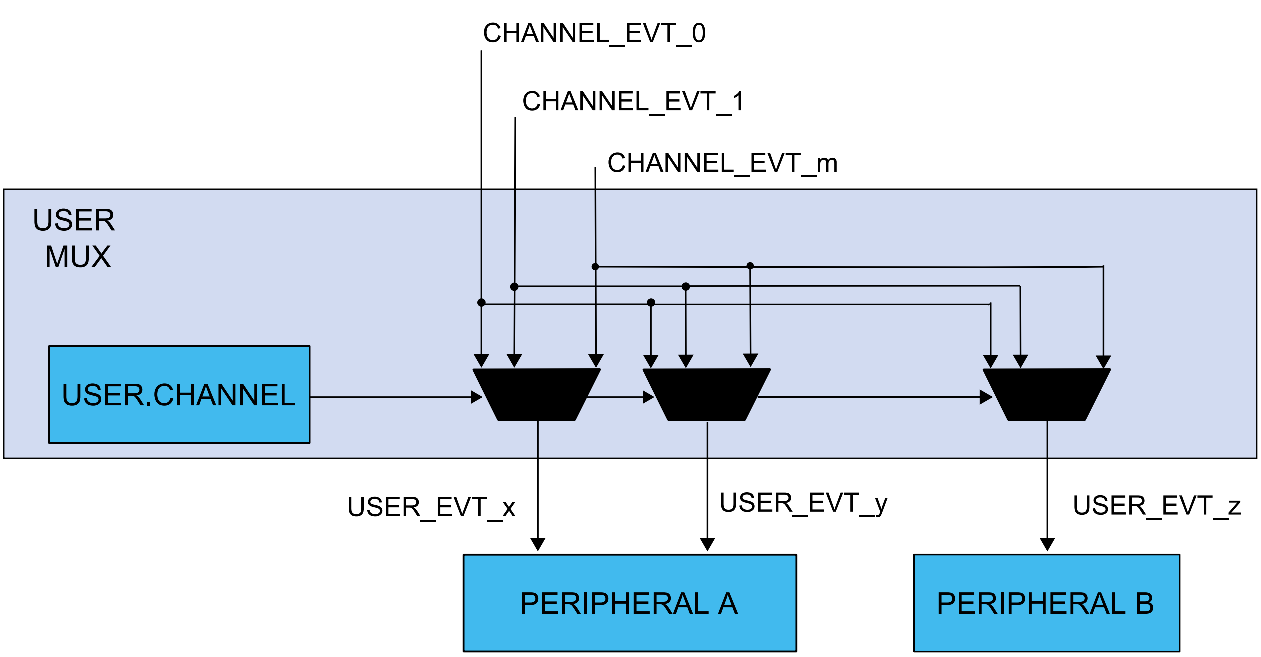
23.6.2.3 User Multiplexer Setup

Each user multiplexer is dedicated to one event user. A user multiplexer receives all event channel outputs and must be configured to select one of these channels. The user must always be configured before the channel is configured. A full list of selectable users can be found in the User Multiplexer register (USER) description. Refer to Table 23-6 for details.

To configure a user multiplexer, the USER register must be written in a single 16-bit write.

It is possible to read out the configuration of a user by first selecting the user by writing to USER.USER using an 8-bit write and then performing a read of the 16-bit USER register.

Figure 23-2. User MUX



23.6.2.4 Channel Setup

The channel to be used with an event user must be configured with an event generator. The path of the channel should be configured, and when using a synchronous path or resynchronized path, the edge selection should be configured. All these configurations are available in the Channel register (CHANNEL).

To configure a channel, the Channel register must be written in a single 32-bit write.

It is possible to read out the configuration of a channel by first selecting the channel by writing to CHANNEL.CHANNEL using a, 8-bit write, and then performing a read of the CHANNEL register.

Event Generators

The event generator is selected by writing to the Event Generator bit group in the Channel register (CHANNEL.EVGEN).

A full list of selectable generators can be found in the CHANNEL register description. Refer to Table 23-4 for details.

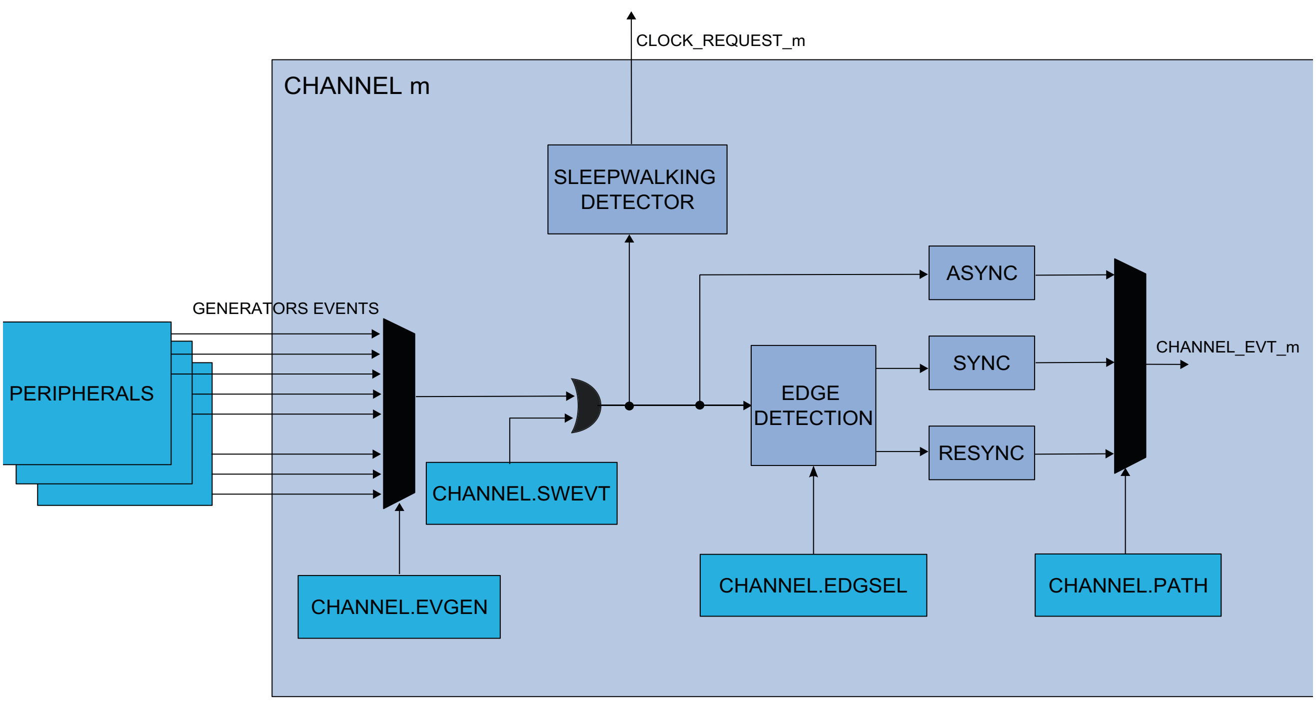
The channels are not connected to any of the event generators (CHANNEL.EVGEN = 0x00) by default.

23.6.2.5 Channel Path

There are three different ways to propagate the event provided by an event generator:

- Asynchronous path
- Synchronous path
- Resynchronized path

Figure 23-3. Channel



The path is selected by writing to the Path Selection bit group in the Channel register (CHANNEL.PATH).

Asynchronous Path

When using the asynchronous path, the events are propagated from the event generator to the event user with no intervention from the event system. This means that if the GCLK_EVSYS_CHANNELx for the channel used is inactive, the event will still be propagated to the user.

Events propagated in the asynchronous path cannot generate any interrupts, and no channel status bits will indicate the state of the channel. No edge detection is available; this must be handled in the event user.

When the event generator and the event user share the same generic clock, using the asynchronous path will propagate the event with the least amount of latency.

Synchronous Path

The synchronous path should be used when the event generator and the event channel share the same generic clock. If they do not share the same clock, a logic change from the event generator to the event channel might not be detected in the channel, which means that the event will not be propagated to the event user.

When using the synchronous path, the channel is capable of generating interrupts. The channel status bits in the Channel Status register (CHSTATUS) are also updated and available for use.

If the Generic Clocks Request bit in the Control register (CTRL.GCLKREQ) is zero, the channel operates in SleepWalking mode and request the configured generic clock only when an event is to be propagated through the channel. If CTRL.GCLKREQ is one, the generic clock will always be on for the configured channel.

Resynchronized Path

The resynchronized path should be used when the event generator and the event channel do not share the same clock. When the resynchronized path is used, resynchronization of the event from the event generator is done in the channel.

When the resynchronized path is used, the channel is capable of generating interrupts. The channel status bits in the Channel Status register (CHSTATUS) are also updated and available for use.

If the Generic Clocks Request bit in the Control register (CTRL.GCLKREQ) is zero, the channel operates in SleepWalking mode and request the configured generic clock only when an event is to be propagated through the channel. If CTRL.GCLKREQ is one, the generic clock will always be on for the configured channel.

23.6.2.6 Edge Detection

When synchronous or resynchronized paths are used, edge detection must be used. The event system can perform edge detection in three different ways:

- Generate an event only on the rising edge
- Generate an event only on the falling edge
- Generate an event on rising and falling edges.

Edge detection is selected by writing to the Edge Selection bit group in the Channel register (CHANNEL.EDGSEL).

If the generator event is a pulse, both edges cannot be selected. Use the rising edge or falling edge detection methods, depending on the generator event default level.

23.6.2.7 Channel Status

The Channel Status register (CHSTATUS) contains the status of the channels when a synchronous or resynchronized path is in use. There are two different status bits in CHSTATUS for each of the available channels: The Channel Busy bit (CHSTATUS.CHBUSYx) is set to one if an event on the corresponding channel x has not been handled by all event users connected to that channel.

The CHSTATUS.USRRDYx bit is set to one if all event users connected to the corresponding channel x are ready to handle incoming events on that channel.

23.6.2.8 Software Event

A software event can be initiated on a channel by writing a one to the Software Event bit in the Channel register (CHANNEL.SWEVT) at the same time as writing the Channel bits (CHANNEL.CHANNEL). This will generate a software event on the selected channel.

The software event can be used for application debugging, and functions like any event generator. To use the software event, the event path must be configured to either a synchronous path or resynchronized path

(CHANNEL.PATH = 0x0 or 0x1), edge detection must be configured to rising-edge detection (CHANNEL.EDGSEL= 0x1) and the Generic Clock Request bit must be set to one (CTRL.GCLKREQ=0x1).

23.6.3 Interrupts

The EVSYS has the following interrupt sources:

- Overrun Channel x interrupt (INTFLAG)
- Event Detected Channel x interrupt (INTFLAG)

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the EVSYS is reset.

See the INTFLAG register for details on how to clear interrupt flags. The EVSYS has one common interrupt request line for all the interrupt sources. The user must read the INTFLAG register to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated.

Refer to "Nested Vector Interrupt Controller" on page 23 for details.

23.6.3.1 The Overrun Channel x Interrupt

The Overrun Channel x interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVRx) is set and the optional interrupt is generated in the following two cases:

- At least one of the event users on channel x is not ready when a new event occurs
- An event occurs when the previous event on channel x has not yet been handled by all event users

INTFLAG.OVRx will be set when using a synchronous or resynchronized path, but not when using an asynchronous path.

23.6.3.2 The Event Detected Channel x Interrupt

The Event Detected Channel x interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.EVDx) is set when an event coming from the event generator configured on channel x is detected.

INTFLAG.EVDx will be set when using a synchronous and resynchronized path, but not when using an asynchronous path.

23.6.4 Sleep Mode Operation

The EVSYS can generate interrupts to wake up the device from any sleep mode.

23.7 Register Summary

Table 23-1. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0				GCLKREQ				SWRST
0x01 ... 0x03	Reserved									
0x04	CHANNEL	7:0						CHANNEL[2:0]		
0x05		15:8								SWEVT
0x06		23:16			EVGEN[5:0]					
0x07		31:24					EDGSEL[1:0]		PATH[1:0]	
0x08	USER	7:0				USER[4:0]				
0x09		15:8					CHANNEL[3:0]			
0x0A	Reserved									
0x0B	Reserved									
0x0C	CHSTATUS	7:0			USRRDY5	USRRDY4	USRRDY3	USRRDY2	USRRDY1	USRRDY0
0x0D		15:8			CHBUSY5	CHBUSY4	CHBUSY3	CHBUSY2	CHBUSY1	CHBUSY0
0x0E		23:16								
0x0F		31:24								
0x10	INTENCLR	7:0			OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
0x11		15:8			EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
0x12		23:16								
0x13		31:24								
0x14	INTENSET	7:0			OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
0x15		15:8			EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
0x16		23:16								
0x17		31:24								
0x18	INTFLAG	7:0			OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
0x19		15:8			EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
0x1A		23:16								
0x1B		31:24								

23.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to "Register Access Protection" on page 404 and "PAC – Peripheral Access Controller" on page 27 for details.

23.8.1 Control

Name: CTRL

Offset: 0x00

Reset: 0x00

Access: Write-Only

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
				GCLKREQ				SWRST
Access	R	R	R	R/W	R	R	R	W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 4 – GCLKREQ: Generic Clock Requests**
  This bit is used to determine whether the generic clocks used for the different channels should be on all the time or only when an event needs the generic clock. Events propagated through asynchronous paths will not need a generic clock.

  0: Generic clock is requested and turned on only if an event is detected.

  1: Generic clock for a channel is always on.

- **Bits 3:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – SWRST: Software Reset**
  Writing a zero to this bit has no effect.

  Writing a one to this bit resets all registers in the EVSYS to their initial state.

  Writing a one to CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.

23.8.2 Channel

Name: CHANNEL

Offset: 0x04

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
					EDGSEL[1:0]		PATH[1:0]	
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			EVGEN[5:0]					
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
								SWEVT
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						CHANNEL[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to configure the channel specified in the CHANNEL bit group. To write to this register, do a single 32-bit write of all the configuration and channel selection data.

To read from this register, first do an 8-bit write to the CHANNEL.CHANNEL bit group specifying the channel configuration to be read, and then read the Channel register (CHANNEL).

- **Bits 31:28 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 27:26 – EDGSEL[1:0]: Edge Detection Selection**
  These bits set the type of edge detection to be used on the channel.

  These bits must be written to zero when using the asynchronous path.

Table 23-2. Edge Detection Selection

EDGSEL[1:0]	Name	Description
0x0	NO_EVT_OUTPUT	No event output when using the resynchronized or synchronous path
0x1	RISING_EDGE	Event detection only on the rising edge of the signal from the event generator when using the resynchronized or synchronous path
0x2	FALLING_EDGE	Event detection only on the falling edge of the signal from the event generator when using the resynchronized or synchronous path
0x3	BOTH_EDGES	Event detection on rising and falling edges of the signal from the event generator when using the resynchronized or synchronous path

- **Bits 25:24 – PATH[1:0]: Path Selection**
  These bits are used to choose the path to be used by the selected channel.
  The path choice can be limited by the channel source, see Table 23-6.

Table 23-3. Path Selection

PATH[1:0]	Name	Description
0x0	SYNCHRONOUS	Synchronous path
0x1	RESYNCHRONIZED	Resynchronized path
0x2	ASYNCHRONOUS	Asynchronous path
0x3		Reserved

- **Bits 23:22 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 21:16 – EVGEN[5:0]: Event Generator Selection**
  These bits are used to choose which event generator to connect to the selected channel.

Table 23-4. **Event Generator Selection**

Value	Event Generator	Description
0x00	NONE	No event generator selected
0x01	RTC CMP0	Compare 0 (mode 0 and 1) or Alarm 0 (mode 2)
0x02	RTC CMP1	Compare 1
0x03	RTC OVF	Overflow
0x04	RTC PER0	Period 0
0x05	RTC PER1	Period 1
0x06	RTC PER2	Period 2
0x07	RTC PER3	Period 3
0x08	RTC PER4	Period 4
0x09	RTC PER5	Period 5
0x0A	RTC PER6	Period 6
0x0B	RTC PER7	Period 7
0x0C	EIC EXTINT0	External Interrupt 0
0x0D	EIC EXTINT1	External Interrupt 1
0x0E	EIC EXTINT2	External Interrupt 2
0x0F	EIC EXTINT3	External Interrupt 3
0x10	EIC EXTINT4	External Interrupt 4
0x11	EIC EXTINT5	External Interrupt 5
0x12	EIC EXTINT6	External Interrupt 6
0x13	EIC EXTINT7	External Interrupt 7
0x14	DMAC CH0	Channel 0
0x15	DMAC CH1	Channel 1
0x16	DMAC CH2	Channel 2
0x17	DMAC CH3	Channel 3

Table 23-4. Event Generator Selection (Continued)

Value	Event Generator	Description
0x18-0x1E	Reserved	
0x1F	TC1 OVF	Overflow/Underflow
0x20	TC1 MC0	Match/Capture 0
0x21	TC1 MC1	Match/Capture 1
0x22	TC2 OVF	Overflow/Underflow
0x23	TC2 MC0	Match/Capture 0
0x24	TC2 MC1	Match/Capture 1
0x25	ADC RESRDY	Result Ready
0x26	ADC WINMON	Window Monitor
0x26-0x7F	Reserved	

- **Bits 15:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 8 – SWEVT: Software Event**
  This bit is used to insert a software event on the channel selected by the CHANNEL.CHANNEL bit group.

  This bit must be written together with CHANNEL.CHANNELusing a 16-bit write.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will trigger a software event for the corresponding channel.

  This bit will always return zero when read.

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – CHANNEL[2:0]: Channel Selection**
  These bits are used to select the channel to be set up or read from.

23.8.3 User Multiplexer

Name: USER

Offset: 0x08

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
					CHANNEL[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				USER[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register is used to configure a specified event user. To write to this register, do a single 16-bit write of all the configuration and event user selection data.

To read from this register, first do an 8-bit write to the USER.USER bit group specifying the event user configuration to be read, and then read USER.

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:8 – CHANNEL[3:0]: Channel Event Selection**
  These bits are used to select the channel to connect to the event user.
  Note that to select channel n, the value (n+1) must be written to the USER.CHANNEL bit group.

Table 23-5. Channel Event Selection

CHANNEL[3:0]	Channel Number
0x0	No Channel Output Selected
0x1-0x5	Channel n-1 selected
0x2-0xff	Reserved

- **Bits 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:0 – USER[4:0]: User Multiplexer Selection**
  These bits select the event user to be configured with a channel, or the event user to read the channel value from.

Table 23-6. User Multiplexer Selection

USER[7:0]	User Multiplexer	Description	Path Type
0x00	DMAC CH0	Channel 0	Resynchronized path only
0x01	DMAC CH1	Channel 1	Resynchronized path only
0x02	DMAC CH2	Channel 2	Resynchronized path only
0x03	DMAC CH3	Channel 3	Resynchronized path only
0x04-0x09	Reserved		Reserved
0x0A	TC1_EVU		Asynchronous, synchronous and resynchronized paths
0x0B	TC2_EVU		Asynchronous, synchronous and resynchronized paths
0x0C	ADC_START		Asynchronous path only
0x0D	ADC_SYNC		Asynchronous path only
0x0E-0x1F	Reserved		Reserved

23.8.4 Channel Status

Name: CHSTATUS

Offset: 0x0C

Reset: 0x0000003F

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			CHBUSY5	CHBUSY4	CHBUSY3	CHBUSY2	CHBUSY1	CHBUSY0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			USRRDY5	USRRDY4	USRRDY3	USRRDY2	USRRDY1	USRRDY0
Access	R	R	R	R	R	R	R	R
Reset	0	0	1	1	1	1	1	1

- **Bits 31:14 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 13:8 – CHBUSYx [x=5..0]: Channel x Busy**
  This bit is cleared when channel x is idle

  This bit is set if an event on channel x has not been handled by all event users connected to channel x.

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – USRRDYx [x=5..0]: Channel x User Ready**
  This bit is cleared when at least one of the event users connected to the channel is not ready.

  This bit is set when all event users connected to channel x are ready to handle incoming events on channel x.

23.8.5 Interrupt Enable Clear

Name: INTENCLR

Offset: 0x10

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

- **Bits 31:14 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 13:8 – EVDx [x=5..0]: Channel x Event Detection Interrupt Enable**
  0: The Event Detected Channel x interrupt is disabled.

  1: The Event Detected Channel x interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Event Detected Channel x Interrupt Enable bit, which disables the Event Detected Channel x interrupt.

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – OVRx [x=5..0]: Channel x Overrun Interrupt Enable**

  0: The Overrun Channel x interrupt is disabled.

  1: The Overrun Channel x interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Overrun Channel x Interrupt Enable bit, which disables the Overrun Channel x interrupt.

23.8.6 Interrupt Enable Set

Name: INTENSET

Offset: 0x14

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

- **Bits 31:14 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 13:8 – EVDx [x=5..0]: Channel x Event Detection Interrupt Enable**
  0: The Event Detected Channel x interrupt is disabled.

  1: The Event Detected Channel x interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the Event Detected Channel x Interrupt Enable bit, which enables the Event Detected Channel x interrupt.

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – OVRx [x=5..0]: Channel x Overrun Interrupt Enable**

  0: The Overrun Channel x interrupt is disabled.

  1: The Overrun Channel x interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the Overrun Channel x Interrupt Enable bit, which enables the Overrun Channel x interrupt.

23.8.7 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x18

Reset: 0x00000000

Access: Read-Write

Property: -

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:14 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 13:8 – EVDx [x=5..0]: Channel x Event Detection**
  This flag is set on the next CLK_EVSYS_APB cycle when an event is being propagated through the channel, and an interrupt request will be generated if INTENCLR/SET.EVDx is one.

  When the event channel path is asynchronous, the EVDx interrupt flag will not be set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Event Detected Channel n interrupt flag.

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – OVRx [x=5..0]: Channel x Overrun**
  This flag is set on the next CLK_EVSYS cycle after an overrun channel condition occurs, and an interrupt request will be generated if INTENCLR/SET.OVRx is one.

When the event channel path is asynchronous, the OVRx interrupt flag will not be set.

Writing a zero to this bit has no effect.

Writing a one to this bit will clear the Overrun Channel x interrupt flag.

24. SERCOM – Serial Communication Interface

24.1 Overview

The serial communication interface (SERCOM) can be configured to support a number of modes; I^2C, SPI, and USART. Once configured and enabled, all SERCOM resources are dedicated to the selected mode.

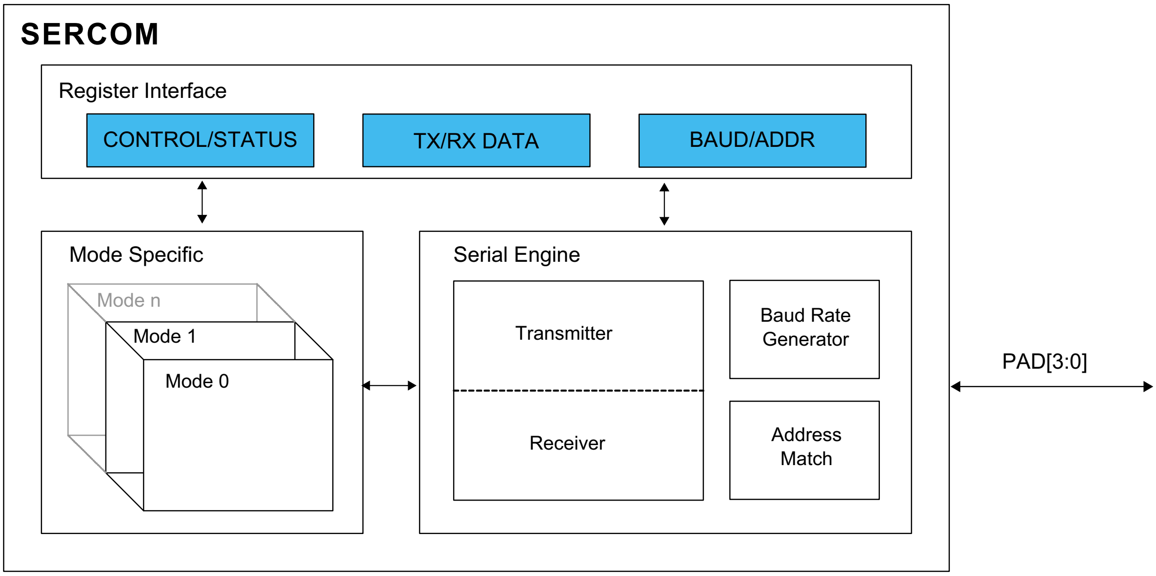
The SERCOM serial engine consists of a transmitter and receiver, baud-rate generator and address matching functionality. It can be configured to use the internal generic clock or an external clock, making operation in all sleep modes possible.

24.2 Features

- Combined interface configurable as one of the following:
  - I^2C – Two-wire serial interface
    - SMBus™ compatible.
  - SPI – Serial peripheral interface
  - USART – Universal synchronous and asynchronous serial receiver and transmitter
- Single transmit buffer and double receive buffer
- Baud-rate generator
- Address match/mask logic
- Operational in all sleep modes
- Can be used with DMA

24.3 Block Diagram

Figure 24-1. SERCOM Block Diagram



24.4 Signal Description

See the respective SERCOM mode chapters for details:

- "SERCOM USART – SERCOM Universal Synchronous and Asynchronous Receiver and Transmitter" on page 435
- "SERCOM SPI – SERCOM Serial Peripheral Interface" on page 473
- "SERCOM I2C – SERCOM Inter-Integrated Circuit" on page 506

24.5 Product Dependencies

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

24.5.1 I/O Lines

Using the SERCOM I/O lines requires the I/O pins to be configured using port configuration (PORT). Refer to "PORT" on page 375 for details.

From Figure 24-1 one can see that the SERCOM has four internal pads, PAD[3:0]. The signals from I^2C, SPI and USART are routed through these SERCOM pads via a multiplexer. The configuration of the multiplexer is available from the different SERCOM modes. Refer to the mode specific chapters for details:

- "SERCOM USART – SERCOM Universal Synchronous and Asynchronous Receiver and Transmitter" on page 435
- "SERCOM SPI – SERCOM Serial Peripheral Interface" on page 473
- "SERCOM I2C – SERCOM Inter-Integrated Circuit" on page 506

24.5.2 Power Management

The SERCOM can operate in any sleep mode.SERCOM interrupts can be used to wake up the device from sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

24.5.3 Clocks

The SERCOM bus clock (CLK_SERCOMx_APB) is enabled by default, and can be enabled and disabled in the Power Manager. Refer to "PM – Power Manager" on page 107 for details.

Two generic clocks are used by the SERCOM: GCLK_SERCOMx_CORE and GCLK_SERCOMx_SLOW. The core clock (GCLK_SERCOMx_CORE) is required to clock the SERCOM while operating as a master, while the slow clock (GCLK_SERCOMx_SLOW) is only required for certain functions. See specific mode chapters for details.

These clocks must be configured and enabled in the Generic Clock Controller (GCLK) before using the SERCOM. Refer to "GCLK – Generic Clock Controller" on page 85 for details.

These generic clocks are asynchronous to the user interface clock (CLK_SERCOMx_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 434 for further details.

24.5.4 DMA

The DMA request lines are connected to the DMA controller (DMAC). Using the SERCOM DMA requests, requires the DMA controller to be configured first. Refer to "DMAC – Direct Memory Access Controller" on page 268 for details.

24.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the SERCOM interrupts requires the Interrupt Controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

24.5.6 Events

Not applicable.

24.5.7 Debug Operation

When the CPU is halted in debug mode, the SERCOM continues normal operation. If the SERCOM is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging. The SERCOM can be forced to halt operation during debugging.

24.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Interrupt Flag Status and Clear register (INTFLAG)
- Address register (ADDR)
- Data register (DATA)

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode, all write-protection is automatically disabled. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

24.5.9 Analog Connections

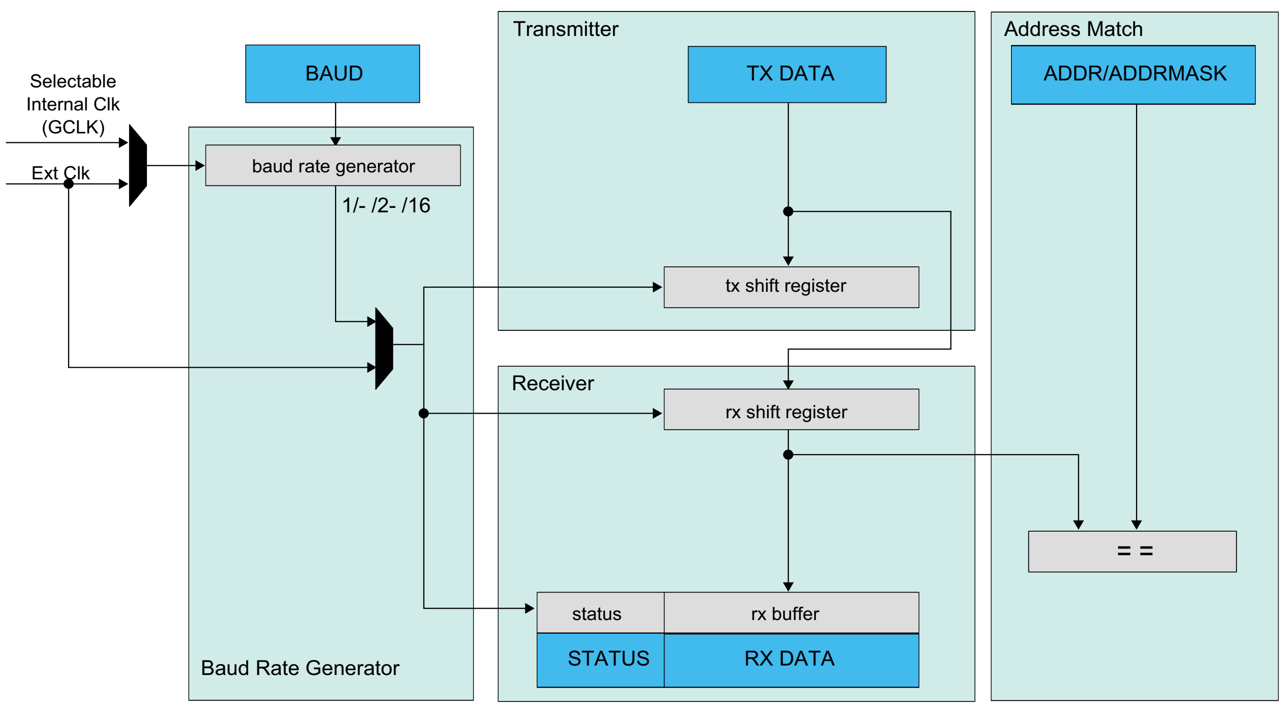
Not applicable.

24.6 Functional Description

24.6.1 Principle of Operation

The basic structure of the SERCOM serial engine is shown in Figure 24-2. Fields shown in capital letters are synchronous to the system clock and accessible by the CPU, while fields with lowercase letters can be configured to run on the GCLK_SERCOMx_CORE clock or an external clock.

Figure 24-2. SERCOM Serial Engine



The transmitter consists of a single write buffer and a shift register. The receiver consists of a two-level receive buffer and a shift register. The baud-rate generator is capable of running on the GCLK_SERCOMx_CORE clock or an external clock. Address matching logic is included for SPI and I²C operation.

24.6.2 Basic Operation

24.6.2.1 Initialization

The SERCOM must be configured to the desired mode by writing to the Operating Mode bits in the Control A register (CTRLA.MODE). Refer to Figure 24-1 for details.

Table 24-1. SERCOM Modes

CTRLA.MODE	Description
0x0	USART with external clock
0x1	USART with internal clock
0x2	SPI in slave operation
0x3	SPI in master operation
0x4	I^2C slave operation
0x5	I^2C master operation
0x6-0x7	Reserved

For further initialization information, see the respective SERCOM mode chapters.

24.6.2.2 Enabling, Disabling and Resetting

The SERCOM is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The SERCOM is disabled by writing a zero to CTRLA.ENABLE.

The SERCOM is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the SERCOM, except DBGCTRL, will be reset to their initial state, and the SERCOM will be disabled. Refer to the CTRLA register descriptions for details.

24.6.2.3 Clock Generation – Baud-Rate Generator

The baud-rate generator, as shown in Figure 24-3, is used for internal clock generation for asynchronous and synchronous communication. The generated output frequency (f_{BAUD}) is determined by the Baud register (BAUD) setting and the baud reference frequency (f_{REF}). The baud reference clock is the serial engine clock, and it can be internal or external.

For asynchronous operation, the /16 (divide-by-16) output is used when transmitting and the /1 (divide-by-1) output is used when receiving. For synchronous operation the /2 (divide-by-2) output is used. This functionality is automatically configured, depending on the selected operating mode.

Figure 24-3. Baud Rate Generator

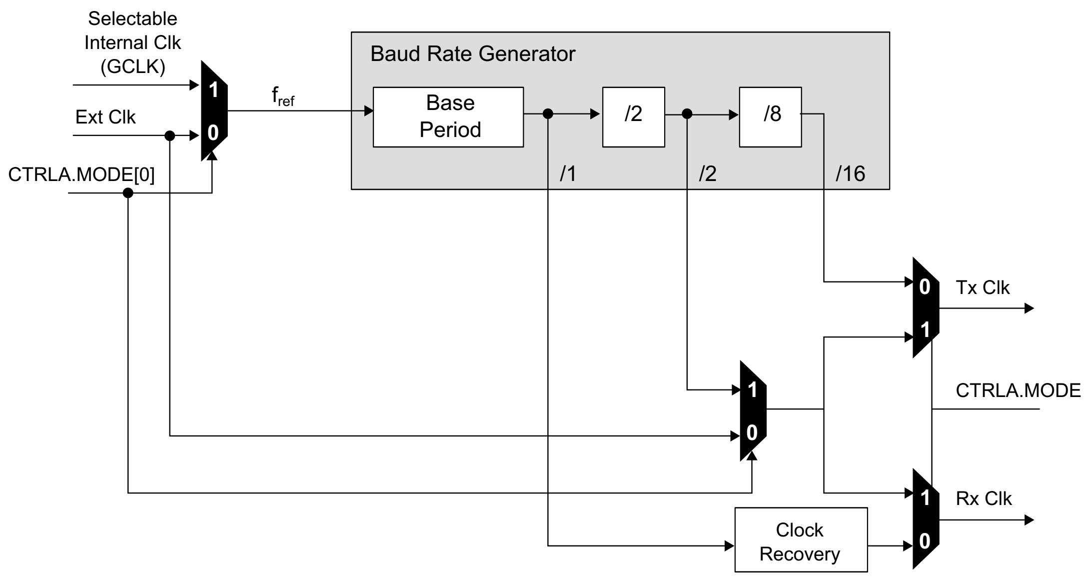


Table 24-2 contains equations for calculating the baud rate (in bits per second) and for calculating the BAUD register value for each mode of operation.

For asynchronous operation there are two different modes. Using the arithmetic mode, the BAUD register value is 16 bits (0 to 65,535). Using the fractional mode, the BAUD register is 13 bits, while the fractional adjustment is 3 bits. In this mode the BAUD setting must be greater than or equal to 1.

For synchronous mode, the BAUD register value is 8 bits (0 to 255).

Table 24-2. Baud Rate Equations

Operating Mode	Condition	Baud Rate (Bits Per Second)	BAUD Register Value Calculation
Asynchronous Arithmetic	$f_{BAUD} \le \frac{f_{REF}}{S}$	$f_{BAUD} = \frac{f_{REF}}{S}(1 - BAUD/65{,}536)$	$BAUD = 65{,}536\left(1 - S\frac{f_{BAUD}}{f_{REF}}\right)$
Asynchronous Fractional	$f_{BAUD} \le \frac{f_{REF}}{S}$	$f_{BAUD} = \frac{f_{REF}}{S(BAUD + (FP/8))}$	$BAUD = \frac{f_{REF}}{S \times f_{BAUD}} - \frac{FP}{8}$
Synchronous	$f_{BAUD} \le \frac{f_{REF}}{2}$	$f_{BAUD} = \frac{f_{REF}}{2(BAUD + 1)}$	$BAUD = \frac{f_{REF}}{2 f_{BAUD}} - 1$

S – Number of samples per bit. Can be 16, 8, or 3.

The Asynchronous Fractional option is used for auto-baud detection.

The baud rate error is represented by the following formula:

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

Asynchronous Arithmetic Mode BAUD Value Selection

The formula given for f_{BAUD} calculates the average frequency over 65,536 f_{REF} cycles. Although the BAUD register can be set to any value between 0 and 65,536, the values that will change the average frequency of f_{BAUD} over a single frame are more constrained. The BAUD register values that will affect the average frequency over a single frame lead to an integer increase in the cycles per frame (CPF).

$$CPF = \frac{f_{REF}}{f_{BAUD}}(D + S)$$

where

- D represent the data bits per frame
- S represent the sum of start and first stop bits, if present

Table 24-3 shows the BAUD register value versus baud frequency at a serial engine frequency of 48MHz. This assumes a D value of 8 bits and an S value of 2 bits (10 bits, including start and stop bits).

Table 24-3. BAUD Register Value vs. Baud Frequency

BAUD Register Value	Serial Engine CPF	f_{BAUD} at 48MHz Serial Engine Frequency (f_{REF})
0 – 406	160	3MHz
407 – 808	161	2.981MHz
809 – 1205	162	2.963MHz
...		
65206	31775	15.11kHz
65207	31871	15.06kHz
65208	31969	15.01kHz

24.6.3 Additional Features

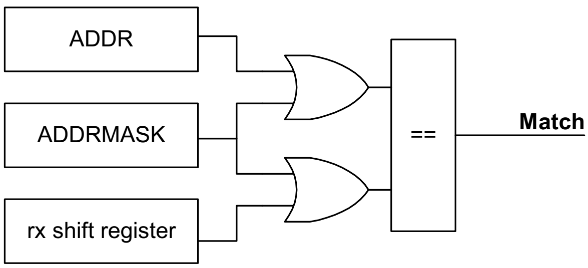
24.6.3.1 Address Match and Mask

The SERCOM address match and mask feature is capable of matching one address with a mask, two unique addresses or a range of addresses, based on the mode selected. The match uses seven or eight bits, depending on the mode.

Address With Mask

An address written to the Address bits in the Address register (ADDR.ADDR) with a mask written to the Address Mask bits in the Address register (ADDR.ADDRMASK) will yield an address match. All bits that are masked are not included in the match. Note that setting the ADDR.ADDRMASK to all zeros will match a single unique address, while setting ADDR.ADDRMASK to all ones will result in all addresses being accepted.

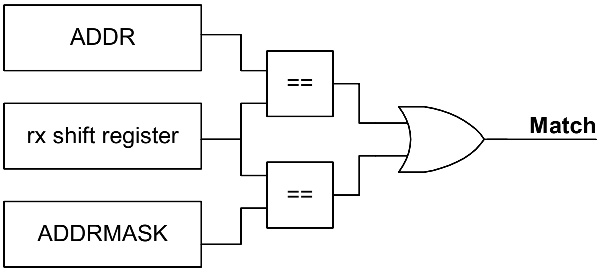
Figure 24-4. Address With Mask



Two Unique Addresses

The two addresses written to ADDR and ADDRMASK will cause a match.

Figure 24-5. Two Unique Addresses



Address Range

The range of addresses between and including ADDR.ADDR and ADDR.ADDRMASK will cause a match. ADDR.ADDR and ADDR.ADDRMASK can be set to any two addresses, with ADDR.ADDR acting as the upper limit and ADDR.ADDRMASK acting as the lower limit.

Figure 24-6. Address Range

24.6.4 DMA Operation

Not applicable.

24.6.5 Interrupts

Interrupt sources are mode-specific. See the respective SERCOM mode chapters for details.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the SERCOM is reset. See the register description for details on how to clear interrupt flags.

The SERCOM has one common interrupt request line for all the interrupt sources. The user must read the INTFLAG register to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

24.6.6 Events

Not applicable.

24.6.7 Sleep Mode Operation

The peripheral can operate in any sleep mode where the selected serial clock is running. This clock can be external or generated by the internal baud-rate generator.

The SERCOM interrupts can be used to wake up the device from sleep modes. Refer to the different SERCOM mode chapters for details.

24.6.8 Synchronization

Due to the asynchronicity between CLK_SERCOMx_APB and GCLK_SERCOMx_CORE, some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set immediately, and cleared when synchronization is complete. The Synchronization Ready interrupt can be used to signal when synchronization is complete.

If an operation that requires synchronization is executed while STATUS.SYNCBUSY is one, the bus will be stalled. All operations will complete successfully, but the CPU will be stalled and interrupts will be pending as long as the bus is stalled.

25. SERCOM USART – SERCOM Universal Synchronous and Asynchronous Receiver and Transmitter

25.1 Overview

The universal synchronous and asynchronous receiver and transmitter (USART) is one of the available modes in the Serial Communication Interface (SERCOM).

Refer to "SERCOM – Serial Communication Interface" on page 427 for details.

The USART uses the SERCOM transmitter and receiver configured as shown in Figure 25-1. Fields shown in capital letters are synchronous to the CLK_SERCOMx_APB and accessible by the CPU, while fields with lowercase letters can be configured to run on the internal generic clock or an external clock.

The transmitter consists of a single write buffer, a shift register and control logic for handling different frame formats. The write buffer allows continuous data transmission without any delay between frames.

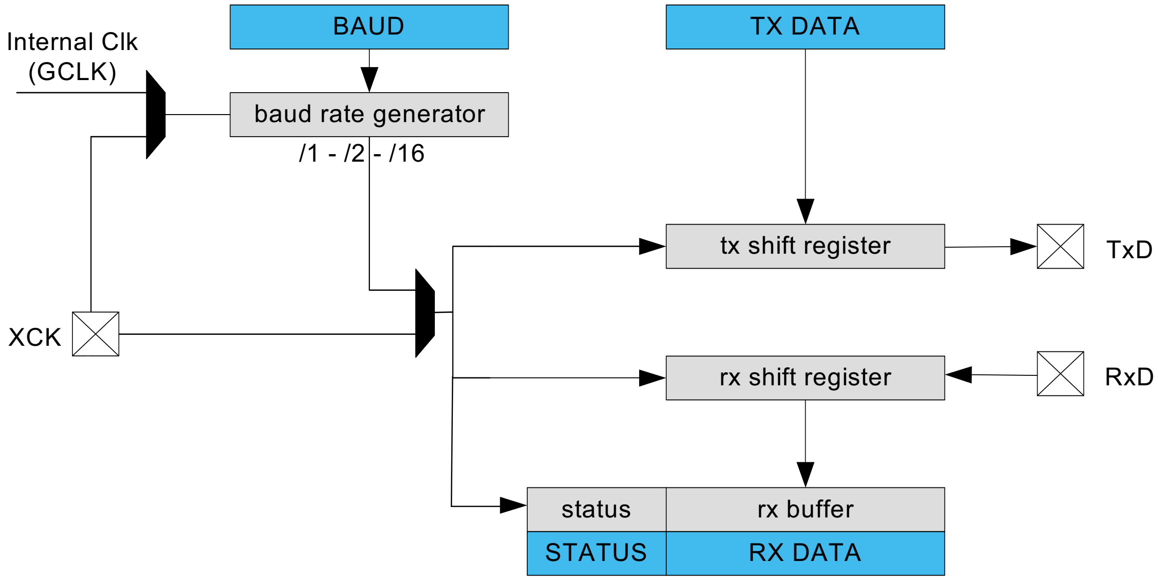
The receiver consists of a two-level receive buffer and a shift register. Status information for the received data is available for error checking. Data and clock recovery units ensure robust synchronization and noise filtering during asynchronous data reception.

25.2 Features

- Full-duplex operation
- Asynchronous (with clock reconstruction) or synchronous operation
- Internal or external clock source for asynchronous and synchronous operation
- Baud-rate generator
- Supports serial frames with 5, 6, 7, 8 or 9 data bits and 1 or 2 stop bits
- Odd or even parity generation and parity check
- Selectable LSB- or MSB-first data transfer
- Buffer overflow and frame error detection
- Noise filtering, including false start-bit detection and digital low-pass filter
- Collision detection
- Can operate in all sleep modes
- Operation at speeds up to half the system clock for internally generated clocks
- Operation at speeds up to the system clock for externally generated clocks
- RTS and CTS flow control
- IrDA modulation and demodulation up to 115.2 kbps
- LIN slave support
  - Auto-baud and break character detection
- Start-of-frame detection
- Can be used with DMA

25.3 Block Diagram

Figure 25-1. USART Block Diagram



25.4 Signal Description

Signal Name	Type	Description
PAD[3:0]	Digital I/O	General SERCOM pins

Please refer to “I/O Multiplexing and Considerations” on page 10 for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

25.5 Product Dependencies

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

25.5.1 I/O Lines

Using the USART’s I/O lines requires the I/O pins to be configured using port configuration (PORT).

Refer to “PORT” on page 375 for details.

When the SERCOM is used in USART mode, the pins should be configured according to Table 25-1. If the receiver or transmitter is disabled, these pins can be used for other purposes.

Table 25-1. USART Pin Configuration

Pin	Pin Configuration
TxD	Output
RxD	Input
XCK	Output or input

The combined configuration of PORT and the Transmit Data Pinout and Receive Data Pinout bit groups (refer to the Control A register description) will define the physical position of the USART signals in Table 25-1.

25.5.2 Power Management

The USART can continue to operate in any sleep mode where the selected source clock is running. The USART interrupts can be used to wake up the device from sleep modes. The events can trigger other operations in the system without exiting sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

25.5.3 Clocks

The SERCOM bus clock (CLK_SERCOMx_APB, where x represents the specific SERCOM instance number) can be enabled and disabled in the Power Manager, and the default state of CLK_SERCOMx_APB can be found in the Peripheral Clock Masking section in "PM – Power Manager" on page 107.

A generic clock (GCLK_SERCOMx_CORE) is required to clock the SERCOMx_CORE. This clock must be configured and enabled in the Generic Clock Controller before using the SERCOMx_CORE. Refer to "GCLK – Generic Clock Controller" on page 85 for details.

This generic clock is asynchronous to the bus clock (CLK_SERCOMx_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 448 for further details.

25.5.4 DMA

The DMA request lines are connected to the DMA controller (DMAC). Using the SERCOM DMA requests, requires the DMA controller to be configured first. Refer to "DMAC – Direct Memory Access Controller" on page 268 for details..

25.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the USART interrupts requires the Interrupt Controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

25.5.6 Events

Not applicable.

25.5.7 Debug Operation

When the CPU is halted in debug mode, the USART continues normal operation. If the USART is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging. The USART can be forced to halt operation during debugging.

Refer to DBGCTRL for details.

25.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Interrupt Flag Status and Clear register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode, all write-protection is automatically disabled.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

25.5.9 Analog Connections

Not applicable.

25.6 Functional Description

25.6.1 Principle of Operation

The USART uses three communication lines for data transfer:

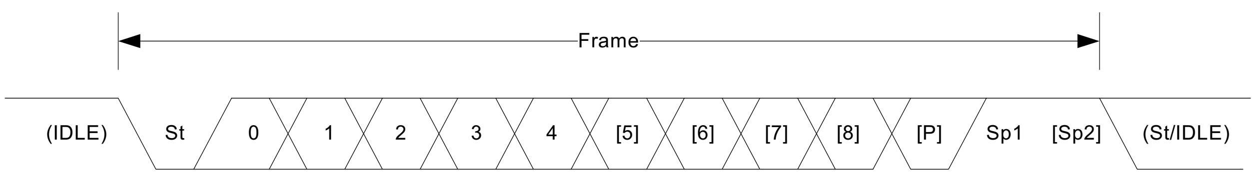
- RxD for receiving
- TxD for transmitting
- XCK for the transmission clock in synchronous operation

USART data transfer is frame based, where a serial frame consists of:

- 1 start bit
- 5, 6, 7, 8 or 9 data bits
- MSB or LSB first
- No, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by one character of data bits. If enabled, the parity bit is inserted after the data bits and before the first stop bit. One frame can be directly followed by a new frame, or the communication line can return to the idle (high) state. Figure 25-2 illustrates the possible frame formats. Bits inside brackets are optional.

Figure 25-2. Frame Formats



St Start bit; always low

(n) Data bits; 0 to 8

P Parity bit; odd or even

Sp Stop bit; always high

IDLE No transfers on the communication line; always high in this state

25.6.2 Basic Operation

25.6.2.1 Initialization

The following registers are enable-protected, meaning they can only be written when the USART is disabled (CTRL.ENABLE is zero):

- Control A register (CTRLA), except the Enable (ENABLE) and Software Reset (SWRST) bits
- Control B register (CTRLB), except the Receiver Enable (RXEN) and Transmitter Enable (TXEN) bits
- Baud register (BAUD)

Any writes to these registers when the USART is enabled or is being enabled (CTRL.ENABLE is one) will be discarded. Writes to these registers) while the peripheral is being disabled will be completed after the disabling is complete.

Before the USART is enabled, it must be configured, as outlined in the following steps:

- USART mode with external or internal clock must be selected first by writing 0x0 or 0x1 to the Operating Mode bit group in the Control A register (CTRLA.MODE)
- Communication mode (asynchronous or synchronous) must be selected by writing to the Communication Mode bit in the Control A register (CTRLA.CMODE)
- SERCOM pad to use for the receiver must be selected by writing to the Receive Data Pinout bit group in the Control A register (CTRLA.RXPO)
- SERCOM pads to use for the transmitter and external clock must be selected by writing to the Transmit Data Pinout bit in the Control A register (CTRLA.TXPO)
- Character size must be selected by writing to the Character Size bit group in the Control B register (CTRLB.CHSIZE)
- MSB- or LSB-first data transmission must be selected by writing to the Data Order bit in the Control A register (CTRLA.DORD)
- When parity mode is to be used, even or odd parity must be selected by writing to the Parity Mode bit in the Control B register (CTRLB.PMODE) and enabled by writing 0x1 to the Frame Format bit group in the Control A register (CTRLA.FORM)
- Number of stop bits must be selected by writing to the Stop Bit Mode bit in the Control B register (CTRLB.SBMODE)
- When using an internal clock, the Baud register (BAUD) must be written to generate the desired baud rate
- The transmitter and receiver can be enabled by writing ones to the Receiver Enable and Transmitter Enable bits in the Control B register (CTRLB.RXEN and CTRLB.TXEN)

25.6.2.2 Enabling, Disabling and Resetting

The USART is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The USART is disabled by writing a zero to CTRLA.ENABLE.

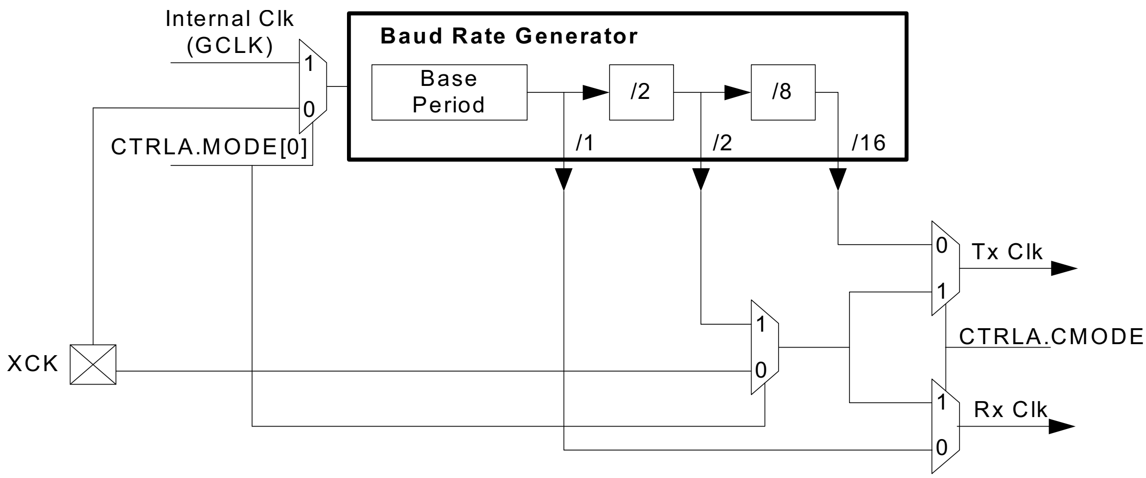
The USART is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the USART, except DBGCTRL, will be reset to their initial state, and the USART will be disabled. Refer to the CTRLA register for details.

25.6.2.3 Clock Generation and Selection

For both synchronous and asynchronous modes, the clock used for shifting and sampling data can be generated internally by the SERCOM baud-rate generator or supplied externally through the XCK line. Synchronous mode is selected by writing a one to the Communication Mode bit in the Control A register (CTRLA.CMODE) and asynchronous mode is selected by writing a zero to CTRLA.CMODE. The internal clock source is selected by writing 0x1 to the Operation Mode bit group in the Control A register (CTRLA.MODE) and the external clock source is selected by writing 0x0 to CTRLA.MODE.

The SERCOM baud-rate generator is configured as shown in Figure 25-3. When CTRLA.CMODE is zero, the baud-rate generator is automatically set to asynchronous mode and the 16-bit Baud register value is used. When CTRLA.CMODE is one, the baud-rate generator is automatically set to synchronous mode and the eight LSBs of the Baud register are used. Refer to "Clock Generation – Baud-Rate Generator" on page 430 for details on configuring the baud rate.

Figure 25-3. Clock Generation

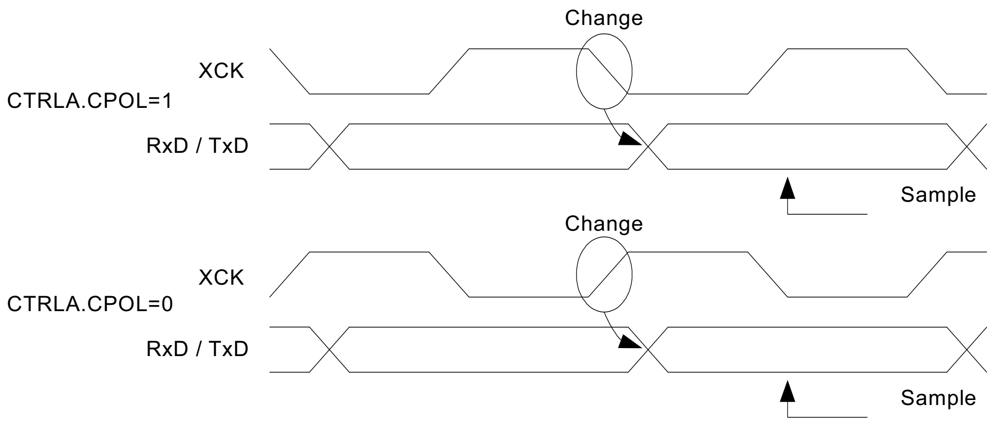


Synchronous Clock Operation

When synchronous mode is used, the CTRLA.MODE bit group controls whether the transmission clock (XCK line) is an input or output. The dependency between the clock edges and data sampling or data change is the same for internal and external clocks. Data input on the RxD pin is sampled at the opposite XCK clock edge as data is driven on the TxD pin.

The Clock Polarity bit in the Control A register (CTRLA.CPOL) selects which XCK clock edge is used for RxD sampling and which is used for TxD change. As shown in Figure 25-4, when CTRLA.CPOL is zero, the data will be changed on the rising XCK edge and sampled on the falling XCK edge. If CTRLA.CPOL is one, the data will be changed on the falling edge of XCK and sampled on the rising edge of XCK.

Figure 25-4. Synchronous Mode XCK Timing



When the clock is provided through XCK (CTRLA.MODE is 0x0), the shift registers operate directly on the XCK clock. This means that XCK is not synchronized with the system clock and, therefore, can operate at frequencies up to the system frequency.

25.6.2.4 Data Register

The USART Transmit Data register (TxDATA) and USART Receive Data register(RxDATA) share the same I/O address, referred to as the Data register (DATA). Writing the DATA register will update the Transmit Data register. Reading the DATA register will return the contents of the Receive Data register.

25.6.2.5 Data Transmission

A data transmission is initiated by loading the DATA register with the data to be sent. The data in TxDATA is moved to the shift register when the shift register is empty and ready to send a new frame. When the shift register is loaded with data, one complete frame will be transmitted.

The Transmit Complete interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.TXC) is set, and the optional interrupt is generated, when the entire frame plus stop bit(s) have been shifted out and there is no new data written to the DATA register.

The DATA register should only be written when the Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) is set, which indicates that the register is empty and ready for new data.

Disabling the Transmitter

Disabling the transmitter will not become effective until any ongoing and pending transmissions are completed, i.e., when the transmit shift register and TxDATA do not contain data to be transmitted. The transmitter is disabled by writing a zero to the Transmitter Enable bit in the Control B register (CTRLB.TXEN).

25.6.2.6 Data Reception

The receiver starts data reception when a valid start bit is detected. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the receive shift register until the first stop bit of a frame is received. When the first stop bit is received and a complete serial frame is present in the receive shift register, the contents of the shift register will be moved into the two-level receive buffer. The Receive Complete interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set, and the optional interrupt is generated. A second stop bit will be ignored by the receiver.

The received data can be read by reading the DATA register. DATA should not be read unless the Receive Complete interrupt flag is set.

Disabling the Receiver

Disabling the receiver by writing a zero to the Receiver Enable bit in the Control B register (CTRLB.RXEN) will flush the two-level receive buffer, and data from ongoing receptions will be lost.

Error Bits

The USART receiver has three error bits. The Frame Error (FERR), Buffer Overflow (BUFOVF) and Parity Error (PERR) bits can be read from the Status (STATUS) register. Upon error detection, the corresponding bit will be set until it is cleared by writing a one to it. These bits are also automatically cleared when the receiver is disabled.

There are two methods for buffer overflow notification. When the immediate buffer overflow notification bit (CTRLA.IBON) is set, STATUS.BUFOVF is raised immediately upon buffer overflow. Software can then empty the receive FIFO by reading RxDATA until the receive complete interrupt flag (INTFLAG.RXC) goes low.

When CTRLA.IBON is zero, the buffer overflow condition travels with data through the receive FIFO. After the received data is read, STATUS.BUFOVF will be set along with INTFLAG.RXC.

Asynchronous Data Reception

The USART includes a clock recovery and data recovery unit for handling asynchronous data reception. The clock recovery logic is used to synchronize the incoming asynchronous serial frames at the RxD pin to the internally generated baud-rate clock. The data recovery logic samples and applies a low-pass filter to each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud-rate clock, the rate of the incoming frames and the frame size (in number of bits).

Asynchronous Operational Range

The operational range of the receiver depends on the difference between the received bit rate and the internally generated baud rate. If the baud rate of an external transmitter is too high or too low compared to the internally generated baud rate, the receiver will not be able to synchronize the frames to the start bit.

There are two possible sources for a mismatch in baud rate. The reference clock will always have some minor instability. In addition, the baud-rate generator can not always do an exact division of the reference clock frequency to get the baud rate desired. In this case, the BAUD register value should be selected to give the lowest possible error. Refer to "Asynchronous Arithmetic Mode BAUD Value Selection" on page 432 for details.

Recommended maximum receiver baud-rate errors for various character sizes are shown in the table below.

Table 25-2. Asynchronous Receiver Error for x16 Oversampling

D (Data bits + Parity)	R_{SLOW}(%)	R_{FAST}(%)	Max Total Error (%)	Recommended Max Rx Error (%)
5	94.12	107.69	+5.88/-7.69	±2.5
6	94.92	106.67	+5.08/-6.67	±2.0
7	95.52	105.88	+4.48/-5.88	±2.0
8	96.00	105.26	+4.00/-5.26	±2.0
9	96.39	104.76	+3.61/-4.76	±1.5
10	96.70	104.35	+3.30/-4.35	±1.5

The recommended maximum receiver baud-rate error assumes that the receiver and transmitter equally divide the maximum total error.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate:

$$R_{SLOW} = \frac{16(D+1)}{16(D+1)+6} \qquad R_{FAST} = \frac{16(D+2)}{16(D+1)+8}$$

where:

- S is the number of samples per bit (S = 16, 8 or 3)
- S_F is the first sample number used for majority voting (S_F = 7, 3, or 2) when CTRLA.SAMPA=0.
- S_M is the middle sample number used for majority voting (S_M = 8, 4, or 2) when CTRLA.SAMPA=0.
- D is the sum of character size and parity size (D = 5 to 10 bits)
- R_{SLOW} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- R_{FAST} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate

25.6.3 Additional Features

25.6.3.1 Parity

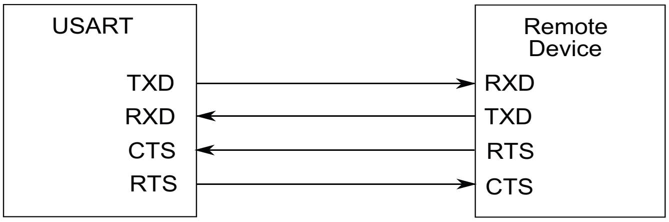
Even or odd parity can be selected for error checking by writing 0x1 to the Frame Format bit group in the Control A register (CTRLA.FORM). If even parity is selected by writing a zero to the Parity Mode bit in the Control B register (CTRLB.PMODE), the parity bit of the outgoing frame is set to one if the number of data bits that are one is odd (making the total number of ones even). If odd parity is selected by writing a one to CTRLB.PMODE, the parity bit of the outgoing frame is set to one if the number of data bits that are one is even (making the total number of ones odd).

When parity checking is enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit of the corresponding frame. If a parity error is detected, the Parity Error bit in the Status register (STATUS.PERR) is set.

25.6.3.2 Hardware Handshaking

The USART features an out-of-band hardware handshaking flow control mechanism, implemented by connecting the RTS and CTS pins with the remote device, as shown in Figure 25-5.

Figure 25-5. Connection with a Remote Device for Hardware Handshaking

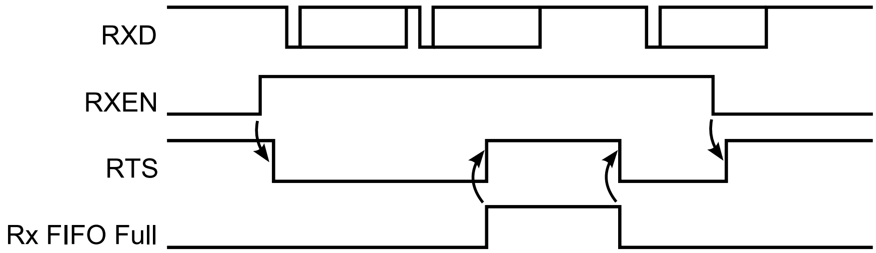


Hardware handshaking is only available with the following configuration:

- USART with internal clock (CTRLA.MODE = 1).
- Asynchronous mode (CTRLA.CMODE = 0).
- Flow control pinout (CTRLA.TXPO = 2).

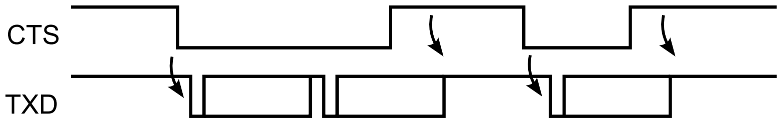
The receiver drives its RTS pin high when disabled, or when the receive FIFO is full. This indicates to the remote device that it must stop transmitting after the ongoing transmission is complete. Enabling and disabling the receiver by writing RXEN will set/clear the RTS pin after a synchronization delay. When the receive FIFO goes full, RTS is immediately set and the frame that is currently being received will be stored in the shift register until the receive FIFO is no longer full.

Figure 25-6. Receiver Behavior when Operating with Hardware Handshaking



The current CTS level is available in the STATUS register (STATUS.CTS). Character transmission will only start if CTS is low. When CTS goes high, the transmitter will stop transmitting after the ongoing transmission is complete.

Figure 25-7. Transmitter Behavior when Operating with Hardware Handshaking



25.6.3.3 IrDA Modulation and Demodulation

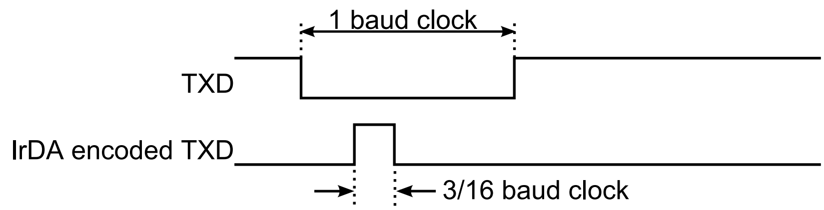
IrDA modulation and demodulation is available with the following configuration. When enabled, transmission and reception is IrDA compliant up to 115.2 kb/s.

- IrDA encoding enabled (CTRLB.ENC=1).
- Asynchronous mode (CTRLA.CMODE = 0).

- 16x sample rate (CTRLA.SAMPR[0] = 0).

During transmission, each low bit is transmitted as a high pulse with width as 3/16 of the baud rate period as illustrated in
Figure 25-8.

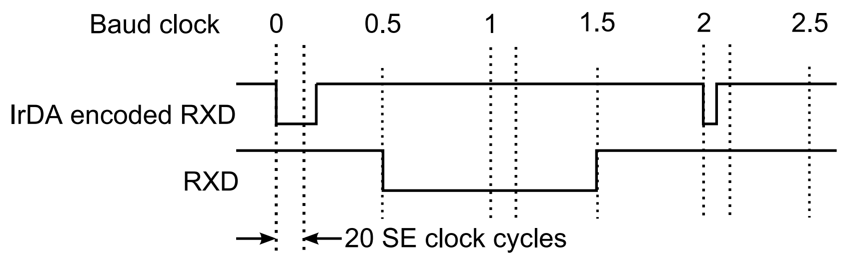
Figure 25-8. IrDA Transmit Encoding



The reception decoder has two main functions. The first is to synchronize the incoming data to the IrDA baud rate counter. Synchronization is performed at the start of each zero pulse. The second function is to decode incoming Rx data. If a pulse width meets the minimum length set by configuration (RXPL.RXPL), it is accepted. When the baud rate counter reaches its middle value (1/2 bit length), it is transferred to the receiver.

Figure 25-9 illustrates reception where RXPL.RXPL is set to 19. This indicates that the pulse width should be at least 20 SE clock cycles. When assuming BAUD = 0xE666 or 160 SE cycles per bit, this corresponds to 2/16 baud clock as minimum pulse width required. In this case the first bit is accepted as a zero, the second bit is a one, and the third bit is also a one. A low pulse is rejected since it does not meet the minimum requirement of 2/16 baud clock.

Figure 25-9. IrDA Receive Decoding



Note that the polarity of the transmitter and receiver are opposite. During transmission, a zero bit is transmitted as a one pulse. During reception, an accepted zero pulse is received as a zero bit.

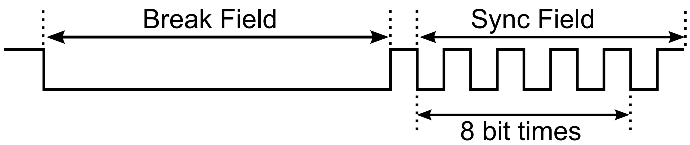
25.6.3.4 Break Character Detection and Auto-baud

Break character detection and auto-baud are available with the following configuration:

- Auto-baud frame format (CTRLA.FORM = 0x04 or 0x05)
- Asynchronous mode (CTRLA.CMODE = 0).
- 16x sample rate using fractional baud rate generation (CTRLA.SAMPR = 1).

The auto-baud follows the LIN format. All LIN Frames start with a Break Field followed by a Sync Field. The USART uses a break detection threshold of greater than 11 nominal bit times at the configured baud rate. At any time, if more than 11 consecutive dominant bits are detected on the bus, the USART detects a Break Field. When a Break Field has been detected, the Receive Break interrupt flag (INTFLAG.RXBRK) is set and the USART expects the Sync Field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized. If the received Sync character is not 0x55, then the Inconsistent Sync Field error flag (STATUS.ISF) is set along with the Error interrupt flag (INTFLAG.ERROR) and the baud rate is unchanged.

Figure 25-10.LIN Break and Sync Fields



After a break field is detected and the start bit of the Sync Field is detected, a counter is started. The counter is then incremented for the next 8 bit times of the Sync Field. At the end of these 8 bit times, the counter is stopped. At this moment, the 13 most significant bits of the counter (value divided by 8) gives the new clock divider (BAUD.BAUD) and the 3 least significant bits of this value (the remainder) gives the new Fractional Part (BAUD.FP). When the Sync Field has been received, the clock divider (BAUD.BAUD) and the Fractional Part (BAUD.FP) are updated in the Baud Rate Generator register (BAUD) after a synchronization delay.

After the Break and Sync Fields, n characters of data can be received.

25.6.3.5 Collision Detection

When the receiver and transmitter are connected either through pin configuration or externally, transmit collision can be detected by setting the Collision Detection Enable bit (CTRLB.COLDEN). For collision to be detected, the receiver and transmitter must be enabled (CTRLB.RXEN=1 and CTRLB.TXEN=1).

Collision detection is performed for each bit transmitted by checking the received value vs the transmit value as shown in Figure 25-11. While the transmitter is idle (no transmission in progress), characters can be received on RxD without triggering a collision.

Figure 25-11.Collision Checking

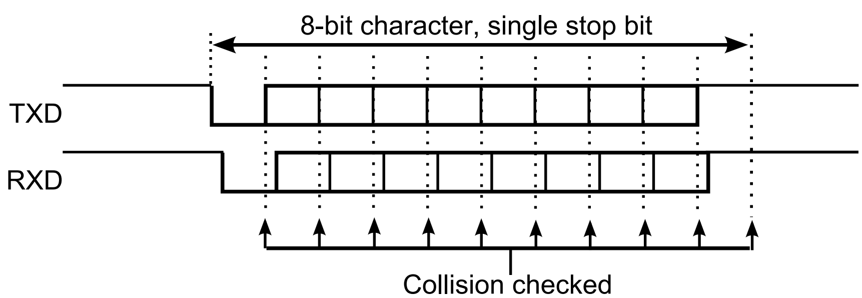
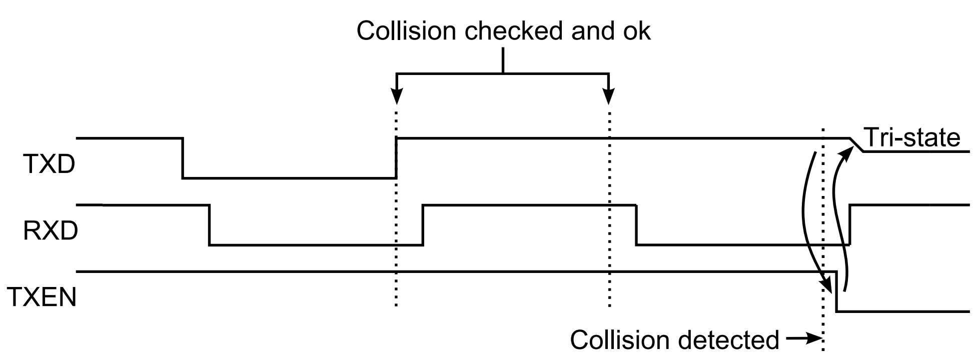


Figure 25-12 shows the conditions for a collision detection. In this case, the start bit and the first data bit are received with the same value as transmitted. The second received data bit is found to be different than the transmitted bit at the detection point which indicates a collision.

Figure 25-12.Collision Detected



When a collision is detected, the USART automatically follows this sequence:

- The current transfer is aborted.

- The transmit buffer is flushed.
- The transmitter is disabled (CTRLB.TXEN=0).
  - This commences immediately and is complete after synchronization time. The CTRLB Synchronization Busy bit (SYNCBUSY.CTRLB) will be set until this is complete.
  - This results in the TxD pin being tri-stated.
- The Collision Detected bit (STATUS.COLL) is set along with the Error interrupt flag (INTFLAG.ERROR).
- Since the transmit buffer no longer contains data, the Transmit Complete interrupt flag (INTFLAG.TXC) is set.

After a collision, software must manually enable the transmitter before continuing. Software must ensure CTRLB Synchronization Busy bit (SYNCBUSY.CTRLB) is not asserted before re-enabling the transmitter.

25.6.3.6 Loop-back Mode

By configuring the Receive Data Pinout (CTRLA.RXPO) and Transmit Data Pinout (CTRLA.TXPO) to use the same data pins for transmit and receive, loop-back is achieved. The loop-back is through the pad, so the signal is also available externally.

25.6.3.7 Start-of-Frame Detection

The USART start-of-frame detector can wake up the CPU when it detects a start bit. In standby sleep mode, the internal fast startup oscillator must be selected as the GCLK_SERCOMx_CORE source.

When a 1-to-0 transition is detected on RxD, the 8MHz Internal Oscillator is powered up and the USART clock is enabled. After startup, the rest of the data frame can be received, provided that the baud rate is slow enough in relation to the fast startup internal oscillator start-up time. Refer to "Electrical Characteristics" on page 648 for details. The start-up time of this oscillator varies with supply voltage and temperature.

The USART start-of-frame detection works both in asynchronous and synchronous modes. It is enabled by writing a one to the Start of Frame Detection Enable bit in the Control B register (CTRLB.SFDE). If the Receive Start Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.RXS) is set, the Receive Start interrupt is generated immediately when a start is detected. When using start-of-frame detection without the Receive Start interrupt, start detection will force the 8MHz Internal Oscillator and USART clock active while the frame is being received, but the CPU will not wakeup until the Receive Complete interrupt is generated, if enabled.

25.6.3.8 Sample Adjustment

In asynchronous mode (CTRLA.CMODE=0), three samples in the middle are used to determine the value based on majority voting. The three samples used for voting can be selected using the Sample Adjustment bit field in Control A register (CTRLA.SAMPA). When CTRLA.SAMPA is set to zero, samples 7-8-9 are used for 16x over sampling and samples 3-4-5 are used for 8x over sampling.

25.6.4 DMA, Interrupts and Events

Table 25-3. Module Request for SERCOM USART

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Data Register Empty	x			x	When data is written
Transmit Complete	x				
Receive Complete	x			x	When data is read
Receive Start	x				
Clear to Send Input Change	x				
Receive Break	x				
Error	x				

25.6.4.1 DMA Operation

The USART generates the following DMA requests.

- Data received (RX): The request is set when data is available in the receive FIFO. The request is cleared when DATA is read.
- Data transmit (TX): The request is set when the transmit buffer (TX DATA) is empty. The request is cleared when DATA is written.

25.6.4.2 Interrupts

The USART has the following interrupt sources:

- Error
- Received Break
- Clear to Send Input Change
- Receive Start
- Receive Complete
- Transmit Complete
- Data Register Empty

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the USART is reset. See the register description for details on how to clear interrupt flags.

The USART has one common interrupt request line for all the interrupt sources. The user must read INTFLAG to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

25.6.4.3 Events

Not applicable.

25.6.5 Sleep Mode Operation

When using internal clocking, writing the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY) to one will allow GCLK_SERCOMx_CORE to be enabled in all sleep modes. Any interrupt can wake up the device.

When using external clocking, writing a one to CTRLA.RUNSTDBY will allow the Receive Start or Receive Complete interrupt.to wake up the device.

If CTRLA.RUNSTDBY is zero, the internal clock will be disabled when any ongoing transfer is finished. A Receive Start or Transfer Complete interrupt can wake up the device. When using external clocking, this will be disconnected when any ongoing transfer is finished, and all reception will be dropped.

25.6.6 Synchronization

Due to the asynchronicity between CLK_SERCOMx_APB and GCLK_SERCOMx_CORE, some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the corresponding Synchronization Busy bit in the Synchronization Busy register (SYNCBUSY) will be set immediately, and cleared when synchronization is complete.

If an operation that requires synchronization is executed while the corresponding SYNCBUSY bit is one, a peripheral bus error is generated.

The following bits need synchronization when written:

- Software Reset bit in the Control A register (CTRLA.SWRST). SYNCBUSY.SWRST is set to one while synchronization is in progress.
- Enable bit in the Control A register (CTRLA.ENABLE). SYNCBUSY.ENABLE is set to one while synchronization is in progress.
- Receiver Enable bit in the Control B register (CTRLB.RXEN). SYNCBUSY.CTRLB is set to one while synchronization is in progress.
- Transmitter Enable bit in the Control B register (CTRLB.TXEN). SYNCBUSY.CTRLB is set to one while synchronization is in progress.

Synchronization is denoted by the Write-Synchronized property in the register description.

25.7 Register Summary

Table 25-4. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
0x01		15:8	SAMPR[2:0]							IBON
0x02		23:16	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
0x03		31:24		DORD	CPOL	CMODE	FORM[3:0]			
0x04	CTRLB	7:0		SBMODE				CHSIZE[2:0]		
0x05		15:8			PMODE			ENC	SFDE	COLDEN
0x06		23:16							RXEN	TXEN
0x07		31:24								
0x04	CTRLB	7:0		SBMODE				CHSIZE[2:0]		
0x05		15:8			PMODE			ENC		COLDEN
0x06		23:16							RXEN	TXEN
0x07		31:24								
0x08	Reserved									
0x09	Reserved									
0x0A	Reserved									
0x0B	Reserved									
0x0C	BAUD	7:0	BAUD[7:0]							
0x0D		15:8	FP[2:0]/BAUD[15:13]			BAUD[12:8]				
0x0E	RXPL	7:0	RXPL[7:0]							
0x0F	Reserved									
0x10	Reserved									
0x11	Reserved									
0x12	Reserved									
0x13	Reserved									
0x14	INTENCLR	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x14	INTENCLR	7:0	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
0x15	Reserved									
0x16	INTENSET	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x16	INTENSET	7:0	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x18	INTFLAG	7:0	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
0x19	Reserved									
0x1A	STATUS	7:0			COLL	ISF	CTS	BUFOVF	FERR	PERR
0x1B		15:8								

Offset	Name	Bit Pos.								
0x1C	SYNCBUSY	7:0						CTRLB	ENABLE	SWRST
0x1D		15:8								
0x1E		23:16								
0x1F		31:24								
0x20	Reserved									
0x21	Reserved									
0x22	Reserved									
0x23	Reserved									
0x24	Reserved									
0x25	Reserved									
0x26	Reserved									
0x27	Reserved									
0x28	DATA	7:0	DATA[7:0]							
0x29		15:8								DATA[8]
0x2A	Reserved									
0x2B	Reserved									
0x2C	Reserved									
0x2D	Reserved									
0x2E	Reserved									
0x2F	Reserved									
0x30	DBGCTRL	7:0								DBGSTOP

25.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to "Register Access Protection" on page 437 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Synchronized property in each individual register description. Refer to "Synchronization" on page 448 for details.

Some registers are enable-protected, meaning they can only be written when the USART is disabled. Enable-protection is denoted by the Enable-Protected property in each individual register description.

25.8.1 Control A

Name: CTRLA

Offset: 0x00

Reset: 0x00000000

Property: Enable-Protected, Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
		DORD	CPOL	CMODE	FORM[3:0]			
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
Access	R/W	R/W	R/W	R/W	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SAMPR[2:0]							IBON
Access	R/W	R/W	R/W	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 31 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 30 – DORD: Data Order**
  This bit indicates the data order when a character is shifted out from the Data register.

  0: MSB is transmitted first.

  1: LSB is transmitted first.

  This bit is not synchronized.

- **Bit 29 – CPOL: Clock Polarity**
  This bit indicates the relationship between data output change and data input sampling in synchronous mode.

  This bit is not synchronized.

Table 25-5. Clock Polarity

CPOL	TxD Change	RxD Sample
0x0	Rising XCK edge	Falling XCK edge
0x1	Falling XCK edge	Rising XCK edge

- **Bit 28 – CMODE: Communication Mode**
  This bit indicates asynchronous or synchronous communication.
  0: Asynchronous communication.
  1: Synchronous communication.
  This bit is not synchronized.

- **Bits 27:24 – FORM[3:0]: Frame Format**
  These bits define the frame format.
  These bits are not synchronized.

Table 25-6. Frame Format

FORM[3:0]	Description
0x0	USART frame
0x1	USART frame with parity
0x2-0x3	Reserved
0x4	Auto-baud -- break detection and auto-baud.
0x5	Auto-baud -- break detection and auto-baud with parity
0x6-0xF	Reserved

- **Bits 23:22 – SAMPA[1:0]: Sample Adjustment**
  These bits define the sample adjustment.
  These bits are not synchronized.

Table 25-7. Sample Adjustment

SAMPA[1:0]	16x Over-sampling (CTRLA.SAMPR=0 or 1)	8x Over-sampling (CTRLA.SAMPR=2 or 3)
0x0	7-8-9	3-4-5
0x1	9-10-11	4-5-6
0x2	11-12-13	5-6-7
0x3	13-14-15	6-7-8

- **Bits 21:20 – RXPO[1:0]: Receive Data Pinout**
  These bits define the receive data (RxD) pin configuration.
  These bits are not synchronized.

Table 25-8. Receive Data Pinout

RXPO[1:0]	Name	Description
0x0	PAD[0]	SERCOM PAD[0] is used for data reception
0x1	PAD[1]	SERCOM PAD[1] is used for data reception
0x2	PAD[2]	SERCOM PAD[2] is used for data reception
0x3	PAD[3]	SERCOM PAD[3] is used for data reception

- **Bits 19:18 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 17:16 – TXPO[1:0]: Transmit Data Pinout**
  These bits define the transmit data (TxD) and XCK pin configurations.
  This bit is not synchronized.

Table 25-9. Transmit Data Pinout

TXPO	TxD Pin Location	XCK Pin Location (When Applicable)	RTS	CTS
0x0	SERCOM PAD[0]	SERCOM PAD[1]	N/A	N/A
0x1	SERCOM PAD[2]	SERCOM PAD[3]	N/A	N/A
0x2	SERCOM PAD[0]	N/A	SERCOM PAD[2]	SERCOM PAD[3]
0x3	Reserved			

- **Bits 15:13 – SAMPR[2:0]: Sample Rate**
  These bits define the sample rate.
  These bits are not synchronized.

Table 25-10. Sample Rate

SAMPR[2:0]	Description
0x0	16x over-sampling using arithmetic baud rate generation.
0x1	16x over-sampling using fractional baud rate generation.
0x2	8x over-sampling using arithmetic baud rate generation.
0x3	8x over-sampling using fractional baud rate generation.
0x4	3x over-sampling using arithmetic baud rate generation.
0x5-0x7	Reserved

- **Bits 12:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 8 – IBON: Immediate Buffer Overflow Notification**
  This bit controls when the buffer overflow status bit (STATUS.BUFOVF) is asserted when a buffer overflow occurs.
  0: STATUS.BUFOVF is asserted when it occurs in the data stream.
  1: STATUS.BUFOVF is asserted immediately upon buffer overflow.

- **Bit 7 – RUNSTDBY: Run In Standby**
  This bit defines the functionality in standby sleep mode.
  This bit is not synchronized.

Table 25-11. Run In Standby

RUNSTDBY	External Clock	Internal Clock
0x0	External clock is disconnected when ongoing transfer is finished. All reception is dropped.	Generic clock is disabled when ongoing transfer is finished. The device can wake up on Receive Start or Transfer Complete interrupt.
0x1	Wake on Receive Start or Receive Complete interrupt.	Generic clock is enabled in all sleep modes. Any interrupt can wake up the device.

- **Bits 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:2 – MODE: Operating Mode**
  These bits must be written to 0x0 or 0x1 to select the USART serial communication interface of the SERCOM.
  0x0: USART with external clock.
  0x1: USART with internal clock.
  These bits are not synchronized.

- **Bit 1 – ENABLE: Enable**
  0: The peripheral is disabled or being disabled.
  1: The peripheral is enabled or being enabled.
  Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable Synchronization Busy bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE is cleared when the operation is complete.
  This bit is not enable-protected.

- **Bit 0 – SWRST: Software Reset**
  0: There is no reset operation ongoing.
  1: The reset operation is ongoing.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.
  Writing a one to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.
  Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.
  This bit is not enable-protected.

25.8.2 Control B

Name: CTRLB

Offset: 0x04

Reset: 0x00000000

Property: Enable-Protected, Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
							RXEN	TXEN
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			PMODE			ENC	SFDE	COLDEN
			PMODE			ENC		COLDEN
Access	R	R	R/W	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		SBMODE				CHSIZE[2:0]		
Access	R	R/W	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:18 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 17 – RXEN: Receiver Enable**
  0: The receiver is disabled or being enabled.
  1: The receiver is enabled or will be enabled when the USART is enabled.
  Writing a zero to this bit will disable the USART receiver. Disabling the receiver will flush the receive buffer and clear the FERR, PERR and BUFOVF bits in the STATUS register.
  Writing a one to CTRLB.RXEN when the USART is disabled will set CTRLB.RXEN immediately. When the USART is enabled, CTRLB.RXEN will be cleared, and SYNCBUSY.CTRLB will be set and remain set until the receiver is enabled. When the receiver is enabled, CTRLB.RXEN will read back as one.
  Writing a one to CTRLB.RXEN when the USART is enabled will set SYNCBUSY.CTRLB, which will remain set until the receiver is enabled, and CTRLB.RXEN will read back as one.
  This bit is not enable-protected.

- **Bit 16 – TXEN: Transmitter Enable**
  0: The transmitter is disabled or being enabled.
  1: The transmitter is enabled or will be enabled when the USART is enabled.
  Writing a zero to this bit will disable the USART transmitter. Disabling the transmitter will not become effective until ongoing and pending transmissions are completed.
  Writing a one to CTRLB.TXEN when the USART is disabled will set CTRLB.TXEN immediately. When the USART is enabled, CTRLB.TXEN will be cleared, and SYNCBUSY.CTRLB will be set and remain set until the transmitter is enabled. When the transmitter is enabled, CTRLB.TXEN will read back as one.
  Writing a one to CTRLB.TXEN when the USART is enabled will set SYNCBUSY.CTRLB, which will remain set until the receiver is enabled, and CTRLB.TXEN will read back as one.
  This bit is not enable-protected.

- **Bits 15:14 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 13 – PMODE: Parity Mode**
  This bit selects the type of parity used when parity is enabled (CTRLA.FORM is one). The transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The receiver will generate a parity value for the incoming data and parity bit, compare it to the parity mode and, if a mismatch is detected, STATUS.PERR will be set.
  0: Even parity.
  1: Odd parity.
  This bit is not synchronized.

- **Bits 12:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 10 – ENC: Encoding Format**
  This bit selects the data encoding format.
  0: Data is not encoded.
  1: Data is IrDA encoded.
  This bit is not synchronized.

- **Bit 9 – SFDE: Start of Frame Detection Enable**
  This bit controls whether the start-of-frame detector will wake up the device when a start bit is detected on the RxD line, according to the table below.
  This bit is not synchronized.

SFDE	INTENSET.RXS	INTENSET.RXC	Description
0	X	X	Start-of-frame detection disabled.
1	0	0	Reserved
1	0	1	Start-of-frame detection enabled. RXC wakes up the device from all sleep modes.
1	1	0	Start-of-frame detection enabled. RXS wakes up the device from all sleep modes.
1	1	1	Start-of-frame detection enabled. Both RXC and RXS wake up the device from all sleep modes.

- **Bit 9 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 8 -- COLDEN: Collision Detection Enable**
  This bit enables collision detection.
  0: Collision detection is not enabled.
  1: Collision detection is enabled.
  This bit is not synchronized.

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 6 – SBMODE: Stop Bit Mode**
  This bit selects the number of stop bits transmitted.
  0: One stop bit.
  1: Two stop bits.
  This bit is not synchronized.

- **Bits 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – CHSIZE[2:0]: Character Size**
  These bits select the number of bits in a character.
  These bits are not synchronized.

Table 25-12. Character Size

CHSIZE[2:0]	Description
0x0	8 bits
0x1	9 bits
0x2-0x4	Reserved
0x5	5 bits
0x6	6 bits
0x7	7 bits

25.8.3 Baud

Name: BAUD

Offset: 0x0C

Reset: 0x0000

Property: Enable-Protected, Write-Protected

Bit	15	14	13	12	11	10	9	8
	FP[2:0]/BAUD[15:13]			BAUD[12:8]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BAUD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Arithmetic Baud Rate Generation (CTRLA.SAMPR[0]=0)

- **Bits 15:0 – BAUD[15:0]: Baud Value**
  These bits control the clock generation, as described in the SERCOM Baud Rate section.

Fractional Baud Rate Generation (CTRLA.SAMPR[0]=1)

- **Bits 15:13 – FP[2:0]: Fractional Part**
  These bits control the clock generation, as described in the SERCOM Baud Rate section.

- **Bits 15:0 – BAUD[12:0]: Baud Value**
  These bits control the clock generation, as described in the SERCOM Baud Rate section.

25.8.4 Receive Pulse Length Register

Name: RXPL

Offset: 0x0E

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	RXPL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:0 – RXPL[7:0]: Receive Pulse Length**
  When the encoding format is set to IrDA (CTRLB.ENC=1), these bits control the minimum pulse length that is required for a pulse to be accepted by the IrDA receiver with regards to the serial engine clock period.

  $$PULSE \geq (RXPL + 1) \times SE_{per}$$

25.8.5 Interrupt Enable Clear

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Name: INTENCLR

Offset: 0x14

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
Access	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7– ERROR: Error Interrupt Enable**
  0: Error interrupt is disabled.
  1: Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.
- **Bit 6 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bit 5 – RXBRK: Receive Break Interrupt Enable**
  0: Receive Break interrupt is disabled.
  1: Receive Break interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Receive Break Interrupt Enable bit, which disables the Receive Break interrupt.
- **Bit 4 – CTSIC: Clear to Send Input Change Interrupt Enable**
  0: Clear To Send Input Change interrupt is disabled.
  1: Clear To Send Input Change interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Clear To Send Input Change Interrupt Enable bit, which disables the Clear To Send Input Change interrupt.
- **Bit 3 – RXS: Receive Start Interrupt Enable**
  0: Receive Start interrupt is disabled.
  1: Receive Start interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Receive Start Interrupt Enable bit, which disables the Receive Start interrupt.
- **Bits 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 2 – RXC: Receive Complete Interrupt Enable**

  0: Receive Complete interrupt is disabled.

  1: Receive Complete interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Receive Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

- **Bit 1 – TXC: Transmit Complete Interrupt Enable**

  0: Transmit Complete interrupt is disabled.

  1: Transmit Complete interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Transmit Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

- **Bit 0 – DRE: Data Register Empty Interrupt Enable**

  0: Data Register Empty interrupt is disabled.

  1: Data Register Empty interrupt is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Data Register Empty Interrupt Enable bit, which disables the Data Register Empty interrupt.

25.8.6 Interrupt Enable Set

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR) .

Name: INTENSET

Offset: 0x16

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
Access	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – ERROR: Error Interrupt Enable**
  0: Error interrupt is disabled.
  1: Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

- **Bits 6 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 5– RXBRK: Receive Break Interrupt Enable**
  0: Receive Break interrupt is disabled.
  1: Receive Break interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Receive Break Interrupt Enable bit, which enables the Receive Break interrupt.

- **Bit 4 – CTSIC: Clear to Send Input Change Interrupt Enable**
  0: Clear To Send Input Change interrupt is disabled.
  1: Clear To Send Input Change interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Clear To Send Input Change Interrupt Enable bit, which enables the Clear To Send Input Change interrupt.

- **Bit 3 – RXS: Receive Start Interrupt Enable**
  0: Receive Start interrupt is disabled.
  1: Receive Start interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Receive Start Interrupt Enable bit, which enables the Receive Start interrupt.

- **Bits 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 2 – RXC: Receive Complete Interrupt Enable**
  0: Receive Complete interrupt is disabled.
  1: Receive Complete interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Receive Complete Interrupt Enable bit, which enables the Receive Complete interrupt.
- **Bit 1– TXC: Transmit Complete Interrupt Enable**
  0: Transmit Complete interrupt is disabled.
  1: Transmit Complete interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Transmit Complete Interrupt Enable bit, which enables the Transmit Complete interrupt.
- **Bit 0 – DRE: Data Register Empty Interrupt Enable**
  0: Data Register Empty interrupt is disabled.
  1: Data Register Empty interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Data Register Empty Interrupt Enable bit, which enables the Data Register Empty interrupt.

25.8.7 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x18

Reset: 0x00

Property:

Bit	7	6	5	4	3	2	1	0
	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
Access	R/W	R	R/W	R/W	R/W	R	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bit 7– ERROR: Error**
  This flag is cleared by writing a one to it.
  This bit is set when any error is detected. Errors that will set this flag have corresponding status flags in the STATUS register. Errors that will set this flag are COLL, ISF, BUFOVF, FERR, and PERR.Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the flag.

- **Bits 6 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 5 – RXBRK: Receive Break**
  This flag is cleared by writing a one to it.
  This flag is set when auto-baud is enabled (CTRLA.FORM) and a break character is received.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the flag.

- **Bit 4 – CTSIC: Clear to Send Input Change**
  This flag is cleared by writing a one to it.
  This flag is set when a change is detected on the CTS pin.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the flag.

- **Bit 3 – RXS: Receive Start**
  This flag is cleared by writing a one to it.
  This flag is set when a start condition is detected on the RxD line and start-of-frame detection is enabled (CTRLB.SFDE is one).
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Receive Start interrupt flag.

- **Bits 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 2 – RXC: Receive Complete**
  This flag is cleared by reading the Data register (DATA) or by disabling the receiver.
  This flag is set when there are unread data in DATA.

Writing a zero to this bit has no effect.

Writing a one to this bit has no effect.

- **Bit 1 – TXC: Transmit Complete**

  This flag is cleared by writing a one to it or by writing new data to DATA.

  This flag is set when the entire frame in the transmit shift register has been shifted out and there are no new data in DATA.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the flag.

- **Bit 0 – DRE: Data Register Empty**

  This flag is cleared by writing new data to DATA.

  This flag is set when DATA is empty and ready to be written.

  Writing a zero to this bit has no effect.

  Writing a one to this bit has no effect.

25.8.8 Status

Name: STATUS

Offset: 0x1A

Reset: 0x0000

Property:

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			COLL	ISF	CTS	BUFOVF	FERR	PERR
Access	R	R	R/W	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 5 – COLL: Collision Detected**
  This bit is cleared by writing a one to the bit or by disabling the receiver.
  This bit is set when collision detection is enabled (CTRLB.COLDEN) and a collision is detected.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear it.

- **Bit 4 – ISF: Inconsistent Sync Field**
  This bit is cleared by writing a one to the bit or by disabling the receiver.
  This bit is set when the frame format is set to auto-baud (CTRLA.FORM) and a sync field not equal to 0x55 is received.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear it.

- **Bit 3 – CTS: Clear to Send**
  This bit indicates the current level of the CTS pin when flow control is enabled (CTRLA.TXPO).
  Writing a zero to this bit has no effect.
  Writing a one to this bit has no effect.

- **Bit 2 – BUFOVF: Buffer Overflow**
  Reading this bit before reading the Data register will indicate the error status of the next character to be read.
  This bit is cleared by writing a one to the bit or by disabling the receiver.
  This bit is set when a buffer overflow condition is detected. A buffer overflow occurs when the receive buffer is full, there is a new character waiting in the receive shift register and a new start bit is detected.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear it.

- **Bit 1 – FERR: Frame Error**
  Reading this bit before reading the Data register will indicate the error status of the next character to be read.

This bit is cleared by writing a one to the bit or by disabling the receiver.

This bit is set if the received character had a frame error, i.e., when the first stop bit is zero.

Writing a zero to this bit has no effect.

Writing a one to this bit will clear it.

- **Bit 0 – PERR: Parity Error**

  Reading this bit before reading the Data register will indicate the error status of the next character to be read.

  This bit is cleared by writing a one to the bit or by disabling the receiver.

  This bit is set if parity checking is enabled (CTRLA.FORM is 0x1 or 0x5) and a parity error is detected.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear it.

25.8.9 Synchronization Busy

Name: SYNCBUSY

Offset: 0x1C

Reset: 0x00000000

Property:

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						CTRLB	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2– CTRLB: CTRLB Synchronization Busy**
  Writing CTRLB when the SERCOM is enabled requires synchronization. When written, the SYNC-BUSY.CTRLB bit will be set until synchronization is complete. If CTRLB is written while SYNCBUSY.CTRLB is asserted, an APB error will be generated.

  0: CTRLB synchronization is not busy.

  1: CTRLB synchronization is busy.

- **Bit 1 – ENABLE: SERCOM Enable Synchronization Busy**
  Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. When written, the SYNC-BUSY.ENABLE bit will be set until synchronization is complete.

  Writes to any register (except for CTRLA.SWRST) while enable synchronization is on-going will be discarded and an APB error will be generated.

  0: Enable synchronization is not busy.

  1: Enable synchronization is busy.

- **Bit 0 – SWRST: Software Reset Synchronization Busy**
  Resetting the SERCOM (CTRLA.SWRST) requires synchronization. When written, the SYNCBUSY.SWRST bit will be set until synchronization is complete.

  Writes to any register while synchronization is on-going will be discarded and an APB error will be generated.

  0: SWRST synchronization is not busy.

  1: SWRST synchronization is busy.

25.8.10 Data

Name: DATA

Offset: 0x28

Reset: 0x0000

Property: -

Bit	15	14	13	12	11	10	9	8
								DATA[8]
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 8:0 – DATA[8:0]: Data**
  Reading these bits will return the contents of the Receive Data register. The register should be read only when the Receive Complete Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set. The status bits in STATUS should be read before reading the DATA value in order to get any corresponding error.

  Writing these bits will write the Transmit Data register. This register should be written only when the Data Register Empty Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.DRE) is set.

25.8.11 Debug Control

Name: DBGCTRL

Offset: 0x30

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
								DBGSTOP
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – DBGSTOP: Debug Stop Mode**
  This bit controls the baud-rate generator functionality when the CPU is halted by an external debugger.

  0: The baud-rate generator continues normal operation when the CPU is halted by an external debugger.

  1: The baud-rate generator is halted when the CPU is halted by an external debugger.

26. SERCOM SPI – SERCOM Serial Peripheral Interface

26.1 Overview

The serial peripheral interface (SPI) is one of the available modes in the Serial Communication Interface (SERCOM). Refer to "SERCOM – Serial Communication Interface" on page 427 for details.

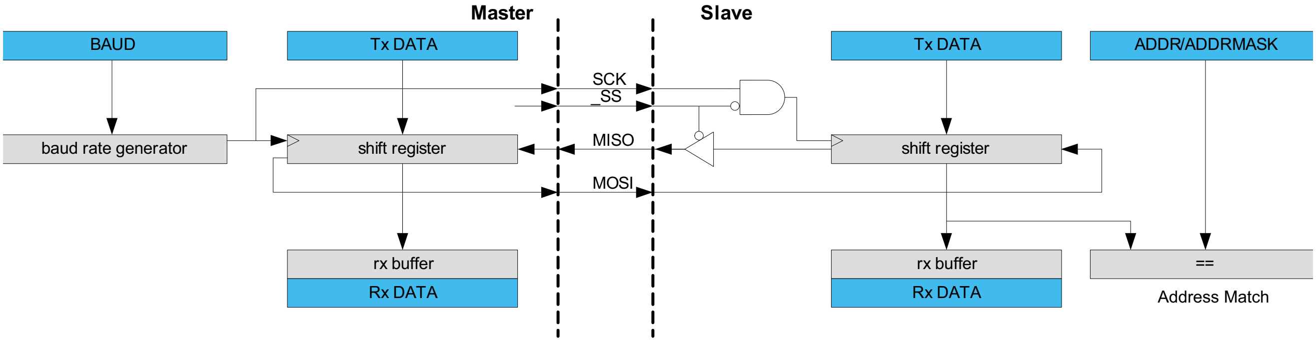
The SPI uses the SERCOM transmitter and receiver configured as shown in "Full-Duplex SPI Master Slave Interconnection" on page 473. Each side, master and slave, depicts a separate SPI containing a shift register, a transmit buffer and two receive buffers. In addition, the SPI master uses the SERCOM baud-rate generator, while the SPI slave can use the SERCOM address match logic. Fields shown in capital letters are synchronous to CLK_SERCOMx_APB and accessible by the CPU, while fields with lowercase letters are synchronous to the SCK clock.

26.2 Features

- Full-duplex, four-wire interface (MISO, MOSI, SCK, $\overline{SS}$)
- Single-buffered transmitter, double-buffered receiver
- Supports all four SPI modes of operation
- Single data direction operation allows alternate function on MISO or MOSI pin
- Selectable LSB- or MSB-first data transfer
- Can be used with DMA
- Master operation:
  - Serial clock speed up to 12MHz
  - 8-bit clock generator
  - Hardware controlled $\overline{SS}$
- Slave operation:
  - Serial clock speed up to the system clock
  - Optional 8-bit address match operation
  - Operation in all sleep modes
  - Wake on $\overline{SS}$ transition

26.3 Block Diagram

Figure 26-1. Full-Duplex SPI Master Slave Interconnection



26.4 Signal Description

Signal Name	Type	Description
PAD[3:0]	Digital I/O	General SERCOM pins

Refer to "I/O Multiplexing and Considerations" on page 10 for details on the pin mapping for this peripheral. One signal can be mapped to one of several pins.

26.5 Product Dependencies

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

26.5.1 I/O Lines

Using the SERCOM's I/O lines requires the I/O pins to be configured using port configuration (PORT). Refer to "PORT" on page 375 for details.

When the SERCOM is configured for SPI operation, the pins should be configured according to Table 26-1. If the receiver is disabled, the data input pin can be used for other purposes. In master mode the slave select line ($\overline{SS}$) is hardware controlled when Master Slave Select Enable (CTRLB.MSSEN) is set to one.

Table 26-1. SPI Pin Configuration

Pin	Master SPI	Slave SPI
MOSI	Output	Input
MISO	Input	Output
SCK	Output	Input
$\overline{SS}$	Output (CTRLB.MSSEN=1)	Input

The combined configuration of PORT and the Data In/Data Out and Data Out Pinout bit groups in Control A register will define the physical position of the SPI signals in Table 26-1.

26.5.2 Power Management

The SPI can continue to operate in any sleep mode. The SPI interrupts can be used to wake up the device from sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

26.5.3 Clocks

The SERCOM bus clock (CLK_SERCOMx_APB) can be enabled and disabled in the Power Manager, and the default state of CLK_SERCOMx_APB can be found in the Peripheral Clock Masking section in the "PM – Power Manager" on page 107.

A generic clock (GCLK_SERCOMx_CORE) is required to clock the SPI. This clock must be configured and enabled in the Generic Clock Controller before using the SPI. Refer to "GCLK – Generic Clock Controller" on page 85 for details.

This generic clock is asynchronous to the bus clock (CLK_SERCOMx_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 483 for further details.

26.5.4 DMA

The DMA request lines are connected to the DMA controller (DMAC). Using the SERCOM DMA requests, requires the DMA controller to be configured first. Refer to "DMAC – Direct Memory Access Controller" on page 268 for details.

26.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the SPI, interrupts requires the Interrupt Controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

26.5.6 Events

Not applicable.

26.5.7 Debug Operation

When the CPU is halted in debug mode, the SPI continues normal operation. If the SPI is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging. The SPI can be forced to halt operation during debugging. Refer to the Debug Control (DBGCTRL) register for details.

26.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Interrupt Flag Clear and Status register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode, all write-protection is automatically disabled.

Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

26.5.9 Analog Connections

Not applicable.

26.6 Functional Description

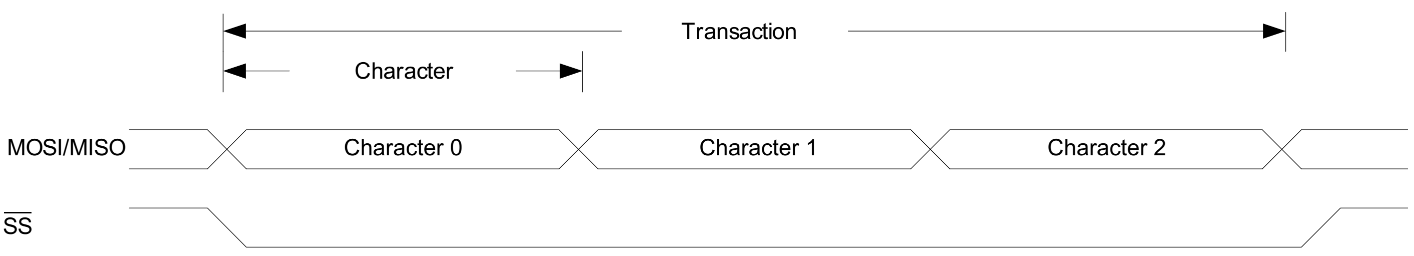
26.6.1 Principle of Operation

The SPI is a high-speed synchronous data transfer interface. It allows fast communication between the device and peripheral devices.

The SPI can operate as master or slave. As master, the SPI initiates and controls all data transactions. The SPI is single buffered for transmitting and double buffered for receiving. When transmitting data, the Data register can be loaded with the next character to be transmitted while the current transmission is in progress. For receiving, this means that the data is transferred to the two-level receive buffer upon reception, and the receiver is ready for a new character.

The SPI transaction format is shown in Figure 26-2, where each transaction can contain one or more characters. The character size is configurable, and can be either 8 or 9 bits.

Table 26-2. SPI Transaction Format



The SPI master must initiate a transaction by pulling low the slave select line ($\overline{SS}$) of the desired slave. The master and slave prepare data to be sent in their respective shift registers, and the master generates the serial clock on the SCK line. Data are always shifted from master to slave on the master output, slave input line (MOSI), and from slave to master on the master input, slave output line (MISO). The master signals the end of the transaction by pulling the SS line high.

As each character is shifted out from the master, another character is shifted in from the slave.

26.6.2 Basic Operation

26.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the SPI is disabled (CTRL.ENABLE is zero):

- Control A register (CTRLA), except Enable (CTRLA.ENABLE) and Software Reset (CTRLA.SWRST)
- Control B register (CTRLB), except Receiver Enable (CTRLB.RXEN)
- Baud register (BAUD)
- Address register (ADDR)

Any writes to these registers when the SPI is enabled or is being enabled (CTRLA.ENABLE is one) will be discarded. Writes to these registers while the SPI is being disabled will be completed after the disabling is complete.

Enable-protection is denoted by the Enable-Protection property in the register description.

Before the SPI is enabled, it must be configured, as outlined by the following steps:

- SPI mode in master or slave operation must be selected by writing 0x2 or 0x3 to the Operating Mode bit group in the Control A register (CTRLA.MODE)
- Transfer mode must be selected by writing the Clock Polarity bit and the Clock Phase bit in the Control A register (CTRLA.CPOL and CTRLA.CPHA)
- Transaction format must be selected by writing the Frame Format bit group in the Control A register (CTRLA.FORM)
- SERCOM pad to use for the receiver must be selected by writing the Data In Pinout bit group in the Control A register (CTRLA.DIPO)
- SERCOM pads to use for the transmitter, slave select and serial clock must be selected by writing the Data Out Pinout bit group in the Control A register (CTRLA.DOPO)
- Character size must be selected by writing the Character Size bit group in the Control B register (CTRLB.CHSIZE)
- Data direction must be selected by writing the Data Order bit in the Control A register (CTRLA.DORD)
- If the SPI is used in master mode, the Baud register (BAUD) must be written to generate the desired baud rate
- If the SPI is used in master mode and Hardware SS control is required, the Master Slave Select Enable bit in CTRLB register (CTRLB.MSSEN) should be set to 1.
- The receiver can be enabled by writing a one to the Receiver Enable bit in the Control B register (CTRLB.RXEN)

26.6.2.2 Enabling, Disabling and Resetting

The SPI is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The SPI is disabled by writing a zero to CTRLA.ENABLE.

The SPI is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the SPI, except DBGCTRL, will be reset to their initial state, and the SPI will be disabled. Refer to CTRLA for details.

26.6.2.3 Clock Generation

In SPI master operation (CTRLA.MODE is 0x3), the serial clock (SCK) is generated internally using the SERCOM baud-rate generator. When used in SPI mode, the baud-rate generator is set to synchronous mode, and the 8-bit Baud register (BAUD) value is used to generate SCK, clocking the shift register. Refer to "Clock Generation – Baud-Rate Generator" on page 430 for more details.

In SPI slave operation (CTRLA.MODE is 0x2), the clock is provided by an external master on the SCK pin. This clock is used to directly clock the SPI shift register.

26.6.2.4 Data Register

The SPI Transmit Data register (TxDATA) and SPI Receive Data register (RxDATA) share the same I/O address, referred to as the SPI Data register (DATA). Writing the DATA register will update the Transmit Data register. Reading the DATA register will return the contents of the Receive Data register.

26.6.2.5 SPI Transfer Modes

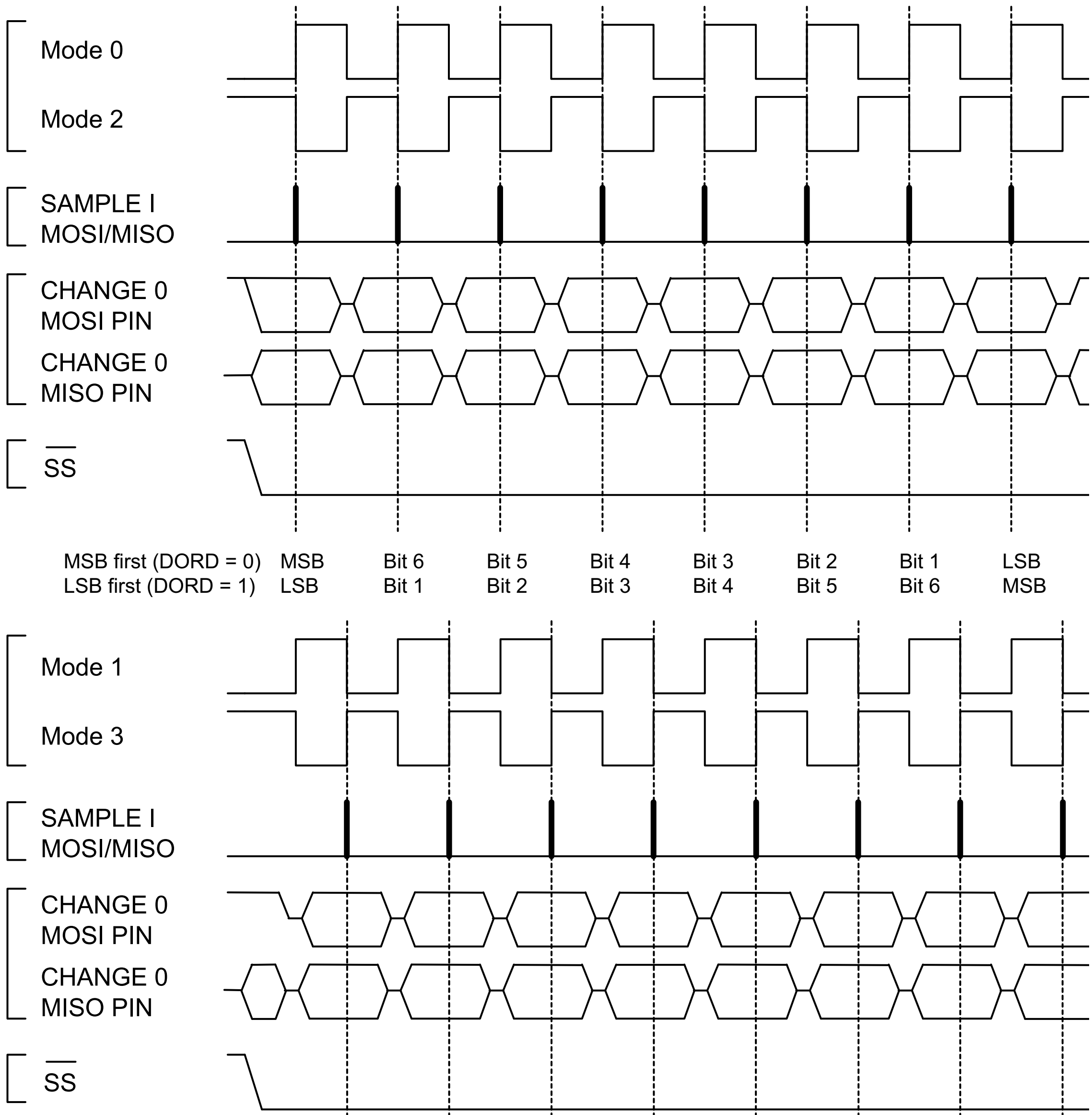
There are four combinations of SCK phase and polarity with respect to the serial data. The SPI data transfer modes are shown in Table 26-3 and Figure 26-2. SCK phase is selected by the Clock Phase bit in the Control A register (CTRLA.CPHA). SCK polarity is selected by the Clock Polarity bit in the Control A register (CTRLA.CPOL). Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for the data signals to stabilize.

Table 26-3. SPI Transfer Modes

Mode	CPOL	CPHA	Leading Edge	Trailing Edge
0	0	0	Rising, sample	Falling, setup
1	0	1	Rising, setup	Falling, sample
2	1	0	Falling, sample	Rising, setup
3	1	1	Falling, setup	Rising, sample

Leading edge is the first clock edge in a clock cycle, while trailing edge is the second clock edge in a clock cycle.

Figure 26-2. SPI Transfer Modes



26.6.2.6 Transferring Data

Master

When configured as a master (CTRLA.MODE is 0x3), if Master Slave Select Enable (CTRLB.MSSEN) is set to zero the SS line can be located at any general purpose I/O pin, and must be configured as an output. When the SPI is ready for a data transaction, software must pull the SS line low. If Master Slave Enable Select (CTRLB.MSSEN) is set to one, hardware controls the SS line.

When writing a character to the Data register (DATA), the character will be transferred to the shift register when the shift register is empty. Once the contents of TxDATA have been transferred to the shift register, the Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) is set, and a new character can be written to DATA.

As each character is shifted out from the master, another character is shifted in from the slave. If the receiver is enabled (CTRLA.RXEN is one), the contents of the shift register will be transferred to the two-level receive buffer. The transfer takes place in the same clock cycle as the last data bit is shifted in, and the Receive Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) will be set. The received data can be retrieved by reading DATA.

When the last character has been transmitted and there is no valid data in DATA, the Transmit Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.TXC) is set. When the transaction is finished, the master must indicate this to the slave by pulling the $\overline{SS}$ line high. If Master Slave Select Enable (CTRLB.MSSEN) is set to zero, the software must pull the $\overline{SS}$ line high.

Slave

When configured as a slave (CTRLA.MODE is 0x2), the SPI interface will remain inactive, with the MISO line tri-stated as long as the $\overline{SS}$ pin is pulled high. Software may update the contents of DATA at any time, as long as the Data Register Empty flag in the Interrupt Status and Clear register (INTFLAG.DRE) is set.

When $\overline{SS}$ is pulled low and SCK is running, the slave will sample and shift out data according to the transaction mode set. When the contents of TxDATA have been loaded into the shift register, INTFLAG.DRE is set, and new data can be written to DATA. Similar to the master, the slave will receive one character for each character transmitted. On the same clock cycle as the last data bit of a character is received, the character will be transferred into the two-level receive buffer. The received character can be retrieved from DATA when Receive Complete interrupt flag (INTFLAG.RXC) is set.

When the master pulls the $\overline{SS}$ line high, the transaction is done and the Transmit Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.TXC) is set.

Once DATA is written, it takes up to three SCK clock cycles before the content of DATA is ready to be loaded into the shift register. When the content of DATA is ready to be loaded, this will happen on the next character boundary. As a consequence, the first character transferred in a SPI transaction will not be the content of DATA. This can be avoided by using the preloading feature.

Refer to "Preloading of the Slave Shift Register" on page 480.

When transmitting several characters in one SPI transaction, the data has to be written to DATA while there are at least three SCK clock cycles left in the current character transmission. If this criteria is not met, then the previous character received will be transmitted.

After the DATA register is empty, it takes three CLK_SERCOM_APB cycles for INTFLAG.DRE to be set.

26.6.2.7 Receiver Error Bit

The SPI receiver has one error bit: the Buffer Overflow bit (BUFOVF), which can be read from the Status register (STATUS). Upon error detection, the bit will be set until it is cleared by writing a one to it. The bit is also automatically cleared when the receiver is disabled.

There are two methods for buffer overflow notification. When the immediate buffer overflow notification bit (CTRLA.IBON) is set, STATUS.BUFOVF is set immediately upon buffer overflow. Software can then empty the receive FIFO by reading RxDATA until the receive complete interrupt flag (INTFLAG.RXC) goes low.

When CTRLA.IBON is zero, the buffer overflow condition travels with data through the receive FIFO. After the received data is read, STATUS.BUFOVF and INTFLAG.ERROR will be set along with INTFLAG.RXC, and RxDATA will be zero.

26.6.3 Additional Features

26.6.3.1 Address Recognition

When the SPI is configured for slave operation (CTRLA.MODE is 0x2) with address recognition (CTRLA.FORM is 0x2), the SERCOM address recognition logic is enabled. When address recognition is enabled, the first character in a transaction is checked for an address match. If there is a match, then the Receive Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set, the MISO output is enabled and the transaction is processed. If there is no match, the transaction is ignored.

If the device is in sleep mode, an address match can wake up the device in order to process the transaction. If the address does not match, then the complete transaction is ignored. If a 9-bit frame format is selected, only the lower 8 bits of the shift register are checked against the Address register (ADDR).

Refer to “Address Match and Mask” on page 433 for further details.

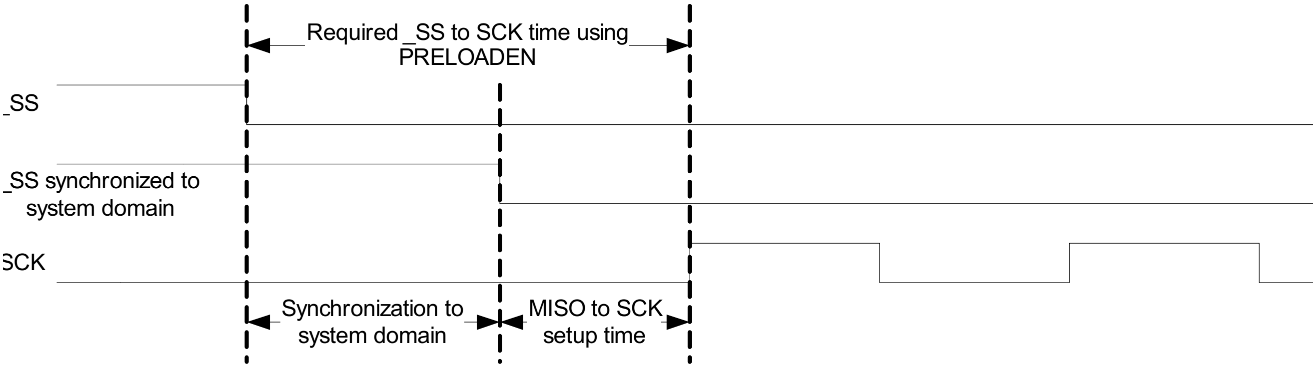
26.6.3.2 Preloading of the Slave Shift Register

When starting a transaction, the slave will first transmit the contents of the shift register before loading new data from DATA. The first character sent can be either the reset value of the shift register (if this is the first transmission since the last reset) or the last character in the previous transmission. Preloading can be used to preload data to the shift register while $\overline{SS}$ is high and eliminate sending a dummy character when starting a transaction.

In order to guarantee enough set-up time before the first SCK edge, enough time must be given between $\overline{SS}$ going low and the first SCK sampling edge, as shown in Figure 26-3.

Preloading is enabled by setting the Slave Data Preload Enable bit in the Control B register (CTRLB.PLOADEN).

Figure 26-3. Timing Using Preloading

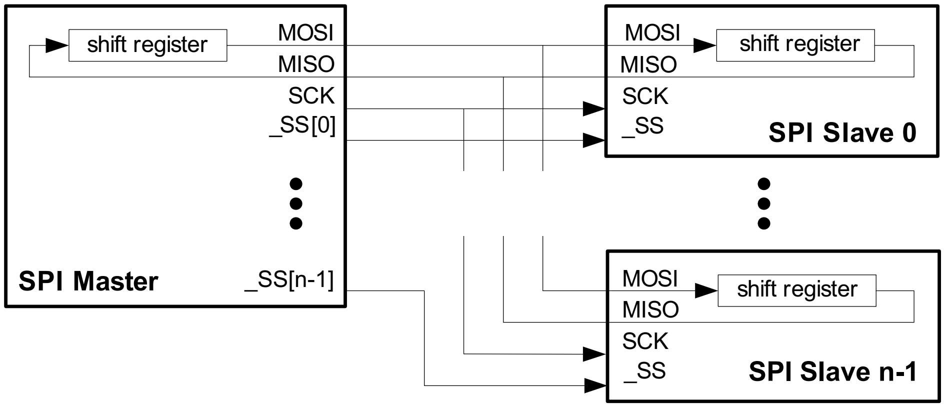


Only one data character written to DATA will be preloaded into the shift register while the synchronized $\overline{SS}$ signal (see Figure 26-3) is high. The next character written to DATA before $\overline{SS}$ is pulled low will be stored in DATA until transfer begins. If the shift register is not preloaded, the current contents of the shift register will be shifted out.

26.6.3.3 Master with Several Slaves

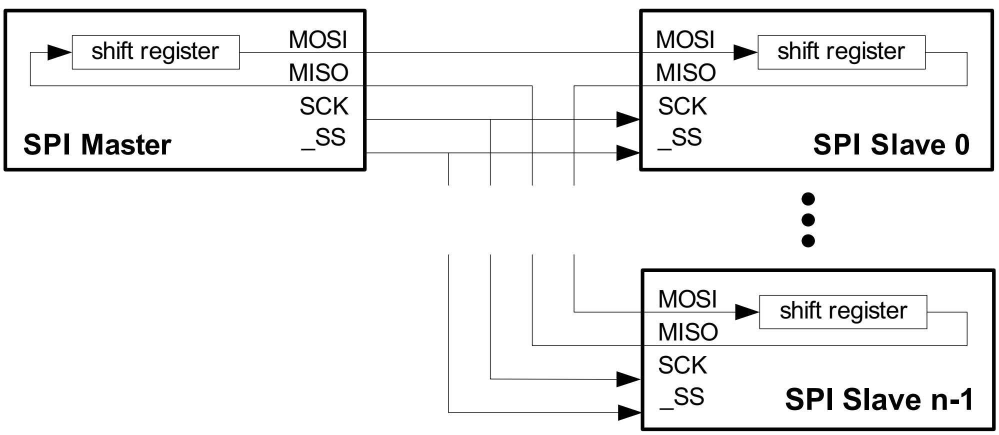
Master with multiple slaves in parallel feature is available only when Master Slave Select Enable (CTRLB.MSSEN) is set to zero and hardware $\overline{SS}$ control is disabled. If the bus consists of several SPI slaves, an SPI master can use general purpose I/O pins to control the $\overline{SS}$ line to each of the slaves on the bus, as shown in Figure 26-4. In this configuration, the single selected SPI slave will drive the tri-state MISO line.

Figure 26-4. Multiple Slaves in Parallel



An alternate configuration is shown in Figure 26-5. In this configuration, all n attached slaves are connected in series. A common $\overline{SS}$ line is provided to all slaves, enabling them simultaneously. The master must shift n characters for a complete transaction. Depending on the Master Slave Select Enable bit (CTRLB.MSSEN), $\overline{SS}$ line is controlled either by hardware or by user software and normal GPIO

Figure 26-5. Multiple Slaves in Series



26.6.3.4 Loop-back Mode

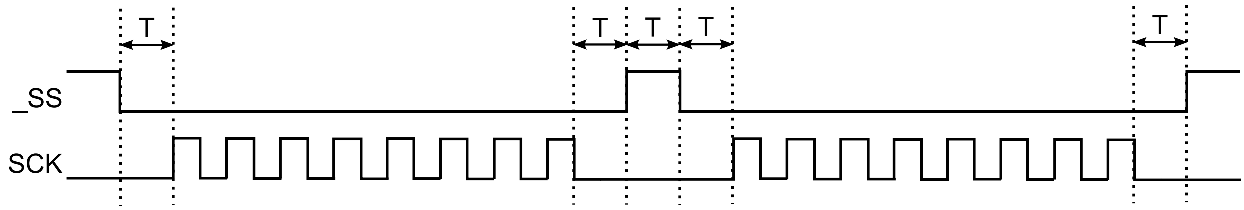
By configuring the Data In Pinout (CTRLA.DIPO) and Data Out Pinout (CTRLA.DOPO) to use the same data pins for transmit and receive, loop-back is achieved. The loop-back is through the pad, so the signal is also available externally.

26.6.3.5 Hardware Controlled $\overline{SS}$

In master mode, a single $\overline{SS}$ chip select can be controlled by hardware by setting the Master Slave Select Enable (CTRLB.MSSEN) bit to one. In this mode, the $\overline{SS}$ pin is driven low for a minimum of one baud cycle before transmission begins, and stays low for a minimum of one baud cycle after transmission completes. If back-to-back frames are transmitted, the $\overline{SS}$ pin will always be driven high for a minimum of one baud cycle between frames.

In Figure 26-6, the time T is between one and two baud cycles depending on the SPI transfer mode.

Figure 26-6. Hardware Controlled $\overline{SS}$



When MSSEN is set to zero, the $\overline{SS}$ pin(s) is/are controlled by user software and normal GPIO.

26.6.3.6 Slave Select Low Detection

In slave mode the SPI is capable of waking the CPU when the slave select ($\overline{SS}$) goes low. When the Slave Select Low Detect is enabled (CTRLB.SSDE=1), a high to low transition will set the Slave Select Low interrupt flag (INTFLAG.SSL) and the device will wake if applicable.

26.6.4 DMA, Interrupts and Events

Table 26-4. Module Request for SERCOM SPI

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Data Register Empty	x			x	When data is written
Transmit Complete	x				
Receive Complete	x			x	When data is read
Slave Select low	x				
Error	x				

26.6.4.1 DMA Operation

The SPI generates the following DMA requests:

- Data received (RX): The request is set when data is available in the receive FIFO. The request is cleared when DATA is read.
- Data transmit (TX): The request is set when the transmit buffer (TX DATA) is empty. The request is cleared when DATA is written.

26.6.4.2 Interrupts

The SPI has the following interrupt sources:

- Error
- Slave Select Low
- Receive Complete
- Transmit Complete
- Data Register Empty

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the SPI is reset. See the register description for details on how to clear interrupt flags.

The SPI has one common interrupt request line for all the interrupt sources. The user must read INTFLAG to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

For details on clearing interrupt flags, refer to INTFLAG.

26.6.4.3 Events

Not applicable.

26.6.5 Sleep Mode Operation

During master operation, the generic clock will continue to run in idle sleep mode. If the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY) is one, the GCLK_SERCOM_CORE will also be enabled in standby sleep mode. Any interrupt can wake up the device.

If CTRLA.RUNSTDBY is zero during master operation, GLK_SERCOMx_CORE will be disabled when the ongoing transaction is finished. Any interrupt can wake up the device.

During slave operation, writing a one to CTRLA.RUNSTDBY will allow the Receive Complete interrupt to wake up the device.

If CTRLA.RUNSTDBY is zero during slave operation, all reception will be dropped, including the ongoing transaction.

26.6.6 Synchronization

Due to the asynchronicity between CLK_SERCOMx_APB and GCLK_SERCOMx_CORE, some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the corresponding Synchronization Busy bit in the Synchronization Busy register (SYNCBUSY) will be set immediately, and cleared when synchronization is complete.

If an operation that requires synchronization is executed while the corresponding SYNCBUSY bit is one, a peripheral bus error is generated.

The following bits need synchronization when written:

- Software Reset bit in the Control A register (CTRLA.SWRST). SYNCBUSY.SWRST is set to one while synchronization is in progress.
- Enable bit in the Control A register (CTRLA.ENABLE). SYNCBUSY.ENABLE is set to one while synchronization is in progress.
- Receiver Enable bit in the Control B register (CTRLB.RXEN). SYNCBUSY.CTRLB is set to one while synchronization is in progress.

CTRLB.RXEN behaves somewhat differently than described above. Refer to CTRLB for details.

Write-synchronization is denoted by the Write-Synchronized property in the register description.

26.7 Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
0x01		15:8								IBON
0x02		23:16			DIPO[1:0]				DOPO[1:0]	
0x03		31:24		DORD	CPOL	CPHA	FORM[3:0]			
0x04	CTRLB	7:0		PLOADEN				CHSIZE[2:0]		
0x05		15:8	AMODE[1:0]		MSSEN				SSDE	
0x06		23:16							RXEN	
0x07		31:24								
0x08	Reserved									
0x09	Reserved									
0x0A	Reserved									
0x0B	Reserved									
0x0C	BAUD	7:0	BAUD[7:0]							
0x0D	Reserved									
0x0E	Reserved									
0x0F	Reserved									
0x10	Reserved									
0x11	Reserved									
0x12	Reserved									
0x13	Reserved									
0x14	INTENCLR	7:0	ERROR				SSL	RXC	TXC	DRE
0x15	Reserved									
0x16	INTENSET	7:0	ERROR				SSL	RXC	TXC	DRE
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR				SSL	RXC	TXC	DRE
0x19	Reserved									
0x1A	STATUS	7:0						BUFOVF		
0x1B		15:8								
0x1C	SYNCBUSY	7:0						CTRLB	ENABLE	SWRST
0x1D		15:8								
0x1E		23:16								
0x1F		31:24								
0x20	Reserved									
0x21	Reserved									
0x22	Reserved									
0x23	Reserved									

Offset	Name	Bit Pos.								
0x24	ADDR	7:0	ADDR[7:0]							
0x25		15:8								
0x26		23:16	ADDRMASK[7:0]							
0x27		31:24								
0x28	DATA	7:0	DATA[7:0]							
0x29		15:8								DATA[8]
0x2A	Reserved									
0x2B	Reserved									
0x2C	Reserved									
0x2D	Reserved									
0x2E	Reserved									
0x2F	Reserved									
0x30	DBGCTRL	7:0								DBGSTOP

26.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to "Register Access Protection" on page 475 for details.

Some registers require synchronization when read and/or written. Write-synchronization is denoted by the Write-Synchronized property in each individual register description. Refer to "Synchronization" on page 483 for details.

Some registers are enable-protected, meaning they can only be written when the USART is disabled. Enable-protection is denoted by the Enable-Protected property in each individual register description.

26.8.1 Control A

Name: CTRLA

Offset: 0x00

Reset: 0x00000000

Property: Write-Protected, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
		DORD	CPOL	CPHA	FORM[3:0]			
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			DIPO[1:0]				DOPO[1:0]	
Access	R	R	R/W	R/W	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
								IBON
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 31 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 30 – DORD: Data Order**
  This bit indicates the data order when a character is shifted out from the Data register.

  0: MSB is transferred first.

  1: LSB is transferred first.

  This bit is not synchronized.

- **Bit 29 – CPOL: Clock Polarity**
  In combination with the Clock Phase bit (CPHA), this bit determines the SPI transfer mode.

  0: SCK is low when idle. The leading edge of a clock cycle is a rising edge, while the trailing edge is a falling edge.

  1: SCK is high when idle. The leading edge of a clock cycle is a falling edge, while the trailing edge is a rising edge.

  This bit is not synchronized.

- **Bit 28 – CPHA: Clock Phase**
  In combination with the Clock Polarity bit (CPOL), this bit determines the SPI transfer mode.
  0: The data is sampled on a leading SCK edge and changed on a trailing SCK edge.
  1: The data is sampled on a trailing SCK edge and changed on a leading SCK edge.
  This bit is not synchronized.

Table 26-5. SPI Transfer Modes

Mode	CPOL	CPHA	Leading Edge	Trailing Edge
0x0	0	0	Rising, sample	Falling, change
0x1	0	1	Rising, change	Falling, sample
0x2	1	0	Falling, sample	Rising, change
0x3	1	1	Falling, change	Rising, sample

- **Bits 27:24 – FORM[3:0]: Frame Format**
  Table 26-6 shows the various frame formats supported by the SPI. When a frame format with address is selected, the first byte received is checked against the ADDR register.

Table 26-6. Frame Format

FORM[3:0]	Name	Description
0x0	SPI	SPI frame
0x1	-	Reserved
0x2	SPI_ADDR	SPI frame with address
0x3-0xF	-	Reserved

- **Bits 23:22 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 21:20 – DIPO[1:0]: Data In Pinout**
  These bits define the data in (DI) pad configurations.
  In master operation, DI is MISO.
  In slave operation, DI is MOSI.
  These bits are not synchronized.

Table 26-7. Data In Pinout

DIPO[1:0]	Name	Description
0x0	PAD[0]	SERCOM PAD[0] is used as data input
0x1	PAD[1]	SERCOM PAD[1] is used as data input
0x2	PAD[2]	SERCOM PAD[2] is used as data input
0x3	PAD[3]	SERCOM PAD[3] is used as data input

- **Bits 19:18 – Reserved**
These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 17:16 – DOPO: Data Out Pinout**
This bit defines the available pad configurations for data out (DO) and the serial clock (SCK). In slave operation, the slave select line (_SS) is controlled by DOPO, while in master operation the _SS line is controlled by the port configuration.
In master operation, DO is MOSI.
In slave operation, DO is MISO.
These bits are not synchronized.

Table 26-8. Data Out Pinout

DOPO	DO	SCK	Slave_SS	Master_SS
0x0	PAD[0]	PAD[1]	PAD[2]	System configuration
0x1	PAD[2]	PAD[3]	PAD[1]	System configuration
0x2	PAD[3]	PAD[1]	PAD[2]	System configuration
0x3	PAD[0]	PAD[3]	PAD[1]	System configuration

- **Bits 15:9 – Reserved**
These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 8 – IBON: Immediate Buffer Overflow Notification**
This bit controls when the buffer overflow status bit (STATUS.BUFOVF) is asserted when a buffer overflow occurs.
0: STATUS.BUFOVF is asserted when it occurs in the data stream.
1: STATUS.BUFOVF is asserted immediately upon buffer overflow.
This bit is not synchronized.

- **Bit 7 – RUNSTDBY: Run In Standby**
This bit defines the functionality in standby sleep mode.
These bits are not synchronized.

Table 26-9. Run In Standby Configuration

RUNSTDBY	Slave	Master
0x0	Disabled. All reception is dropped, including the ongoing transaction.	Generic clock is disabled when ongoing transaction is finished. All interrupts can wake up the device.
0x1	Wake on Receive Complete interrupt.	Generic clock is enabled while in sleep modes. All interrupts can wake up the device.

- **Bits 6:5 – Reserved**
These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:2 – MODE: Operating Mode**
These bits must be written to 0x2 or 0x3 to select the SPI serial communication interface of the SERCOM.
0x2: SPI slave operation

0x3: SPI master operation

These bits are not synchronized.

- **Bit 1 – ENABLE: Enable**

  0: The peripheral is disabled or being disabled.

  1: The peripheral is enabled or being enabled.

  Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately and the Synchronization Enable Busy bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE is cleared when the operation is complete.

  This bit is not enable-protected.

- **Bit 0 – SWRST: Software Reset**

  0: There is no reset operation ongoing.

  1: The reset operation is ongoing.

  Writing a zero to this bit has no effect.

  Writing a one to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.

  Writing a one to CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.

  Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY. SWRST will both be cleared when the reset is complete.

  This bit is not enable-protected.

26.8.2 Control B

Name: CTRLB

Offset: 0x04

Reset: 0x00000000

Property: Write-Protected, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
							RXEN	
Access	R	R	R	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AMODE[1:0]		MSSEN				SSDE	
Access	R/W	R/W	R/W	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		PLOADEN				CHSIZE[2:0]		
Access	R	R/W	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:18 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 17 – RXEN: Receiver Enable**
  0: The receiver is disabled or being enabled.

  1: The receiver is enabled or it will be enabled when SPI is enabled.

  Writing a zero to this bit will disable the SPI receiver immediately. The receive buffer will be flushed, data from ongoing receptions will be lost and STATUS.BUFOVF will be cleared.

  Writing a one to CTRLB.RXEN when the SPI is disabled will set CTRLB.RXEN immediately. When the SPI is enabled, CTRLB.RXEN will be cleared, SYNCBUSY.CTRLB will be set and remain set until the receiver is enabled. When the receiver is enabled CTRLB.RXEN will read back as one.

  Writing a one to CTRLB.RXEN when the SPI is enabled will set SYNCBUSY.CTRLB, which will remain set until the receiver is enabled, and CTRLB.RXEN will read back as one.

  This bit is not enable-protected.

- **Bit 16 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bits 15:14 – AMODE: Address Mode**
  These bits set the slave addressing mode when the frame format (CTRLA.FORM) with address is used. They are unused in master mode.

Table 26-10. Address Mode

AMODE[1:0]	Name	Description
0x0	MASK	ADDRMASK is used as a mask to the ADDR register
0x1	2_ADDRS	The slave responds to the two unique addresses in ADDR and ADDRMASK
0x2	RANGE	The slave responds to the range of addresses between and including ADDR and ADDRMASK. ADDR is the upper limit
0x3		Reserved

- **Bit 13 – MSSEN: Master Slave Select Enable**
  This bit enables hardware slave select (_SS) control.
  0: Hardware _SS control is disabled.
  1: Hardware _SS control is enabled.
- **Bits 12:10 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 9 – SSDE: Slave Select Low Detect Enable**
  This bit enables wake up when the slave select (_SS) pin transitions from high to low.
  0: _SS low detector is disabled.
  1: _SS low detector is enabled.
- **Bits 8:7 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 6 – PLOADEN: Slave Data Preload Enable**
  Setting this bit will enable preloading of the slave shift register when there is no transfer in progress. If the _SS line is high when DATA is written, it will be transferred immediately to the shift register.
- **Bits 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 2:0 – CHSIZE[2:0]: Character Size**

Table 26-11. Character Size

CHSIZE[2:0]	Name	Description
0x0	8BIT	8 bits
0x1	9BIT	9 bits
0x2-0x7		Reserved

26.8.3 Baud Rate

Name: BAUD

Offset: 0x0C

Reset: 0x00

Property: Write-Protected, Enable-Protected

Bit	7	6	5	4	3	2	1	0
	BAUD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:0 – BAUD: Baud Register**
  These bits control the clock generation, as described in the SERCOM "Clock Generation – Baud-Rate Generator" on page 430.

26.8.4 Interrupt Enable Clear

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Name: INTENCLR

Offset: 0x14

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ERROR				SSL	RXC	TXC	DRE
Access	R/W	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7– ERROR: Error Interrupt Enable**
  0: Error interrupt is disabled.
  1: Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

- **Bits 6:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3– SSL: Slave Select Low Interrupt Enable**
  0: Slave Select Low interrupt is disabled.
  1: Slave Select Low interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Slave Select Low Interrupt Enable bit, which disables the Slave Select Low interrupt.

- **Bit 2 – RXC: Receive Complete Interrupt Enable**
  0: Receive Complete interrupt is disabled.
  1: Receive Complete interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Receive Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

- **Bit 1 – TXC: Transmit Complete Interrupt Enable**
  0: Transmit Complete interrupt is disabled.
  1: Transmit Complete interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Transmit Complete Interrupt Enable bit, which disable the Transmit Complete interrupt.

- **Bit 0 – DRE: Data Register Empty Interrupt Enable**
  0: Data Register Empty interrupt is disabled.
  1: Data Register Empty interrupt is enabled.
  Writing a zero to this bit has no effect.

Writing a one to this bit will clear the Data Register Empty Interrupt Enable bit, which disables the Data Register Empty interrupt.

26.8.5 Interrupt Enable Set

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Name: INTENSET

Offset: 0x16

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ERROR				SSL	RXC	TXC	DRE
Access	R/W	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – ERROR: Error Interrupt Enable**
  0: Error interrupt is disabled.
  1: Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

- **Bits 6:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – SSL: Slave Select Low Interrupt Enable**
  0: Slave Select Low interrupt is disabled.
  1: Slave Select Low interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Slave Select Low Interrupt Enable bit, which enables the Slave Select Low interrupt.

- **Bit 2 – RXC: Receive Complete Interrupt Enable**
  0: Receive Complete interrupt is disabled.
  1: Receive Complete interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Receive Complete Interrupt Enable bit, which enables the Receive Complete interrupt.

- **Bit 1 – TXC: Transmit Complete Interrupt Enable**
  0: Transmit Complete interrupt is disabled.
  1: Transmit Complete interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Transmit Complete Interrupt Enable bit, which enables the Transmit Complete interrupt.

- **Bit 0 – DRE: Data Register Empty Interrupt Enable**
  0: Data Register Empty interrupt is disabled.
  1: Data Register Empty interrupt is enabled.
  Writing a zero to this bit has no effect.

Writing a one to this bit will set the Data Register Empty Interrupt Enable bit, which enables the Data Register Empty interrupt.

26.8.6 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x18

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	ERROR				SSL	RXC	TXC	DRE
Access	R/W	R	R	R	R/W	R	R/W	R
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – ERROR: Error**

  This flag is cleared by writing a one to it.

  This bit is set when any error is detected. Errors that will set this flag have corresponding status flags in the STATUS register. The BUFOVF error will set this interrupt flag.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the flag.

- **Bits 6:4 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – SSL: Slave Select Low**

  This flag is cleared by writing a one to it.

  This bit is set when a high to low transition is detected on the _SS pin in slave mode and Slave Select Low Detect (CTRLB.SSDE) is enabled.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the flag.

- **Bit 2 – RXC: Receive Complete**

  This flag is cleared by reading the Data (DATA) register or by disabling the receiver.

  This flag is set when there are unread data in the receive buffer. If address matching is enabled, the first data received in a transaction will be an address.

  Writing a zero to this bit has no effect.

  Writing a one to this bit has no effect.

- **Bit 1 – TXC: Transmit Complete**

  This flag is cleared by writing a one to it or by writing new data to DATA.

  In master mode, this flag is set when the data have been shifted out and there are no new data in DATA.

  In slave mode, this flag is set when the _SS pin is pulled high. If address matching is enabled, this flag is only set if the transaction was initiated with an address match.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the flag.

- **Bit 0 – DRE: Data Register Empty**

  This flag is cleared by writing new data to DATA.

  This flag is set when DATA is empty and ready for new data to transmit.

  Writing a zero to this bit has no effect.

Writing a one to this bit has no effect.

26.8.7 Status

Name: STATUS

Offset: 0x1A

Reset: 0x0000

Property: –

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						BUFOVF		
Access	R	R	R	R	R	R/W	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 15:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – BUFOVF: Buffer Overflow**
  Reading this bit before reading DATA will indicate the error status of the next character to be read.

  This bit is cleared by writing a one to the bit or by disabling the receiver.

  This bit is set when a buffer overflow condition is detected. An overflow condition occurs if the two-level receive buffer is full when the last bit of the incoming character is shifted into the shift register. All characters shifted into the shift registers before the overflow condition is eliminated by reading DATA will be lost.

  When set, the corresponding RxDATA will be 0.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear it.

- **Bits 1:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

26.8.8 Synchronization Busy

Name: SYNCBUSY

Offset: 0x1C

Reset: 0x00000000

Property:

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						CTRLB	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2– CTRLB: CTRLB Synchronization Busy**
  Writing CTRLB when the SERCOM is enabled requires synchronization. When written, the SYNC-BUSY.CTRLB bit will be set until synchronization is complete. If CTRLB is written while SYNCBUSY.CTRLB is asserted, an APB error will be generated.

  0: CTRLB synchronization is not busy.

  1: CTRLB synchronization is busy.

- **Bit 1 – ENABLE: SERCOM Enable Synchronization Busy**
  Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. When written, the SYNC-BUSY.ENABLE bit will be set until synchronization is complete.

  Writes to any register (except for CTRLA.SWRST) while enable synchronization is on-going will be discarded and an APB error will be generated.

  0: Enable synchronization is not busy.

  1: Enable synchronization is busy.

- **Bit 0 – SWRST: Software Reset Synchronization Busy**

  Resetting the SERCOM (CTRLA.SWRST) requires synchronization. When written, the SYNCBUSY.SWRST bit will be set until synchronization is complete.

  Writes to any register while synchronization is on-going will be discarded and an APB error will be generated.

  0: SWRST synchronization is not busy.

  1: SWRST synchronization is busy.

26.8.9 Address

Name: ADDR

Offset: 0x24

Reset: 0x00000000

Property: Write-Protected, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	ADDRMASK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	ADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:24 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 23:16 – ADDRMASK[7:0]: Address Mask**
  These bits hold the address mask when the transaction format (CTRLA.FORM) with address is used.

- **Bits 15:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – ADDR[7:0]: Address**
  These bits hold the address when the transaction format (CTRLA.FORM) with address is used.

26.8.10 Data

Name: DATA

Offset: 0x28

Reset: 0x0000

Property: –

Bit	15	14	13	12	11	10	9	8
								DATA[8]
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 8:0 – DATA[8:0]: Data**
  Reading these bits will return the contents of the receive data buffer. The register should be read only when the Receive Complete Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set.

  Writing these bits will write the transmit data buffer. This register should be written only when the Data Register Empty Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.DRE) is set.

26.8.11 Debug Control

Name: DBGCTRL

Offset: 0x30

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
								DBGSTOP
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – DBGSTOP: Debug Stop Mode**
  This bit controls the functionality when the CPU is halted by an external debugger.

  0: The baud-rate generator continues normal operation when the CPU is halted by an external debugger.

  1: The baud-rate generator is halted when the CPU is halted by an external debugger.

27. SERCOM I²C – SERCOM Inter-Integrated Circuit

27.1 Overview

The inter-integrated circuit (I²C) interface is one of the available modes in the serial communication interface (SERCOM). Refer to "SERCOM – Serial Communication Interface" on page 427 for details.

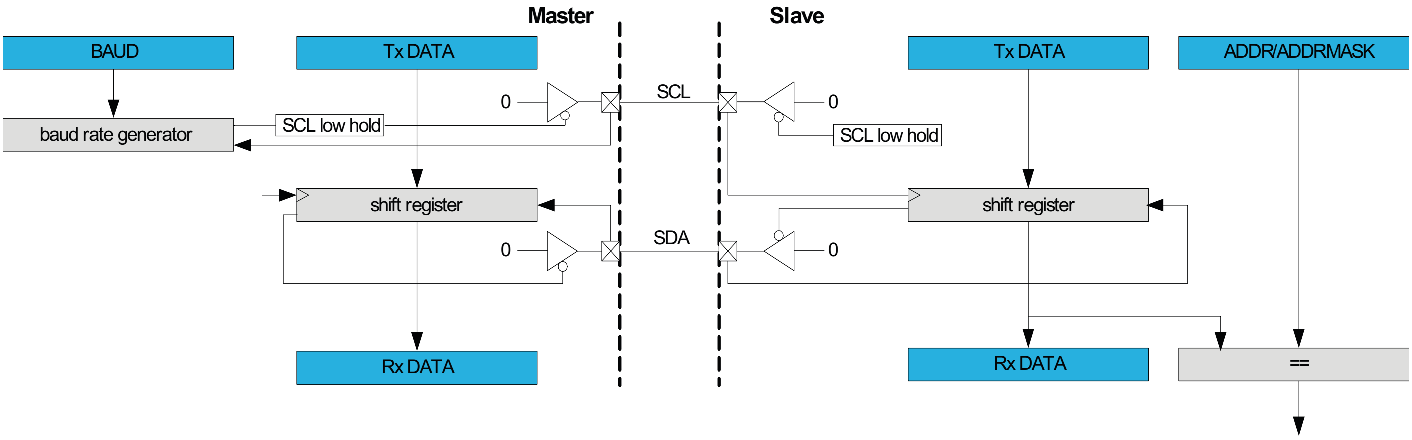
The I²C interface uses the SERCOM transmitter and receiver configured as shown in Figure 27-1. Fields shown in capital letters are registers accessible by the CPU, while lowercase fields are internal to the SERCOM. Each side, master and slave, depicts a separate I²C interface containing a shift register, a transmit buffer and a receive buffer. In addition, the I²C master uses the SERCOM baud-rate generator, while the I²C slave uses the SERCOM address match logic.

27.2 Features

- Master or slave operation
- Can be used with DMA
- Philips I²C compatible
- SMBus™ compatible
- PMBus compatible
- 100kHz and 400kHz, 1MHz support at low system clock frequencies
- Physical interface includes:
  - Slew-rate limited outputs
  - Filtered inputs
- Slave operation:
  - Operation in all sleep modes
  - Wake-up on address match
  - 7-bit and 10-bit Address match in hardware for:
    - Unique address and/or 7-bit general call address
    - Address range
    - Two unique addresses can be used with DMA

27.3 Block Diagram

Figure 27-1. I²C Single-Master Single-Slave Interconnection



27.4 Signal Description

Signal Name	Type	Description
PAD[0]	Digital I/O	SDA
PAD[1]	Digital I/O	SCL
PAD[2]	Digital I/O	SDA_OUT (4-wire)
PAD[3]	Digital I/O	SDC_OUT (4-wire)

Refer to “I/O Multiplexing and Considerations” on page 10 for details on the pin mapping for this peripheral. One signal can be mapped on several pins. Note that not all the pins are I^2C pins. Refer to for details on the pin type for each pin.

27.5 Product Dependencies

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

27.5.1 I/O Lines

Using the SERCOM’s I/O lines requires the I/O pins to be configured. Refer to “PORT” on page 375 for details.

27.5.2 Power Management

The I^2C will continue to operate in any sleep mode where the selected source clock is running. I^2C interrupts can be used to wake up the device from sleep modes. The events can trigger other operations in the system without exiting sleep modes. Refer to “PM – Power Manager” on page 107 for details on the different sleep modes.

27.5.3 Clocks

The SERCOM bus clock (CLK_SERCOMx_APB, where i represents the specific SERCOM instance number) is enabled by default, and can be enabled and disabled in the Power Manager. Refer to “PM – Power Manager” on page 107 for details.

The SERCOM bus clock (CLK_SERCOMx_APB) is enabled by default, and can be enabled and disabled in the Power Manager. Refer to “PM – Power Manager” on page 107 for details.

Two generic clocks are used by the SERCOM (GCLK_SERCOMx_CORE and GCLK_SERCOM_SLOW). The core clock (GCLK_SERCOMx_CORE) is required to clock the SERCOM while operating as a master, while the slow clock (GCLK_SERCOM_SLOW) is required only for certain functions. These clocks must be configured and enabled in the Generic Clock Controller (GCLK) before using the SERCOM. Refer to “GCLK – Generic Clock Controller” on page 85 for details.

These generic clocks are asynchronous to the SERCOM bus clock (CLK_SERCOMx_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to the “Synchronization” on page 524 section for further details.

27.5.4 DMA

The DMA request lines are connected to the DMA controller (DMAC). Using the SERCOM DMA requests, requires the DMA controller to be configured first. Refer to “DMAC – Direct Memory Access Controller” on page 268 for details.

27.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the I^2C interrupts requires the Interrupt Controller to be configured first. Refer to “Nested Vector Interrupt Controller” on page 23 for details.

27.5.6 Events

Not applicable.

27.5.7 Debug Operation

When the CPU is halted in debug mode, the I^2C interface continues normal operation. If the I^2C 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. The I^2C interface can be forced to halt operation during debugging.

Refer to the DBGCTRL register for details.

27.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Interrupt Flag Status and Clear register (INTFLAG)
- Status register (STATUS)
- Address register (ADDR)
- Data register (DATA)

Write-protection is denoted by the Write-Protected property in the register description.

Write-protection does not apply to accesses through en external debugger. Refer to“PAC – Peripheral Access Controller” on page 27 for details.

27.5.9 Analog Connections

Not applicable.

27.6 Functional Description

27.6.1 Principle of Operation

The I^2C interface uses two physical lines for communication:

- Serial Data Line (SDA) for packet transfer
- Serial Clock Line (SCL) for the bus clock

A transaction starts with the start condition, followed by a 7-bit address and a direction bit (read or write) sent from the I^2C master. The addressed I^2C slave will then acknowledge (ACK) the address, and data packet transactions can commence. Every 9-bit data packet consists of 8 data bits followed by a one-bit reply indicating whether the data was acknowledged or not. In the event that a data packet is not acknowledged (NACK), whether sent from the I^2C slave or master, it will be up to the I^2C master to either terminate the connection by issuing the stop condition, or send a repeated start if more data is to be transceived.

Figure 27-2 illustrates the possible transaction formats and Figure 27-3 explains the legend used.

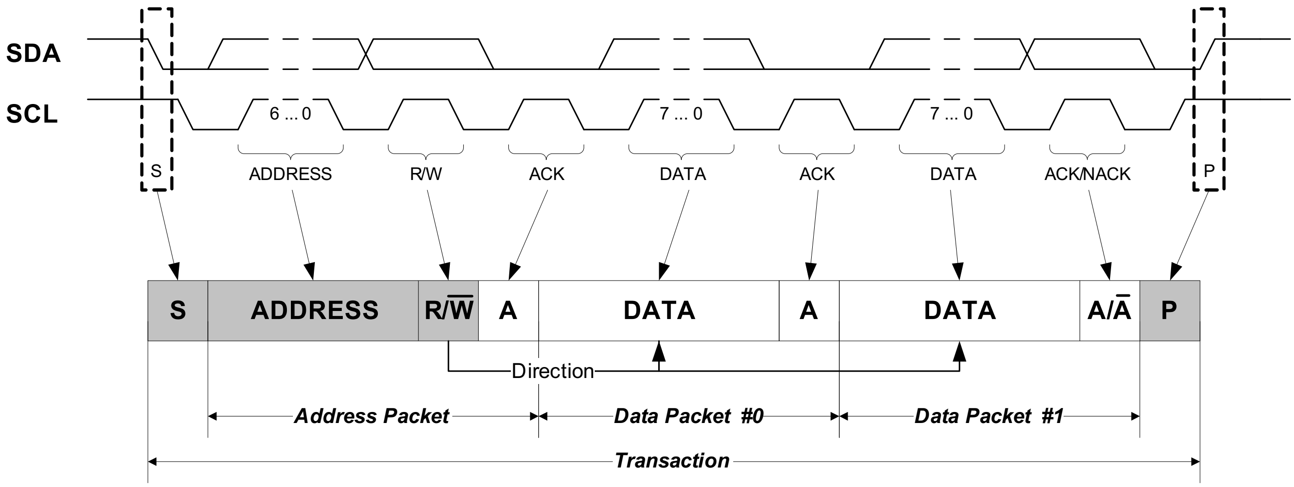


Figure 27-2. Basic I²C Transaction Diagram

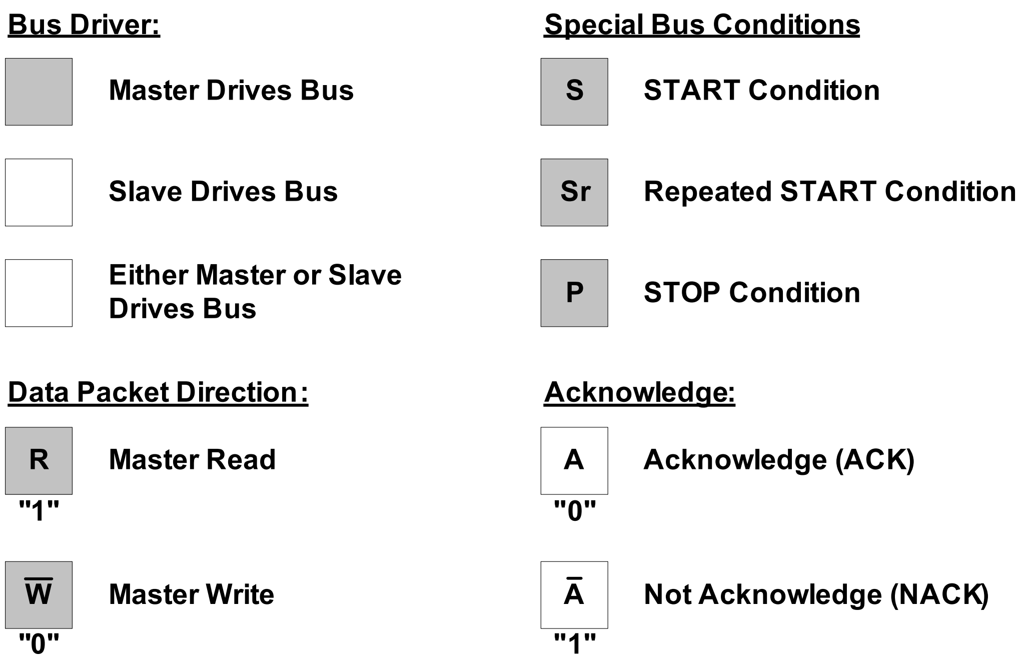


Figure 27-3. Transaction Diagram Syntax

27.6.2 Basic Operation

27.6.2.1 Initialization

The following registers are enable-protected, meaning they can be written only when the I²C interface is disabled (CTRLA.ENABLE is zero):

- Control A register (CTRLA), except Enable (CTRLA.ENABLE) and Software Reset (CTRLA.SWRST)
- Control B register (CTRLB), except Acknowledge Action (CTRLB.ACKACT) and Command (CTRLB.CMD)
- Baud Rate register (BAUD)
- Address register (ADDR) while in slave operation

Any writes to these bits or registers when the I^2C interface is enabled or is being enabled (CTRLA.ENABLE is one) will be discarded. Writes to these registers while the I^2C interface is being disabled will be completed after the disabling is complete.

Enable-protection is denoted by the Enable-Protection property in the register description.

Before the I^2C interface is enabled, it must be configured as outlined by the following steps:

I^2C mode in master or slave operation must be selected by writing 0x4 or 0x5 to the Operating Mode bit group in the Control A register (CTRLA.MODE)

- SCL low time-out can be enabled by writing to the SCL Low Time-Out bit in the Control A register (CTRLA.LOWTOUT)
- In master operation, the inactive bus time-out can be set in the Inactive Time-Out bit group in the Control A register (CTRLA.INACTOUT)
- Hold time for SDA can be set in the SDA Hold Time bit group in the Control A register (CTRLA.SDAHOLD)
- Smart operation can be enabled by writing to the Smart Mode Enable bit in the Control B register (CTRLB.SMEN)
- In slave operation, the address match configuration must be set in the Address Mode bit group in the Control B register (CTRLB.AMODE)
- In slave operation, the addresses must be set, according to the selected address configuration, in the Address and Address Mask bit groups in the Address register (ADDR.ADDR and ADDR.ADDRMASK)
- In master operation, the Baud Rate register (BAUD) must be written to generate the desired baud rate

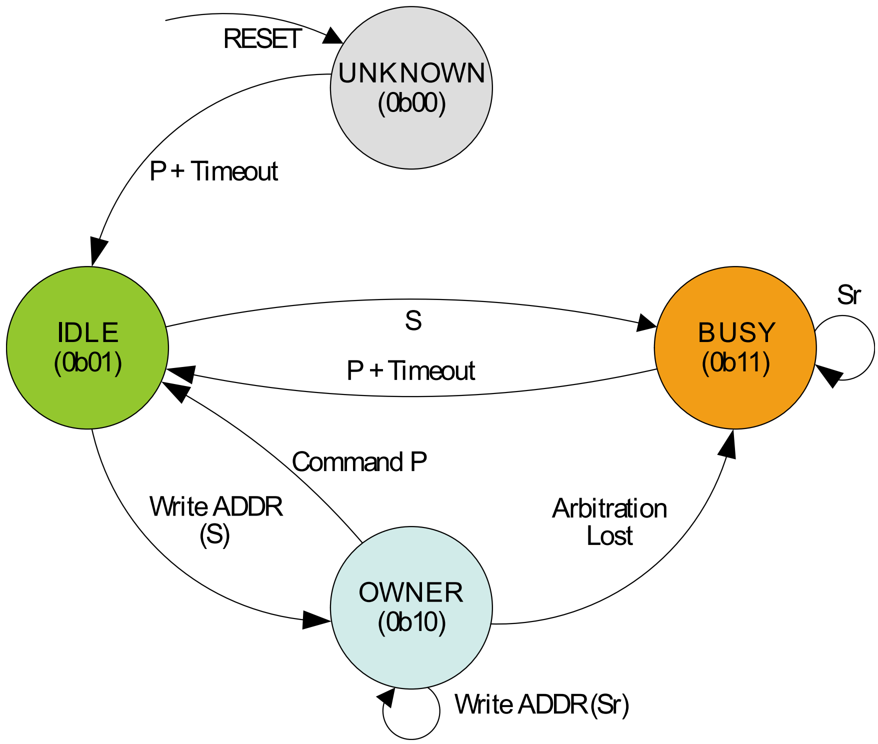
27.6.2.2 Enabling, Disabling and Resetting

The I^2C interface is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The I^2C interface is disabled by writing a zero to CTRLA.ENABLE. The I^2C interface is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the I^2C interface, except DBGCTRL, will be reset to their initial state, and the I^2C interface will be disabled. Refer to CTRLA for details.

27.6.2.3 I^2C Bus State Logic

The bus state logic includes several logic blocks that continuously monitor the activity on the I^2C bus lines in all sleep modes. The start and stop detectors and the bit counter are all essential in the process of determining the current bus state. The bus state is determined according to the state diagram shown in Figure 27-4. Software can get the current bus state by reading the Master Bus State bits in the Status register (STATUS.BUSSTATE). The value of STATUS.BUSSTATE in the figure is shown in binary.

Figure 27-4. Bus State Diagram



The bus state machine is active when the I²C master is enabled. After the I²C master has been enabled, the bus state is unknown. From the unknown state, the bus state machine can be forced to enter the idle state by writing to STATUS.BUSSTATE accordingly. However, if no action is taken by software, the bus state will become idle if a stop condition is detected on the bus. If the inactive bus time-out is enabled, the bus state will change from unknown to idle on the occurrence of a time-out. Note that after a known bus state is established, the bus state logic will not re-enter the unknown state from either of the other states.

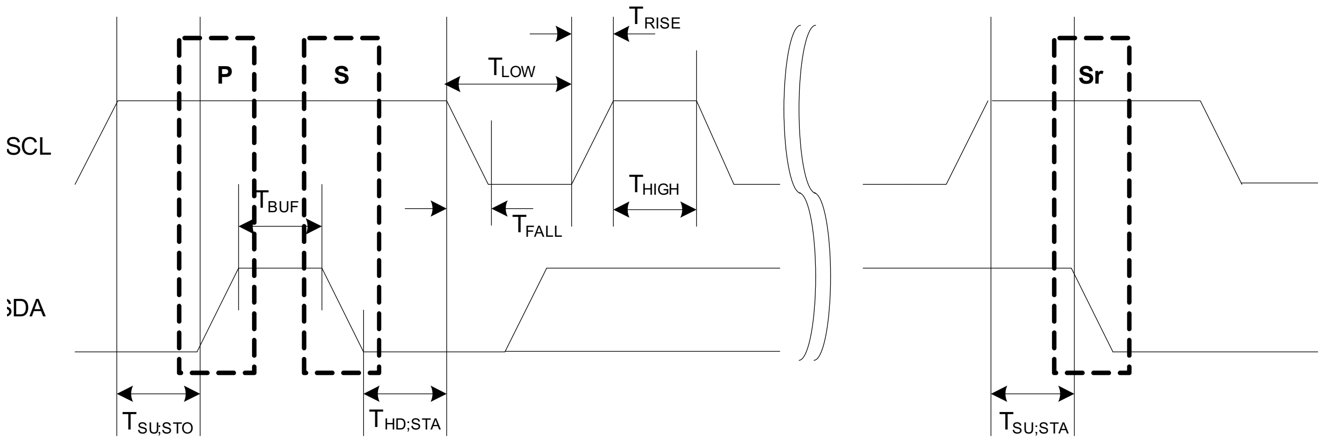
When the bus is idle it is ready for a new transaction. If a start condition is issued on the bus by another I²C master in a multimaster setup, the bus becomes busy until a stop condition is detected. The stop condition will cause the bus to re-enter the IDLE state. If the inactive bus time-out (SMBus) is enabled, the bus state will change from busy to idle on the occurrence of a time-out. If a start condition is generated internally by writing the Address bit group in the Address register (ADDR.ADDR) while in idle state, the owner state is entered. If the complete transaction was performed without interference, i.e., arbitration not lost, the I²C master is allowed to issue a stop condition, which in turn will cause a change of the bus state back to idle. However, if a packet collision is detected when in the owner state, the arbitration is assumed lost and the bus state becomes busy until a stop condition is detected.

A repeated start condition will change the bus state only if arbitration is lost while issuing a repeated start.

27.6.2.4 Clock Generation (Standard-mode, Fast-mode and Fast-mode Plus Transfers)

The Master I²C clock (SCL) frequency is determined by a number of factors. The low (T_{LOW}) and high ($T_{\_HIGH}$) times are determined by the Baud Rate register (BAUD), while the rise (T_{RISE}) and fall (T_{FALL}) times are determined by the bus topology. Because of the wired-AND logic of the bus, T_{FALL} will be considered as part of T_{LOW}. Likewise, T_{RISE} will be in a state between T_{LOW} and T_{HIGH} until a high state has been detected.

Figure 27-5. SCL Timing



The following parameters are timed using the SCL low time period. This comes from the Master Baud Rate Low bit group in the Baud Rate register (BAUD.BAUDLOW) when non-zero, or the Master Baud Rate bit group in the Baud Rate register (BAUD.BAUD) when BAUD.BAUDLOW is zero.

- T_{LOW} – Low period of SCL clock
- $T_{SU;STO}$ – Set-up time for stop condition
- T_{BUF} – Bus free time between stop and start conditions
- $T_{HD;STA}$ – Hold time (repeated) start condition
- $T_{SU;STA}$ – Set-up time for repeated start condition
- T_{HIGH} is timed using the SCL high time count from BAUD.BAUD
- T_{RISE} is determined by the bus impedance; for internal pull-ups. Refer to "Electrical Characteristics" on page 648 for details.
- T_{FALL} is determined by the open-drain current limit and bus impedance; can typically be regarded as zero. Refer to "Electrical Characteristics" on page 648 for details.

The SCL frequency is given by:

$$f_{SCL} = \frac{1}{T_{LOW} + T_{HIGH} + T_{RISE}}$$

When BAUD.BAUDLOW is zero, the BAUD.BAUD value is used to time both SCL high and SCL low. In this case the following formula will give the SCL frequency:

$$f_{SCL} = \frac{f_{GCLK}}{2(5 + BAUD) + f_{GCLK} T_{RISE}}$$

When BAUD.BAUDLOW is non-zero, the following formula is used to determine the SCL frequency:

$$f_{SCL} = \frac{f_{GCLK}}{10 + BAUD + BAUDLOW + f_{GCLK} T_{RISE}}$$

When BAUDLOW is non-zero, the following formula can be used to determine the SCL frequency:

$$f_{SCL} = \frac{f_{GCLK}}{10 + BAUD + BAUDLOW + f_{GCLK} T_{RISE}}$$

The following formulas can be used to determine the SCL T_{LOW} and T_{HIGH} times:

$$T_{low} = \frac{BAUD.BAUDLOW + 5}{f_{GCLK}}$$

$$T_{HIGH} = \frac{BAUD.BAUD + 5}{f_{GCLK}}$$

For Fast-mode Plus the nominal high to low SCL ratio is 1 to 2 and BAUD should be set accordingly. At a minimum, BAUD.BAUD and/or BAUD.BAUDLOW must be non-zero.

27.6.2.5 Master Clock Generation (High-speed mode Transfer)

For High-speed mode transfers, there is no SCL synchronization, so the SCL frequency is determined by the GCLK frequency and the High-speed BAUD setting. When HSBAUDLOW is zero, the HSBAUD value is used to time both SCL high and SCL low. In this case the following formula can be used to determine the SCL frequency.

$$f_{SCL} = \frac{f_{GCLK}}{2(1 + HSBAUD)}$$

When HSBAUDLOW is non-zero, the following formula can be used to determine the SCL frequency.

$$f_{SCL} = \frac{f_{GCLK}}{2 + HSBAUD + HSBAUDLOW}$$

For High-speed the nominal high to low SCL ratio is 1 to 2 and HSBAUD should be set accordingly. At a minimum, BAUD.BAUD and/or BAUD.BAUDLOW must be non-zero.

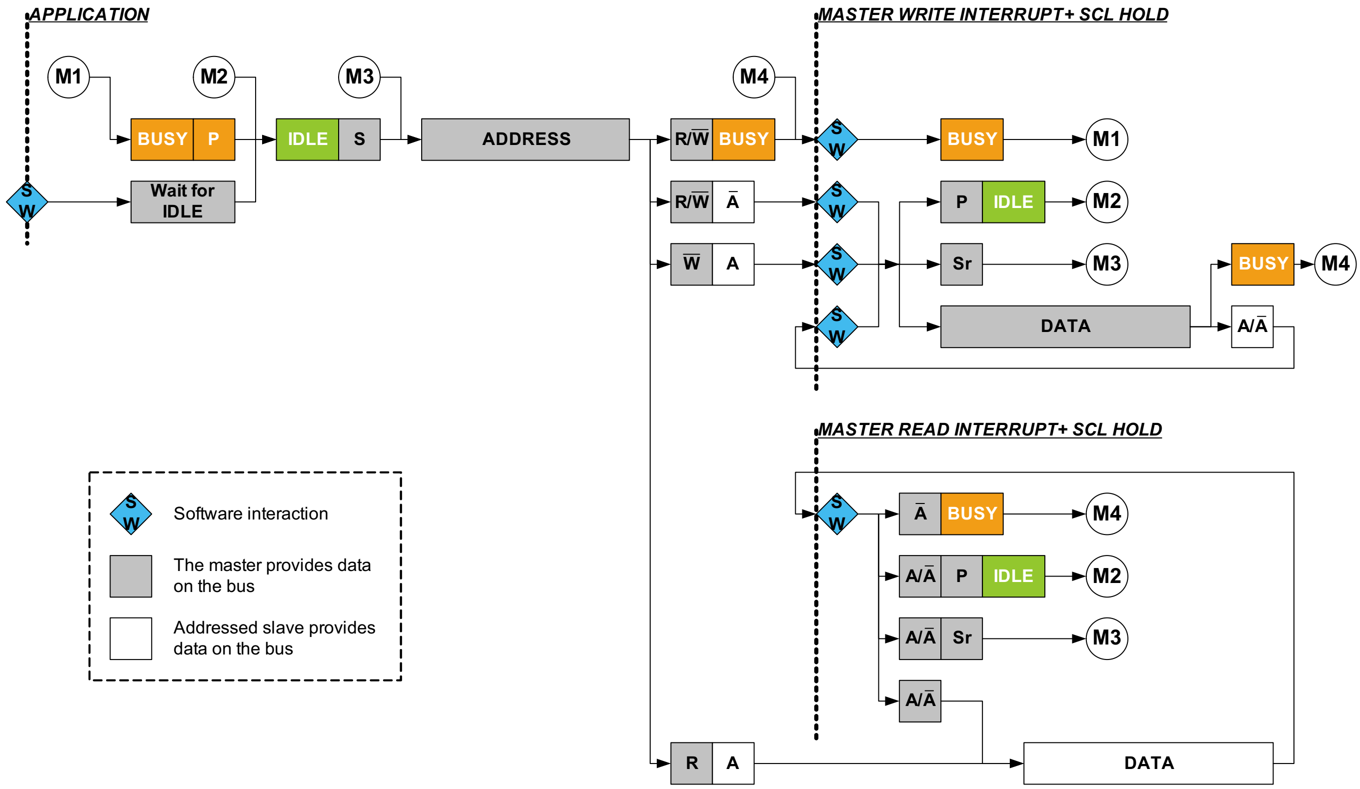
27.6.2.6 I²C Master Operation

The I²C master is byte-oriented and interrupt based. The number of interrupts generated is kept at a minimum by automatic handling of most events. Auto-triggering of operations and a special smart mode, which can be enabled by writing a one to the Smart Mode Enable bit in the Control A register (CTRLA.SMEN), are included to reduce software driver complexity and code size.

The I2C master has two interrupt strategies. When SCL Stretch Mode (CTRLA.SCLSM) is set to zero, SCL is stretched before or after the acknowledge bit . In this mode the I²C master operates according to the behavior diagram shown in Figure 27-6. The circles with a capital letter M followed by a number (M1, M2... etc.) indicate which node in the figure the bus logic can jump to based on software or hardware interaction.

This diagram is used as reference for the description of the I²C master operation throughout the document.

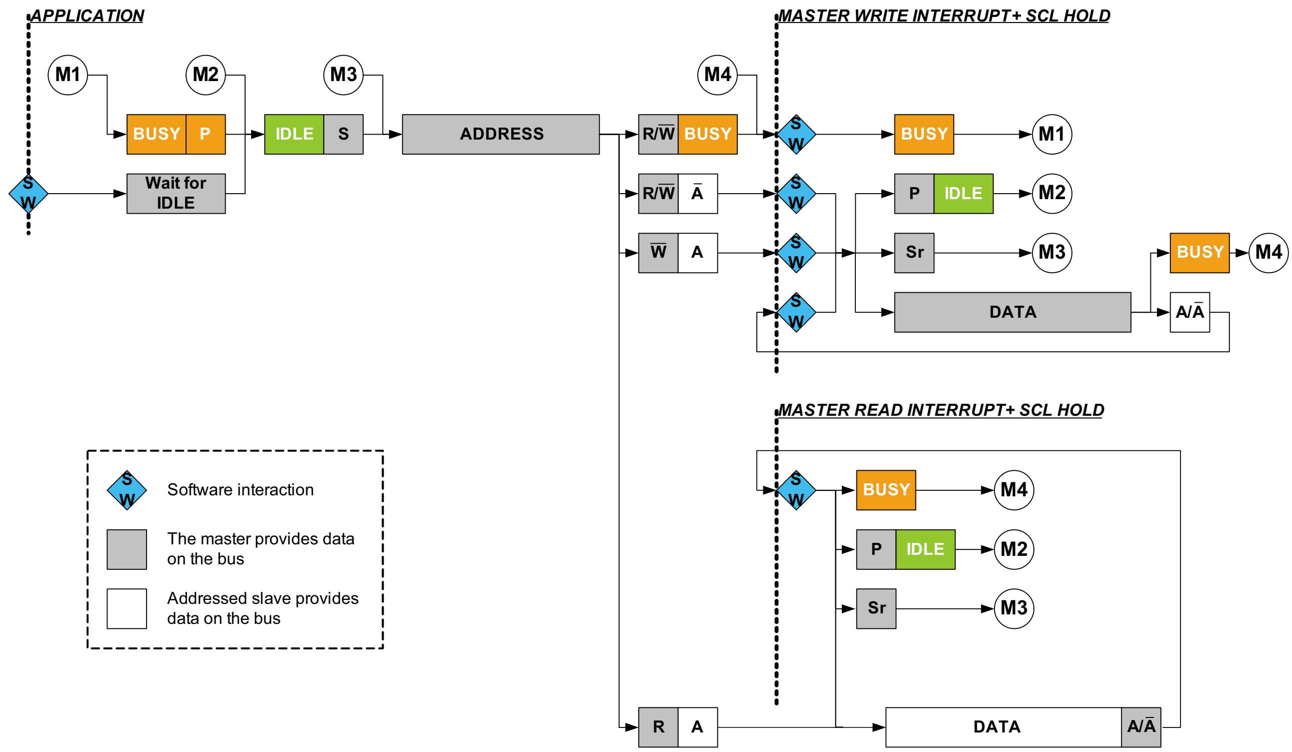
Figure 27-6. I²C Master Behavioral Diagram (SCLSM=0)



In the second strategy (SCLSM=1), interrupts only occur after the ACK bit as shown in Figure 27-7. This strategy can be used when it is not necessary to check DATA before acknowledging.

Note that setting SCLSM to 1 is required for High-speed mode.

Figure 27-7. I²C Master Behavioral Diagram (SCLSM=1)



Transmitting Address Packets

The I²C master starts a bus transaction by writing ADDR.ADDR with the I²C slave address and the direction bit. If the bus is busy, the I²C master will wait until the bus becomes idle before continuing the operation. When the bus is idle, the I²C master will issue a start condition on the bus. The I²C master will then transmit an address packet using the address written to ADDR.ADDR.

After the address packet has been transmitted by the I²C master, one of four cases will arise, based on arbitration and transfer direction.

Case 1: Arbitration lost or bus error during address packet transmission

If arbitration was lost during transmission of the address packet, the Master on Bus bit in the Interrupt Flag register (INTFLAG.MB) and the Arbitration Lost bit in the Status register (STATUS.ARBLOST) are both set. Serial data output to SDA is disabled, and the SCL is released, which disables clock stretching. In effect the I²C master is no longer allowed to perform any operation on the bus until the bus is idle again. A bus error will behave similarly to the arbitration lost condition. In this case, the MB interrupt flag and Master Bus Error bit in the Status register (STATUS.BUSERR) are both set in addition to STATUS.ARBLOST.

The Master Received Not Acknowledge bit in the Status register (STATUS.RXNACK) will always contain the last successfully received acknowledge or not acknowledge indication.

In this case, software will typically inform the application code of the condition and then clear the interrupt flag before exiting the interrupt routine. No other flags have to be cleared at this point, because all flags will be cleared automatically the next time the ADDR.ADDR register is written.

Case 2: Address packet transmit complete – No ACK received

If no I²C slave device responds to the address packet, then the INTFLAG.MB interrupt flag is set and STATUS.RXNACK is set. The clock hold is active at this point, preventing further activity on the bus.

The missing ACK response can indicate that the I²C slave is busy with other tasks or sleeping and, therefore, not able to respond. In this event, the next step can be either issuing a stop condition (recommended) or resending the address packet by using a repeated start condition. However, the reason for the missing acknowledge can be that an invalid I²C slave address has been used or that the I²C slave is for some reason disconnected or faulty. If using SMBus logic, the slave must ACK the address, and hence no action means the slave is not available on the bus.

Case 3: Address packet transmit complete – Write packet, Master on Bus set

If the I²C master receives an acknowledge response from the I²C slave, INTFLAG.MB is set and STATUS.RXNACK is cleared. The clock hold is active at this point, preventing further activity on the bus.

In this case, the software implementation becomes highly protocol dependent. Three possible actions can enable the I²C operation to continue. The three options are:

- The data transmit operation is initiated by writing the data byte to be transmitted into DATA.DATA.
- Transmit a new address packet by writing ADDR.ADDR. A repeated start condition will automatically be inserted before the address packet.
- Issue a stop condition, consequently terminating the transaction.

Case 4: Address packet transmit complete – Read packet, Slave on Bus set

If the I²C master receives an ACK from the I²C slave, the I²C master proceeds to receive the next byte of data from the I²C slave. When the first data byte is received, the Slave on Bus bit in the Interrupt Flag register (INTFLAG.SB) is set and STATUS.RXNACK is cleared. The clock hold is active at this point, preventing further activity on the bus.

In this case, the software implementation becomes highly protocol dependent. Three possible actions can enable the I²C operation to continue. The three options are:

- Let the I²C master continue to read data by first acknowledging the data received. This is automatically done when reading DATA.DATA if the smart mode is enabled.
- Transmit a new address packet.
- Terminate the transaction by issuing a stop condition.

An ACK or NACK will be automatically transmitted for the last two alternatives if smart mode is enabled. The Acknowledge Action bit in the Control B register (CTRLB.ACKACT) determines whether ACK or NACK should be sent.

Transmitting Data Packets

When an address packet with direction set to write has been successfully transmitted, INTFLAG.MB will be set and the I²C master can start transmitting data by writing to DATA.DATA. The I²C master transmits data via the I²C bus while continuously monitoring for packet collisions. If a collision is detected, the I²C master looses arbitration and STATUS.ARBLOST is set. If the transmit was successful, the I²C master automatically receives an ACK bit from the I²C slave and STATUS.RXNACK will be cleared. INTFLAG.MB will be set in both cases, regardless of arbitration outcome.

Testing STATUS.ARBLOST and handling the arbitration lost condition in the beginning of the I²C Master on Bus interrupt is recommended. This can be done, as there is no difference between handling address and data packet arbitration.

STATUS.RXNACK must be checked for each data packet transmitted before the next data packet transmission can commence. The I²C master is not allowed to continue transmitting data packets if a NACK is given from the I²C slave.

Receiving Data Packets (SCLSM=0)

When INTFLAG.SB is set, the I²C master will already have received one data packet. The I²C master must respond by sending either an ACK or NACK. Sending a NACK might not be successfully executed as arbitration can be lost during the transmission. In this case, a loss of arbitration will cause INTFLAG.SB to not be set on completion. Instead, INTFLAG.MB will be used to indicate a change in arbitration. Handling of lost arbitration is the same as for data bit transmission.

Receiving Data Packets (SCLSM=1)

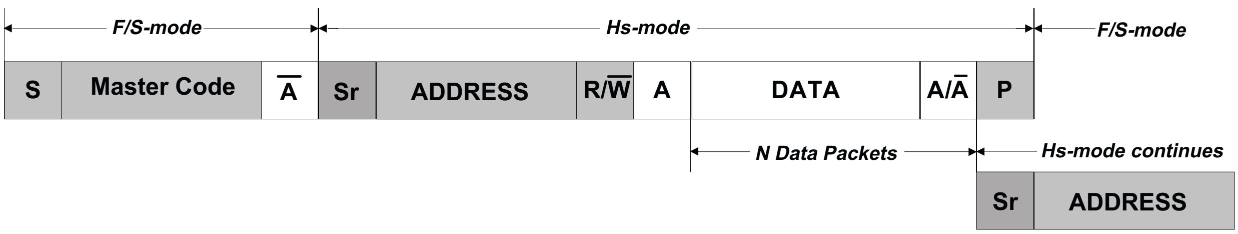
When INTFLAG.SB is set, the I²C master will already have received one data packet and transmitted the ACKACT bit. At this point the ACKACT must be set to the correct value for the next ACK bit, and the transaction can continue by reading DATA and issuing a command if not in smart mode.

High-speed Mode

High-speed transfers are a multi-step process as shown in Figure 27-8. First, a master code (0000 1nnn where nnn is a unique master code) is transmitted in Full-speed mode, followed by a NACK since no slave should acknowledge. Arbitration is performed only during the Full-speed Master Code phase. The master code is transmitted by writing the master code to the address register (ADDR) with the high-speed bit (ADDR.HS) written to zero.

After the Master Code and NACK have been transmitted, the master write interrupt will be asserted. At this point, the slave address can be written to the ADDR register with the ADDR.HS bit set to one. The master will then generate a repeated start followed by the slave address in High-speed mode. The bus will remain in High-speed mode until a stop is generated. If a repeated start is desired, the ADDR.HS bit must again be written to 1 along with the new address to be transmitted.

Figure 27-8. High Speed Transfer



Transmitting in High-speed mode requires the I2C master to be configured in High-speed mode (SPEED=0b10) and the SCL clock stretch mode (SCLSM) bit set to one.

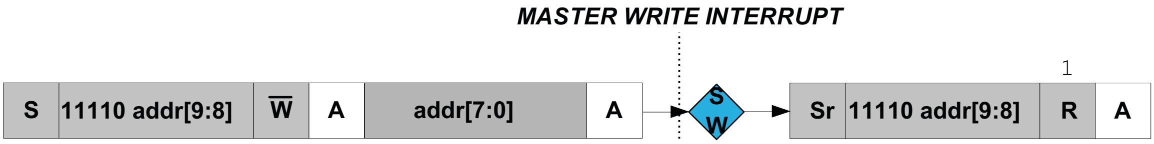
10-Bit Addressing

When 10-bit addressing is enabled (TENBITEN=1) and the ADDR register is written, the two address bytes will be transmitted as shown in Figure 27-9. The addressed slave acknowledges the two address bytes and the transaction continues. Regardless of whether the transaction is a read or write, the master must start by sending the 10-bit address with the read/write bit (ADDR.ADDR[0]) equal to zero.

If the master receives a NACK after the first byte, then the write interrupt flag will be raised and the NACK bit will be set. If the first byte is acknowledged by one or more slaves, then the master will proceed to transmit the second address byte and the master will first see the write interrupt flag after the second byte is transmitted.

If the transaction is a read, the 10-bit address transmission must be followed by a repeated start and the first 7 bits of the address with the read/write bit equal to 1.

Figure 27-9. 10-Bit Address Transmission for a Read Transaction



This implies the following procedure for a 10-bit read operation:

- Write ADDR.ADDR[10:1] with the 10-bit address. ADDR.TENBITEN must be set (can be written simultaneously with ADDR) and read/write bit (ADDR.ADDR[0]) equal to 0.
- When the master write interrupt is asserted, write ADDR[7:0] register to “11110 address[9:8] 1”. ADDR.TENBITEN must be cleared (can be written simultaneously with ADDR).
- Proceed to transmit data.

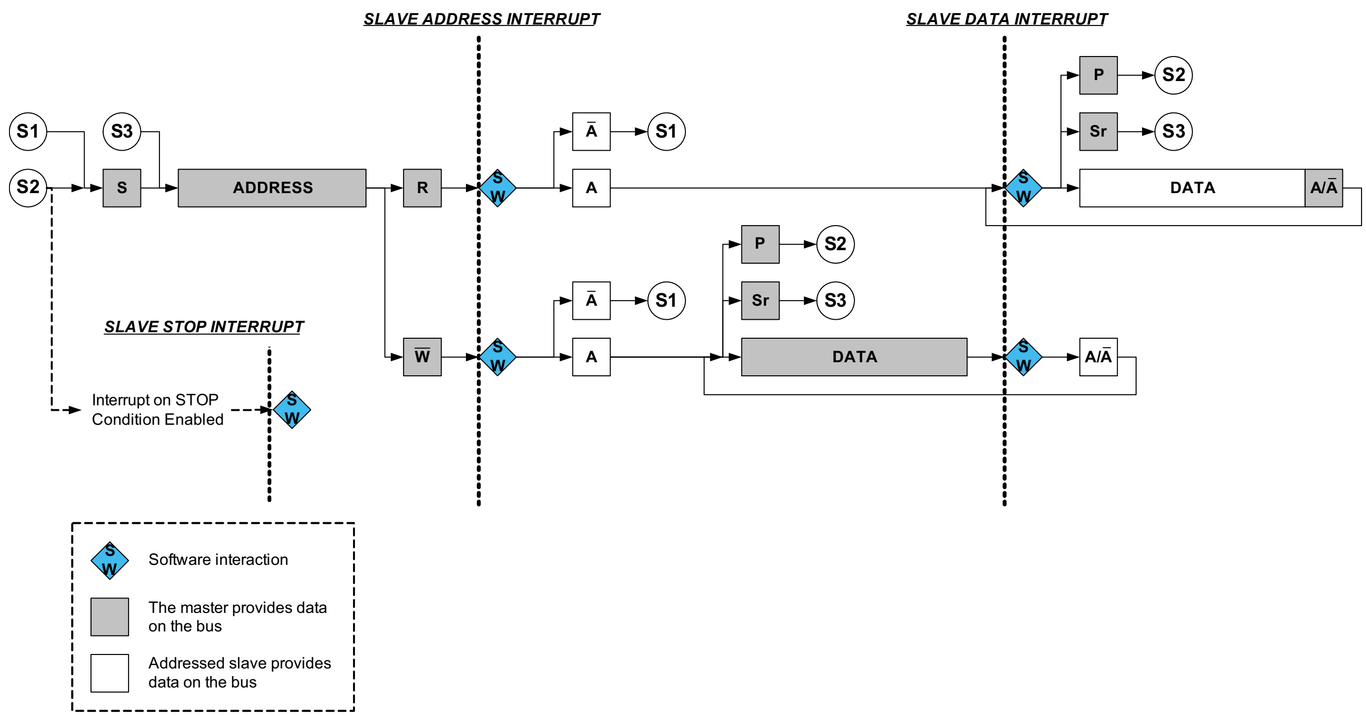
27.6.2.7 I²C Slave Operation

The I²C slave is byte-oriented and interrupt-based. The number of interrupts generated is kept at a minimum by automatic handling of most events. Auto triggering of operations and a special smart mode, which can be enabled by writing a 1 to the Smart Mode Enable bit in the Control A register (CTRLA.SMEN), are included to reduce software's complexity and code size.

The I²C slave has two interrupt strategies. When SCL Stretch Mode (CTRLA.SCLSM) is set to zero, SCL is stretched before or after the acknowledge bit. In this mode, the I²C slave operates according to the behavior diagram shown in Figure 27-10. The circles with a capital S followed by a number (S1, S2... etc.) indicate which node in the figure the bus logic can jump to based on software or hardware interaction.

This diagram is used as reference for the description of the I²C slave operation throughout the document.

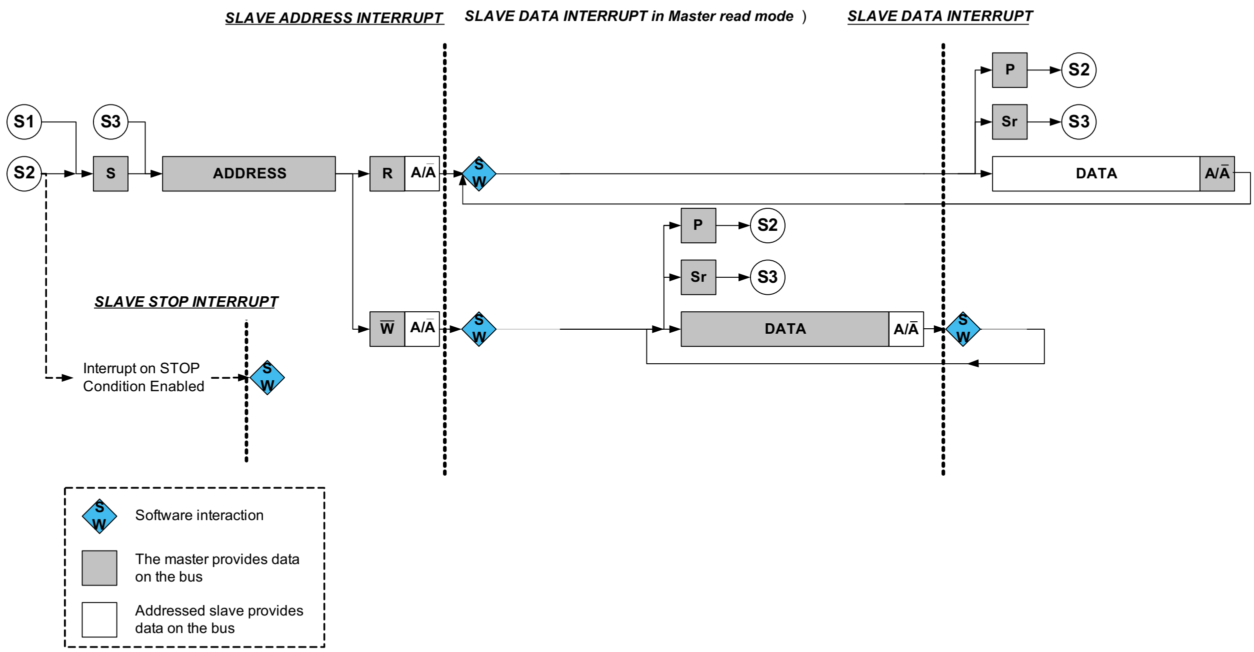
Figure 27-10.I²C Slave Behavioral Diagram (SCLSM=0)



In the second strategy (SCLSM=1), interrupts only occur after the ACK bit as shown in Figure 27-11. This strategy can be used when it is not necessary to check DATA before acknowledging. For master reads, an address and data interrupt will be issued simultaneously after the address acknowledge, while for master writes, the first data interrupt will be seen after the first data byte has been received by the slave and the acknowledge bit has been sent to the master.

Note that setting SCLSM to 1 is required for High-speed mode.

Figure 27-11.Slave Behavioral Diagram (SCLSM=1)



Receiving Address Packets (SCLSM=0)

When SCLSM is zero, the I^2C slave stretches the SCL line according to Figure 27-10. When the I^2C slave is properly configured, it will wait for a start condition to be detected. When a start condition is detected, the successive address packet will be received and checked by the address match logic. If the received address is not a match, the packet is rejected and the I^2C slave waits for a new start condition. The I^2C slave Address Match bit in the Interrupt Flag register (INTFLAG.AMATCH) is set when a start condition followed by a valid address packet is detected. SCL will be stretched until the I^2C slave clears INTFLAG.AMATCH. Because the I^2C slave holds the clock by forcing SCL low, the software is given unlimited time to respond to the address.

The direction of a transaction is determined by reading the Read / Write Direction bit in the Status register (STATUS.DIR), and the bit will be updated only when a valid address packet is received.

If the Transmit Collision bit in the Status register (STATUS.COLL) is set, this indicates that the last packet addressed to the I^2C slave had a packet collision. A collision causes the SDA and SCL lines to be released without any notification to software. The next AMATCH interrupt is, therefore, the first indication of the previous packet's collision. Collisions are intended to follow the SMBus Address Resolution Protocol (ARP).

After the address packet has been received from the I^2C master, one of two cases will arise based on transfer direction.

Case 1: Address packet accepted – Read flag set

The STATUS.DIR bit is one, indicating an I^2C master read operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, I^2C slave hardware will set the Data Ready bit in the Interrupt Flag register (INTFLAG.DRDY), indicating data are needed for transmit. If not acknowledge is sent, the I^2C slave will wait for a new start condition and address match.

Typically, software will immediately acknowledge the address packet by sending an ACK/NACK bit. The I^2C slave command CTRLB.CMD = 3 can be used for both read and write operation as the command execution is dependent on the STATUS.DIR bit.

Writing a one to INTFLAG.AMATCH will also cause an ACK/NACK to be sent corresponding to the CTRLB.ACKACT bit.

Case 2: Address packet accepted – Write flag set

The STATUS.DIR bit is cleared, indicating an I^2C master write operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, the I^2C slave will wait for data to be received. Data, repeated start or stop can be received.

If not acknowledge is sent, the I^2C slave will wait for a new start condition and address match.

Typically, software will immediately acknowledge the address packet by sending an ACK/NACK bit. The I^2C slave command CTRLB.CMD = 3 can be used for both read and write operation as the command execution is dependent on STATUS.DIR.

Writing a one to INTFLAG.AMATCH will also cause an ACK/NACK to be sent corresponding to the CTRLB.ACKACT bit.

Receiving Address Packets (SCLSM=1)

When SCLSM is one, the I^2C slave only stretches the SCL line after an acknowledge according to Figure 27-11. When the I^2C slave is properly configured, it will wait for a start condition to be detected. When a start condition is detected, the successive address packet will be received and checked by the address match logic. If the received address is not a match, the packet is rejected and the I^2C slave waits for a new start condition. If the address matches, the acknowledge action (CTRLB.ACKACT) is automatically sent and the Address Match bit in the Interrupt Flag register (INTFLAG.AMATCH) is set. SCL will be stretched until the I^2C slave clears INTFLAG.AMATCH. Because the I^2C slave holds the clock by forcing SCL low, the software is given unlimited time to respond to the address.

The direction of a transaction is determined by reading the Read / Write Direction bit in the Status register (STATUS.DIR), and the bit will be updated only when a valid address packet is received.

If the Transmit Collision bit in the Status register (STATUS.COLL) is set, this indicates that the last packet addressed to the I^2C slave had a packet collision. A collision causes the SDA and SCL lines to be released without any notification to software. The next AMATCH interrupt is, therefore, the first indication of the previous packet's collision. Collisions are intended to follow the SMBus Address Resolution Protocol (ARP).

After the address packet has been received from the I^2C master, a one can be written to INTFLAG.AMATCH to clear it.

Receiving and Transmitting Data Packets (SCLSM=0)

After the I^2C slave has received an address packet, it will respond according to the direction either by waiting for the data packet to be received or by starting to send a data packet by writing to DATA.DATA. When a data packet is received or sent, INTFLAG.DRDY will be set. Then, if the I^2C slave was receiving data, it will send an acknowledge according to CTRLB.ACKACT.

Case 1: Data received

INTFLAG.DRDY is set, and SCL is held low pending SW interaction.

Case 2: Data sent

When a byte transmission is successfully completed, the INTFLAG.DRDY interrupt flag is set. If NACK is received, the I^2C slave must expect a stop or a repeated start to be received. The I^2C slave must release the data line to allow the I^2C master to generate a stop or repeated start.

Upon stop detection, the Stop Received bit in the Interrupt Flag register (INTFLAG.PREC) will be set and the I^2C slave will return to the idle state.

High Speed Mode

When the I^2C slave is configured in High-speed mode (CTRLA.SPEED=0x2) with SCLSM set to one, switching between Full-speed and High-speed modes is automatic. When the slave recognizes a START followed by a master code transmission and a NACK, it automatically switches to High-speed mode and sets the High-speed status bit (STATUS.HS). The slave will then remain in High-speed mode until a STOP is received.

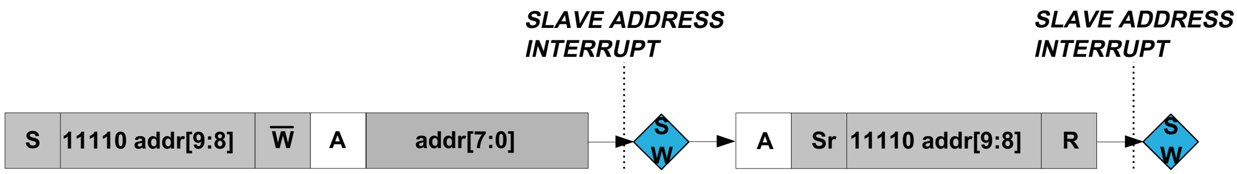
10-Bit Addressing

When 10-bit addressing is enabled (ADDR.TENBITEN=1) the two address bytes following a START will be checked against the 10-bit slave address recognition. The first byte of the address will always be acknowledged and the second byte will raise the address interrupt flag as shown in Figure 27-12.

If the transaction is a write, then the 10-bit address will be followed by N data bytes.

If the operation is a read, the 10-bit address will be followed by a repeated START and reception of "11110 ADDR[9:8] 1" and the second address interrupt will be received with the DIR bit set. The slave matches on the second address as it remembers that is was addressed by the previous 10-bit address.

Figure 27-12.10-bit Addressing



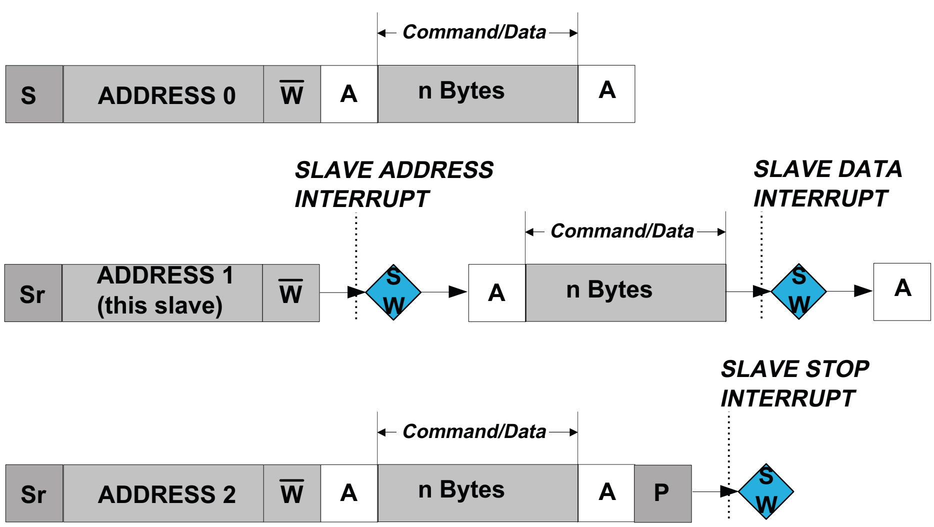
PMBus Group Command

When the group command bit is set (CTRLB.GCMD) and 7-bit addressing is used, a STOP interrupt will be generated if the slave has been addressed since the last STOP condition.

The group command protocol is used to send commands to more than one device. The commands are sent in one continuous transmission with a single STOP condition at the end. When the STOP condition is detected by the slaves addressed during the group command, they all begin executing the command they received.

Figure 27-13 shows an example where this slave is addressed by ADDRESS 1. This slave is addressed after a repeated START condition. There can be multiple slaves addressed before and after, then at the end of the group command, a single STOP is generated by the master. At this point a STOP interrupt is asserted.

Figure 27-13.PMBus Group Command Example



27.6.3 Additional Features

27.6.3.1 SMBus

The I^{2}C hardware incorporates three hardware SCL low time-outs which allows a time-out to occur for SMBus SCL low time-out, master extend time-out, and slave extend time-out. These time-outs are driven by the GCLK_SERCOM_SLOW clock. The GCLK_SERCOM_SLOW clock is used to accurately time the time-out and must be configured to used a 32kHz oscillator. The I^{2}C interface also allows for an SMBus compatible SDA hold time.

- $T_{TIMEOUT}$: SCL low time of 25-35 ms. – Measured for a single SCL low period. Enabled by bit CTRLA.LOWTOUTEN.
- $T_{LOW:SEXT}$:Cumulative clock low extend time of 25 ms – Measured as the cumulative SCL low extend time by a slave device in a single message from the initial START to the STOP. Enabled by bit CTRLA.SEXTTOEN.
- $T_{LOW:MEXT}$: Cumulative clock low extend time of 10 ms. – Measured as the cumulative SCL low extend time by the master device within a single byte from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. Enabled by bit (CTRLA.MEXTTOEN.

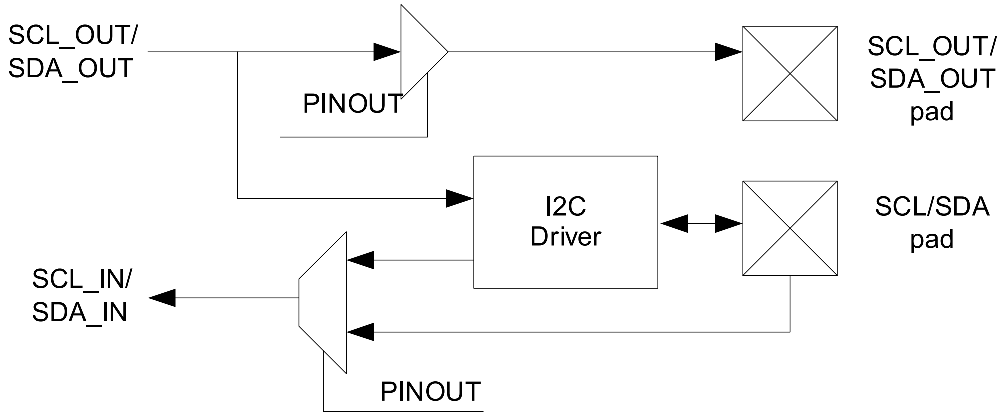
27.6.3.2 Smart Mode

The I^{2}C interface incorporates a special smart mode that simplifies application code and minimizes the user interaction needed to keep hold of the I^{2}C protocol. The smart mode accomplishes this by letting the reading of DATA.DATA automatically issue an ACK or NACK based on the state of CTRLB.ACKACT.

27.6.3.3 4-Wire Mode

Setting the Pin Usage bit in the Control A register (CTRLA.PINOUT) for master or slave to 4-wire mode enables operation as shown in Figure 27-14. In this mode, the internal I^{2}C tri-state drivers are bypassed, and an external, I^{2}C-compliant tri-state driver is needed when connecting to an I^{2}C bus.

Figure 27-14.I2C Pad Interface



27.6.3.4 Quick Command

Setting the Quick Command Enable bit in the Control B register (CTRLB.QCEN) enables quick command. When quick command is enabled, the corresponding interrupt flag is set immediately after the slave acknowledges the address. At this point, the software can either issue a stop command or a repeated start by writing CTRLB.CMD or ADDR.ADDR.

27.6.4 DMA, Interrupts and Events

Table 27-1. Module Request for SERCOM I²C Slave

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Data Ready	x				
Data received (Slave receive mode)				x	when data is read
Data needed for transmit (Slave transmit mode)				x	when data is written
Address Match	x				
Stop received	x				
Error	x				

Table 27-2. Module Request for SERCOM I²C Master

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Master on Bus	x				
Slave on Bus	x				
Data received (Master receive mode)				x	when data is read
Data needed for transmit (Master transmit mode)				x	when data is written
Error	x				

27.6.4.1 DMA Operation

Smart mode (CTRLB.SMEN) must be enabled for DMA operation.

Slave DMA

When using the I²C slave with DMA, an address match will cause the address interrupt flag (INTFLAG.ADDRMATCH) to be raised. After the interrupt has been serviced, data transfer will be performed through DMA.

The I²C slave generates the following requests:

- Write data received (RX): The request is set when master write data is received. The request is cleared when DATA is read.
- Read data needed for transmit (TX): The request is set when data is needed for a master read operation. The request is cleared when DATA is written.

Master DMA

When using the I²C master with DMA, the ADDR register must be written with the desired address (ADDR.ADDR), transaction length (ADDR.LEN), and transaction length enable (ADDR.LENEN). When ADDR.LENEN is written to 1 along with ADDR.ADDR, ADDR.LEN determines the number of data bytes in the transaction from 0 to 255. DMA is then used to transfer ADDR.LEN bytes followed by an automatically generated NACK (for master reads) and a STOP.

If a NACK is received by the slave for a master write transaction before ADDR.LEN bytes, a STOP will be automatically generated and the length error (STATUS.LENERR) will be raised along with the INTFLAG.ERROR interrupt.

The I^2C master generates the following requests:

- Read data received (RX): The request is set when master read data is received. The request is cleared when DATA is read.
- Write data needed for transmit (TX): The request is set when data is needed for a master write operation. The request is cleared when DATA is written.

27.6.4.2 Interrupts

The I^2C slave has the following interrupt sources:

- Error
- Data Ready
- Address Match
- Stop Received

The I^2C master has the following interrupt sources:

- Error
- Slave on Bus
- Master on Bus

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the I^2C is reset. See INTFLAG for details on how to clear interrupt flags.

The I^2C has one common interrupt request line for all the interrupt sources. The user must read INTFLAG to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

27.6.4.3 Events

Not applicable.

27.6.5 Sleep Mode Operation

During I^2C master operation, the generic clock (GCLK_SERCOMx_CORE) will continue to run in idle sleep mode. If the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY) is one, the GLK_SERCOMx_CORE will also run in standby sleep mode. Any interrupt can wake up the device.

If CTRLA.RUNSTDBY is zero during I^2C master operation, the GLK_SERCOMx_CORE will be disabled when an ongoing transaction is finished. Any interrupt can wake up the device.

During I^2C slave operation, writing a one to CTRLA.RUNSTDBY will allow the Address Match interrupt to wake up the device.

In I^2C slave operation, all receptions will be dropped when CTRLA.RUNSTDBY is zero.

27.6.6 Synchronization

Due to the asynchronicity between CLK_SERCOMx_APB and GCLK_SERCOMx_CORE, some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the corresponding Synchronization Busy bit in the Synchronization Busy register (SYNCBUSY) will be set immediately, and cleared when synchronization is complete.

If an operation that requires synchronization is executed while the corresponding SYNCBUSY bit is one, a peripheral bus error is generated.

The following bits need synchronization when written:

- Software Reset bit in the Control A register (CTRLA.SWRST). SYNCBUSY.SWRST is set to one while synchronization is in progress.
- Enable bit in the Control A register (CTRLA.ENABLE). SYNCBUSY.ENABLE is set to one while synchronization is in progress.
- Write to Bus State bits in the Status register (STATUS.BUSSTATE). SYNCBUSY.SYSOP is set to one while synchronization is in progress.
- Address bits in the Address register (ADDR.ADDR) when in master operation. SYNCBUSY.SYSOP is set to one while synchronization is in progress.
- Data (DATA) when in master operation. SYNCBUSY.SYSOP is set to one while synchronization is in progress.

Write-synchronization is denoted by the Write-Synchronized property in the register description.

27.7 Register Summary

Table 27-3. Register Summary – Slave Mode

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]=100			ENABLE	SWRST
0x01		15:8								
0x02		23:16	SEXTTOEN		SDAHOLD[1:0]					PINOUT
0x03		31:24		LOWTOUT			SCLSM		SPEED[1:0]	
0x04	CTRLB	7:0								
0x05		15:8	AMODE[1:0]					AACKEN	GCMD	SMEN
0x06		23:16						ACKACT	CMD[1:0]	
0x07		31:24								
0x08	Reserved									
...	Reserved									
0x13	Reserved									
0x14	INTENCLR	7:0	ERROR					DRDY	AMATCH	PREC
0x15	Reserved									
0x16	INTENSET	7:0	ERROR					DRDY	AMATCH	PREC
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR					DRDY	AMATCH	PREC
0x19	Reserved									
0x1A	STATUS	7:0	CLKHOLD	LOWTOUT		SR	DIR	RXNACK	COLL	BUSERR
0x1B		15:8	SYNCBUSY					HS	SEXTTOUT	

Table 27-3. Register Summary – Slave Mode (Continued)

Offset	Name	Bit Pos.								
0x1C	SYNCBUSY	7:0							ENABLE	SWRST
0x1D		15:8								
0x1E		23:16								
0x1F		31:24								
0x20	Reserved									
0x21	Reserved									
0x22	Reserved									
0x23	Reserved									
0x24	ADDR	7:0	ADDR[6:0]							GENCEN
0x25		15:8	TENBITEN					ADDR[9:7]		
0x26		23:16	ADDRMASK[6:0]							
0x27		31:24						ADDRMASK[9:7]		
0x28	DATA	7:0	DATA[7:0]							
0x29		15:8								

Table 27-4. Register Summary – Master Mode

Offset	Name	Bit Pos								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]=101			ENABLE	SWRST
0x01		15:8								
0x02		23:16	SEXTTOEN	MEXTTOEN	SDAHOLD[1:0]					PINOUT
0x03		31:24		LOWTOUT	INACTOUT[1:0]		SCLSM		SPEED[1:0]	
0x04	CTRLB	7:0								
0x05		15:8							QCEN	SMEN
0x06		23:16						ACKACT	CMD[1:0]	
0x07		31:24								
0x08	Reserved									
0x09	Reserved									
0x0A	Reserved									
0x0B	Reserved									
0x0C	BAUD	7:0	BAUD[7:0]							
0x0D		15:8	BAUDLOW[7:0]							
0x0E		23:16	HSBAUD[7:0]							
0x0F		31:24	HSBAUDLOW[7:0]							
0x10	Reserved									
0x11	Reserved									
0x12	Reserved									
0x13	Reserved									
0x14	INTENCLR	7:0	ERROR						SB	MB

Table 27-4. Register Summary – Master Mode (Continued)

Offset	Name	Bit Pos								
0x15	Reserved									
0x16	INTENSET	7:0	ERROR						SB	MB
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR						SB	MB
0x19	Reserved									
0x1A	STATUS	7:0	CLKHOLD	LOWTOUT	BUSSTATE[1:0]			RXNACK	ARBLOST	BUSERR
0x1B		15:8						LENERR	SEXTTOUT	MEXTTOUT
0x1C	SYNCBUSY	7:0						SYSOP	ENABLE	SWRST
0x1D		15:8								
0x1E		23:16								
0x1F		31:24								
0x20	Reserved									
0x21	Reserved									
0x22	Reserved									
0x23	Reserved									

Table 27-4. Register Summary – Master Mode (Continued)

Offset	Name	Bit Pos								
0x24	ADDR	7:0	ADDR[7:0]							
0x25		15:8	TENBITEN	HS	LENEN			ADDR[10:8]		
0x26		23:16	LEN[7:0]							
0x27		31:24								
0x28	DATA	7:0	DATA[7:0]							
0x29		15:8								
0x2A	Reserved									
...	Reserved									
0x2F	Reserved									
0x30	DBGCTRL	7:0								DBGSTOP

27.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Please refer to "Register Access Protection" on page 508 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Write-Synchronized or the Read-Synchronized property in each individual register description. Please refer to "Synchronization" on page 524 for details.

Some registers are enable-protected, meaning they can only be written when the I^2C is disabled. Enable-protection is denoted by the Enable-Protected property in each individual register description.

27.8.1 I[2]C Slave Register Description

27.8.1.1 Control A

Name: CTRLA

Offset: 0x00

Reset: 0x00000000

Property: Write-Protected, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
		LOWTOUT			SCLSM		SPEED[1:0]	
Access	R	R/W	R	R	R/W	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SEXTTOEN		SDAHOLD[1:0]					PINOUT
Access	R/W	R	R/W	R/W	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]=100			ENABLE	SWRST
Access	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 31 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 30 – LOWTOUT: SCL Low Time-Out**
  This bit enables the SCL low time-out. If SCL is held low for 25ms-35ms, the slave will release its clock hold, if enabled, and reset the internal state machine. Any interrupts set at the time of time-out will remain set.

  0: Time-out disabled.

  1: Time-out enabled.

- **Bits 29:28 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 27– SCLSM: SCL Clock Stretch Mode**
  This bit controls when SCL will be stretch for software interaction.

  0: SCL stretch according to Figure 27-7

1: SCL stretch only after ACK bit according to Figure 27-10.

This bit is not synchronized.

- **Bit 26– Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 25:24 – SPEED[1:0]: Transfer Speed**
  These bits define bus speed.

Table 27-5. Transfer Speed

Value	Description
0x0	Standard-mode (Sm) up to 100 kHz and Fast-mode (Fm) up to 400 kHz
0x1	Fast-mode Plus (Fm+) up to 1 MHz
0x2	High-speed mode (Hs-mode) up to 3.4 MHz
0x3	Reserved

These bits are not synchronized.

- **Bit 23 – SEXTTOEN: Slave SCL Low Extend Time-Out**
  This bit enables the slave SCL low extend time-out. If SCL is cumulatively held low for greater than 25ms from the initial START to a STOP, the slave will release its clock hold if enabled and reset the internal state machine. Any interrupts set at the time of time-out will remain set. If the address was recognized, PREC will be set when a STOP is received.

  0: Time-out disabled

  1: Time-out enabled

  This bit is not synchronized.

- **Bit 22 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 21:20 – SDAHOLD[1:0]: SDA Hold Time**
  These bits define the SDA hold time with respect to the negative edge of SCL.

Table 27-6. SDA Hold Time

Value	Name	Description
0x0	DIS	Disabled
0x1	75	50-100ns hold time
0x2	450	300-600ns hold time
0x3	600	400-800ns hold time

These bits are not synchronized.

- **Bits 19:17 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 16 – PINOUT: Pin Usage**
  This bit sets the pin usage to either two- or four-wire operation:
  0: 4-wire operation disabled
  1: 4-wire operation enabled
  This bit is not synchronized.

- **Bits 15:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 7 – RUNSTDBY: Run in Standby**
  This bit defines the functionality in standby sleep mode.
  0: Disabled – All reception is dropped.
  1: Wake on address match, if enabled.
  This bit is not synchronized.

- **Bits 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:2 – MODE[2:0]: Operating Mode**
  These bits must be written to 0x04 to select the I^2C slave serial communication interface of the SERCOM.
  These bits are not synchronized.

- **Bit 1 – ENABLE: Enable**
  0: The peripheral is disabled.
  1: The peripheral is enabled.
  Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately and the Enable Synchronization Busy bit in the Synchronization busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.
  This bit is not enable-protected.

- **Bit 0 – SWRST: Software Reset**
  0: There is no reset operation ongoing.
  1: The reset operation is ongoing.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.
  Writing a one to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.
  Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.
  This bit is not enable-protected.

27.8.1.2 Control B

Name: CTRLB

Offset: 0x04

Reset: 0x00000000

Property: Write-Protected, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
						ACKACT	CMD[1:0]	
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AMODE[1:0]					AACKEN	GCMD	SMEN
Access	R/W	R/W	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:19 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 18 – ACKACT: Acknowledge Action**
  0: Send ACK
  1: Send NACK
  The Acknowledge Action (ACKACT) bit defines the slave's acknowledge behavior after an address or data byte is received from the master. The acknowledge action is executed when a command is written to the CMD bits. If smart mode is enabled (CTRLB.SMEN is one), the acknowledge action is performed when the DATA register is read.
  This bit is not enable-protected.

- **Bits 17:16 – CMD[1:0]: Command**
  Writing the Command bits (CMD) triggers the slave operation as defined in Table 27-7. The CMD bits are strobe bits, and always read as zero. The operation is dependent on the slave interrupt flags, INTFLAG.DRDY and INTFLAG.AMATCH, in addition to STATUS.DIR (See Table 27-7).

All interrupt flags (INTFLAG.DRDY, INTFLAG.AMATCH and INTFLAG.PREC) are automatically cleared when a command is given.

This bit is not enable-protected.

Table 27-7. Command Description

CMD[1:0]	DIR	Action
0x0	X	(No action)
0x1	X	(Reserved)
0x2	Used to complete a transaction in response to a data interrupt (DRDY)	
	0 (Master write)	Execute acknowledge action succeeded by waiting for any start (S/Sr) condition
	1 (Master read)	Wait for any start (S/Sr) condition
0x3	Used in response to an address interrupt (AMATCH)	
	0 (Master write)	Execute acknowledge action succeeded by reception of next byte
	1 (Master read)	Execute acknowledge action succeeded by slave data interrupt
	Used in response to a data interrupt (DRDY)	
	0 (Master write)	Execute acknowledge action succeeded by reception of next byte
	1 (Master read)	Execute a byte read operation followed by ACK/NACK reception

- **Bits 15:14 – AMODE[1:0]: Address Mode**
  These bits set the addressing mode according to Table 27-8.

Table 27-8. Address Mode Description

Value	Name	Description
0x0	MASK	The slave responds to the address written in ADDR.ADDR masked by the value in ADDR.ADDRMASK(1).
0x1	2_ADDRS	The slave responds to the two unique addresses in ADDR.ADDR and ADDR.ADDRMASK.
0x2	RANGE	The slave responds to the range of addresses between and including ADDR.ADDR and ADDR.ADDRMASK. ADDR.ADDR is the upper limit.
0x3	-	Reserved.

Note: 1. See "SERCOM – Serial Communication Interface" on page 427 for additional information.

These bits are not write-synchronized.

- **Bits 13:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 10– AACKEN: Automatic Acknowledge Enable**
  This bit enables the address to be automatically acknowledged if there is an address match.
  0: Automatic acknowledge is disabled.
  1: Automatic acknowledge is enabled.
  This bit is not write-synchronized.

- **Bit 9 – GCMD: PMBus Group Command**
  This bit enables PMBus group command support. When enabled, a STOP interrupt will be generated if the slave has been addressed since the last STOP condition on the bus.

  0: Group command is disabled.

  1: Group command is enabled.

  This bit is not write-synchronized.

- **Bit 8 – SMEN: Smart Mode Enable**
  This bit enables smart mode. When smart mode is enabled, acknowledge action is sent when DATA.DATA is read.

  0: Smart mode is disabled.

  1: Smart mode is enabled.

  This bit is not write-synchronized.

- **Bits 7:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

27.8.1.3 Interrupt Enable Clear

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Name: INTENCLR

Offset: 0x14

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ERROR					DRDY	AMATCH	PREC
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7– ERROR: Error Interrupt Enable**
  0: Error interrupt is disabled.
  1: Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

- **Bits 6:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – DRDY: Data Ready Interrupt Enable**
  0: The Data Ready interrupt is disabled.
  1: The Data Ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Data Ready bit, which disables the Data Ready interrupt.

- **Bit 1 – AMATCH: Address Match Interrupt Enable**
  0: The Address Match interrupt is disabled.
  1: The Address Match interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Address Match Interrupt Enable bit, which disables the Address Match interrupt.

- **Bit 0 – PREC: Stop Received Interrupt Enable**
  0: The Stop Received interrupt is disabled.
  1: The Stop Received interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Stop Received bit, which disables the Stop Received interrupt.

27.8.1.4 Interrupt Enable Set

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Name: INTENSET

Offset: 0x16

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ERROR					DRDY	AMATCH	PREC
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – ERROR: Error Interrupt Enable**
  0: Error interrupt is disabled.
  1: Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.
- **Bits 6:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 2 – DRDY: Data Ready Interrupt Enable**
  0: The Data Ready interrupt is disabled.
  1: The Data Ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Data Ready bit, which enables the Data Ready interrupt.
- **Bit 1 – AMATCH: Address Match Interrupt Enable**
  0: The Address Match interrupt is disabled.
  1: The Address Match interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Address Match Interrupt Enable bit, which enables the Address Match interrupt.
- **Bit 0 – PREC: Stop Received Interrupt Enable**
  0: The Stop Received interrupt is disabled.
  1: The Stop Received interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Stop Received bit, which enables the Stop Received interrupt.

27.8.1.5 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x18

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	ERROR					DRDY	AMATCH	PREC
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7– ERROR: Error**

  This flag is cleared by writing a one to it.

  This bit is set when any error is detected. Errors that will set this flag have corresponding status flags in the STATUS register. Errors that will set this flag are SEXTTOUT, LOWTOUT, COLL, and BUSERR.Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the flag.

- **Bits 6:3 – Reserved**

  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – DRDY: Data Ready**

  This flag is set when a I^2C slave byte transmission is successfully completed.

  The flag is cleared by hardware when either:

  - Writing to the DATA register.
  - Reading the DATA register with smart mode enabled.
  - Writing a valid command to the CMD register.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Data Ready interrupt flag. Optionally, the flag can be cleared manually by writing a one to INTFLAG.DRDY.

- **Bit 1 – AMATCH: Address Match**

  This flag is set when the I^2C slave address match logic detects that a valid address has been received.

  The flag is cleared by hardware when CTRL.CMD is written.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Address Match interrupt flag. Optionally the flag can be cleared manually by writing a one to INTFLAG.AMATCH. When cleared, an ACK/NACK will be sent according to CTRLB.ACKACT.

- **Bit 0 – PREC: Stop Received**

  This flag is set when a stop condition is detected for a transaction being processed. A stop condition detected between a bus master and another slave will not set this flag.

  This flag is cleared by hardware after a command is issued on the next address match.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Stop Received interrupt flag. Optionally, the flag can be cleared manually by writing a one to INTFLAG.PREC.

27.8.1.6 Status

Name: STATUS

Offset: 0x1A

Reset: 0x0000

Property: -

Bit	15	14	13	12	11	10	9	8
						HS	SEXTTOUT	
Access	R	R	R	R	R	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLKHOLD	LOWTOUT		SR	DIR	RXNACK	COLL	BUSERR
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 10 – HS: High-speed**
  This bit is set if the slave detects a START followed by a Master Code transmission.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the status. However, this flag is automatically cleared when a STOP is received.

- **Bit 9 – SEXTTOUT: Slave SCL Low Extend Time-Out**
  This bit is set if a slave SCL low extend time-out occurs.

  This bit is cleared automatically if responding to a new start condition with ACK or NACK (write 3 to CTRLB.CMD) or when INTFLAG.AMATCH is cleared.

  0: No SCL low extend time-out has occurred.

  1: SCL low extend time-out has occurred.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the status.

- **Bit 8 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 7 – CLKHOLD: Clock Hold**
  The slave Clock Hold bit (STATUS.CLKHOLD) is set when the slave is holding the SCL line low, stretching the I^2C clock. Software should consider this bit a read-only status flag that is set when INTFLAG.DRDY or INTFLAG.AMATCH is set.

  This bit is automatically cleared when the corresponding interrupt is also cleared.

- **Bit 6 – LOWTOUT: SCL Low Time-out**
  This bit is set if an SCL low time-out occurs.

This bit is cleared automatically if responding to a new start condition with ACK or NACK (write 3 to CTRLB.CMD) or when INTFLAG.AMATCH is cleared.

0: No SCL low time-out has occurred.

1: SCL low time-out has occurred.

Writing a zero to this bit has no effect.

Writing a one to this bit will clear the status.

- **Bit 5 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 4 – SR: Repeated Start**
  When INTFLAG.AMATCH is raised due to an address match, SR indicates a repeated start or start condition.

  0: Start condition on last address match

  1: Repeated start condition on last address match

  This flag is only valid while the INTFLAG.AMATCH flag is one.

- **Bit 3 – DIR: Read / Write Direction**
  The Read/Write Direction (STATUS.DIR) bit stores the direction of the last address packet received from a master.

  0: Master write operation is in progress.

  1: Master read operation is in progress.

- **Bit 2 – RXNACK: Received Not Acknowledge**
  This bit indicates whether the last data packet sent was acknowledged or not.

  0: Master responded with ACK.

  1: Master responded with NACK.

- **Bit 1 – COLL: Transmit Collision**
  If set, the I^{2}C slave was not able to transmit a high data or NACK bit, the I^{2}C slave will immediately release the SDA and SCL lines and wait for the next packet addressed to it.

  This flag is intended for the SMBus address resolution protocol (ARP). A detected collision in non-ARP situations indicates that there has been a protocol violation, and should be treated as a bus error.

  Note that this status will not trigger any interrupt, and should be checked by software to verify that the data were sent correctly. This bit is cleared automatically if responding to an address match with an ACK or a NACK (writing 0x3 to CTRLB.CMD), or INTFLAG.AMATCH is cleared.

  0: No collision detected on last data byte sent.

  1: Collision detected on last data byte sent.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the status.

- **Bit 0 – BUSERR: Bus Error**
  The Bus Error bit (STATUS.BUSERR) indicates that an illegal bus condition has occurred on the bus, regardless of bus ownership. An illegal bus condition is detected if a protocol violating start, repeated start or stop is detected on the I^{2}C bus lines. A start condition directly followed by a stop condition is one example of a protocol violation. If a time-out occurs during a frame, this is also considered a protocol violation, and will set STATUS.BUSERR.

  This bit is cleared automatically if responding to an address match with an ACK or a NACK (writing 0x3 to CTRLB.CMD) or INTFLAG.AMATCH is cleared.

  0: No bus error detected.

  1: Bus error detected.

  Writing a one to this bit will clear the status.

Writing a zero to this bit has no effect.

27.8.1.7 Synchronization Busy

Name: SYNCBUSY
Offset: 0x1C
Reset: 0x00000000
Property:

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – ENABLE: SERCOM Enable Synchronization Busy**
  Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. When written, the SYNC-BUSY.ENABLE bit will be set until synchronization is complete.

  Writes to any register (except for CTRLA.SWRST) while enable synchronization is on-going will be discarded and an APB error will be generated.

  0: Enable synchronization is not busy.

  1: Enable synchronization is busy.

- **Bit 0 – SWRST: Software Reset Synchronization Busy**
  Resetting the SERCOM (CTRLA.SWRST) requires synchronization. When written, the SYNCBUSY.SWRST bit will be set until synchronization is complete.

  Writes to any register while synchronization is on-going will be discarded and an APB error will be generated.

  0: SWRST synchronization is not busy.

  1: SWRST synchronization is busy.

27.8.1.8 Address

Name: ADDR

Offset: 0x24

Reset: 0x00000000

Property: Write-Protected, Enable-Protected

Bit	31	30	29	28	27	26	25	24
						ADDRMASK[9:7]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ADDRMASK[6:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TENBITEN					ADDR[9:7]		
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[6:0]							GENCEN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:27 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 26:17 – ADDRMASK[9:0]: Address Mask**
  The ADDRMASK bits acts as a second address match register, an address mask register or the lower limit of an address range, depending on the CTRLB.AMODE setting.

- **Bit 16– Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 15– TENBITEN: Ten Bit Addressing Enable**
  Writing a one to TENBITEN enables 10-bit address recognition.

  0: 10-bit address recognition disabled.

  1: 10-bit address recognition enabled.

- **Bits 14:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 10:1 – ADDR[9:0]: Address**
  The slave address (ADDR) bits contain the I^2C slave address used by the slave address match logic to determine if a master has addressed the slave.

  When using 7-bit addressing, the slave address is represented by ADDR.ADDR[6:0].

  When using 10-bit addressing (ADDR.TENBITEN=1), the slave address is represented by ADDR.ADDR[9:0]

  When the address match logic detects a match, INTFLAG.AMATCH is set and STATUS.DIR is updated to indicate whether it is a read or a write transaction.

- **Bit 0 – GENCEN: General Call Address Enable**
  Writing a one to GENCEN enables general call address recognition. A general call address is an address of all zeroes with the direction bit written to zero (master write).

  0: General call address recognition disabled.

  1: General call address recognition enabled.

27.8.1.9 Data

Name: DATA

Offset: 0x28

Reset: 0x0000

Property: Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – DATA[7:0]: Data**
  The slave data register I/O location (DATA.DATA) provides access to the master transmit and receive data buffers. Reading valid data or writing data to be transmitted can be successfully done only when SCL is held low by the slave (STATUS.CLKHOLD is set). An exception occurs when reading the last data byte after the stop condition has been received.

  Accessing DATA.DATA auto-triggers I^2C bus operations. The operation performed depends on the state of CTRLB.ACKACT, CTRLB.SMEN and the type of access (read/write).

  Writing or reading DATA.DATA when not in smart mode does not require synchronization.

27.8.2 I²C Master Register Description

27.8.2.1 Control A

Name: CTRLA

Offset: 0x00

Reset: 0x00000000

Property: Write-Protected, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
		LOWTOUT	INACTOUT[1:0]		SCLSM		SPEED[1:0]	
Access	R	R/W	R/W	R/W	R/W	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SEXTTOEN	MEXTTOEN	SDAHOLD[1:0]					PINOUT
Access	R/W	R/W	R/W	R/W	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]=101			ENABLE	SWRST
Access	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 31 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 30 – LOWTOUT: SCL Low Time-Out**
  This bit enables the SCL low time-out. If SCL is held low for 25ms-35ms, the master will release its clock hold, if enabled, and complete the current transaction. A stop condition will automatically be transmitted.

  INTFLAG.SB or INTFLAG.MB will be set as normal, but the clock hold will be released. The STATUS.LOW-TOUT and STATUS.BUSERR status bits will be set.

  0: Time-out disabled.

  1: Time-out enabled.

  This bit is not synchronized.

- **Bits 29:28 – INACTOUT[1:0]: Inactive Time-Out**
  If the inactive bus time-out is enabled and the bus is inactive for longer than the time-out setting, the bus state logic will be set to idle. An inactive bus arise when either an I²C master or slave is holding the SCL low. The available time-outs are given in Table 27-9.

Enabling this option is necessary for SMBus compatibility, but can also be used in a non-SMBus set-up.

Table 27-9. Inactive Timout

Value	Name	Description
0x0	DIS	Disabled
0x1	55US	5-6 SCL cycle time-out (50-60µs)
0x2	105US	10-11 SCL cycle time-out (100-110µs)
0x3	205US	20-21 SCL cycle time-out (200-210µs)

Calculated time-out periods are based on a 100kHz baud rate.

These bits are not synchronized.

- **Bit 27– SCLSM: SCL Clock Stretch Mode**
  This bit controls when SCL will be stretch for software interaction.
  0: SCL stretch according to Figure 27-7.
  1: SCL stretch only after ACK bit.
  This bit is not synchronized.

- **Bit 26– Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 25:24 – SPEED[1:0]: Transfer Speed**
  These bits define bus speed.

Table 27-10. Transfer Speed

Value	Description
0x0	Standard-mode (Sm) up to 100 kHz and Fast-mode (Fm) up to 400 kHz
0x1	Fast-mode Plus (Fm+) up to 1 MHz
0x2	High-speed mode (Hs-mode) up to 3.4 MHz
0x3	Reserved

These bits are not synchronized.

- **Bit 23 – SEXTTOEN: Slave SCL Low Extend Time-Out**
  This bit enables the slave SCL low extend time-out. If SCL is cumulatively held low for greater than 25ms from the initial START to a STOP, the slave will release its clock hold if enabled and reset the internal state machine. Any interrupts set at the time of time-out will remain set. If the address was recognized, PREC will be set when a STOP is received.
  0: Time-out disabled
  1: Time-out enabled
  This bit is not synchronized.

- **Bit 22 – MEXTTOEN: Master SCL Low Extend Time-Out**
  This bit enables the master SCL low extend time-out. If SCL is cumulatively held low for greater than 10ms from START-to-ACK, ACK-to-ACK, or ACK-to-STOP the master will release its clock hold if enabled, and complete the current transaction. A STOP will automatically be transmitted.

SB or MB will be set as normal, but CLKHOLD will be release. The MEXTTOUT and BUSERR status bits will be set.

0: Time-out disabled

1: Time-out enabled

This bit is not synchronized.

- **Bits 21:20 – SDAHOLD[1:0]: SDA Hold Time**
  These bits define the SDA hold time with respect to the negative edge of SCL.

Table 27-11. SDA Hold Time

Value	Name	Description
0x0	DIS	Disabled
0x1	75NS	50-100ns hold time
0x2	450NS	300-600ns hold time
0x3	600NS	400-800ns hold time

These bits are not synchronized.

- **Bits 19:17 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 16 – PINOUT: Pin Usage**
  This bit set the pin usage to either two- or four-wire operation:

  0: 4-wire operation disabled.

  1: 4-wire operation enabled.

  This bit is not synchronized.

- **Bits 15:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 7 – RUNSTDBY: Run in Standby**
  This bit defines the functionality in standby sleep mode.

  0: GCLK_SERCOMx_CORE is disabled and the I^2C master will not operate in standby sleep mode.

  1: GCLK_SERCOMx_CORE is enabled in all sleep modes allowing the master to operate in standby sleep mode.

  This bit is not synchronized.

- **Bits 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:2 – MODE[2:0]: Operating Mode**
  These bits must be written to 0x5 to select the I^2C master serial communication interface of the SERCOM.

  These bits are not synchronized.

- **Bit 1 – ENABLE: Enable**
  0: The peripheral is disabled.

  1: The peripheral is enabled.

Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately and the Synchronization Enable Busy bit in the Syncbusy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

This bit is not enable-protected.

- **Bit 0 – SWRST: Software Reset**

  0: There is no reset operation ongoing.

  1: The reset operation is ongoing.

  Writing a zero to this bit has no effect.

  Writing a one to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.

  Writing a one to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.

  Due to synchronization there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.

  This bit is not enable-protected.

27.8.2.2 Control B

Name: CTRLB

Offset: 0x04

Reset: 0x00000000

Property: Write-Protected, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
						ACKACT	CMD[1:0]	
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
							QCEN	SMEN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:19 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 18 – ACKACT: Acknowledge Action**
  The Acknowledge Action (ACKACT) bit defines the I^2C master's acknowledge behavior after a data byte is received from the I^2C slave. The acknowledge action is executed when a command is written to CTRLB.CMD, or if smart mode is enabled (CTRLB.SMEN is written to one), when DATA.DATA is read.

  0: Send ACK.

  1: Send NACK.

  This bit is not enable-protected.

  This bit is not write-synchronized.

- **Bits 17:16 – CMD[1:0]: Command**
  Writing the Command bits (CMD) triggers the master operation as defined in Table 27-12. The CMD bits are strobe bits, and always read as zero. The acknowledge action is only valid in master read mode. In master write mode, a command will only result in a repeated start or stop condition. The CTRLB.ACKACT bit and the CMD

bits can be written at the same time, and then the acknowledge action will be updated before the command is triggered.

Commands can only be issued when the Slave on Bus interrupt flag (INTFLAG.SB) or Master on Bus interrupt flag (INTFLAG.MB) is one.

If CMD 0x1 is issued, a repeated start will be issued followed by the transmission of the current address in ADDR.ADDR. If another address is desired, ADDR.ADDR must be written instead of the CMD bits. This will trigger a repeated start followed by transmission of the new address.

Issuing a command will set STATUS.SYNCBUSY.

Table 27-12. Command Description

CMD[1:0]	Direction	Action
0x0	X	(No action)
0x1	X	Execute acknowledge action succeeded by repeated Start
0x2	0 (Write)	No operation
	1 (Read)	Execute acknowledge action succeeded by a byte read operation
0x3	X	Execute acknowledge action succeeded by issuing a stop condition

These bits are not enable-protected.

- **Bits 15:10 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 9 – QCEN: Quick Command Enable**
  Setting the Quick Command Enable bit (QCEN) enables quick command.
  0: Quick Command is disabled.
  1: Quick Command is enabled.
  This bit is not write-synchronized.

- **Bit 8 – SMEN: Smart Mode Enable**
  This bit enables smart mode. When smart mode is enabled, acknowledge action is sent when DATA.DATA is read.
  0: Smart mode is disabled.
  1: Smart mode is enabled.
  This bit is not write-synchronized.

- **Bits 7:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

27.8.2.3 Baud Rate

Name: BAUD

Offset: 0x0C

Reset: 0x0000

Property: Write-Protected, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	HSBAUDLOW[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HSBAUD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BAUDLOW[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BAUD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:24 – HSBAUDLOW[7:0]: High Speed Master Baud Rate Low**
  HSBAUDLOW not equal to 0
  HSBAUDLOW indicates the SCL low time according to the following formula.
  $$HSBAUDLOW = f_{GCLK} T_{LOW} - 1$$
  HSBAUDLOW equal to 0
  The HSBAUD register is used to time $T_{LOW,}$ $T_{HIGH,}$ $T_{SU;STO,}$ $T_{HD;STA}$ and $T_{SU;STA.}$. T_{BUF} is timed by the BAUD register.

- **Bits 23:16 – HSBAUD[7:0]: High Speed Master Baud Rate**
  The HSBAUD register indicates the SCL high time according to the following formula. When HSBAUDLOW is zero, $T_{LOW,}$ $T_{HIGH,}$ $T_{SU;STO,}$ $T_{HD;STA}$ and $T_{SU;STA}$ are derived using this formula. T_{BUF} is timed by the BAUD register.
  $$BAUD = f_{GCLK} T_{HIGH} - 1$$

- **Bits 15:8 – BAUDLOW[7:0]: Master Baud Rate Low**
  If the Master Baud Rate Low bit group (BAUDLOW) has a non-zero value, the SCL low time will be described by the value written.
  For more information on how to calculate the frequency, see "SERCOM I2C – SERCOM Inter-Integrated Circuit" on page 506.

- **Bits 7:0 – BAUD[7:0]: Master Baud Rate**
  The Master Baud Rate bit group (BAUD) is used to derive the SCL high time if BAUD.BAUDLOW is non-zero. If BAUD.BAUDLOW is zero, BAUD will be used to generate both high and low periods of the SCL.
  For more information on how to calculate the frequency, see "SERCOM I2C – SERCOM Inter-Integrated Circuit" on page 506.

27.8.2.4 Interrupt Enable Clear

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Name: INTENCLR

Offset: 0x14

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ERROR						SB	MB
Access	R/W	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7– ERROR: Error Interrupt Enable**
  0: Error interrupt is disabled.
  1: Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

- **Bits 6:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – SB: Slave on Bus Interrupt Enable**
  0: The Slave on Bus interrupt is disabled.
  1: The Slave on Bus interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Slave on Bus Interrupt Enable bit, which disables the Slave on Bus interrupt.

- **Bit 0 – MB: Master on Bus Interrupt Enable**
  0: The Master on Bus interrupt is disabled.
  1: The Master on Bus interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Master on Bus Interrupt Enable bit, which disables the Master on Bus interrupt.

27.8.2.5 Interrupt Enable Set

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Name: INTENSET

Offset: 0x16

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ERROR						SB	MB
Access	R/W	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – ERROR: Error Interrupt Enable**
  0: Error interrupt is disabled.
  1: Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

- **Bits 6:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – SB: Slave on Bus Interrupt Enable**
  0: The Slave on Bus interrupt is disabled.
  1: The Slave on Bus interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Slave on Bus Interrupt Enable bit, which enables the Slave on Bus interrupt.

- **Bit 0 – MB: Master on Bus Interrupt Enable**
  0: The Master on Bus interrupt is disabled.
  1: The Master on Bus interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Master on Bus Interrupt Enable bit, which enables the Master on Bus interrupt.

27.8.2.6 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x18

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	ERROR						SB	MB
Access	R/W	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7– ERROR: Error**
  This flag is cleared by writing a one to it.
  This bit is set when any error is detected. Errors that will set this flag have corresponding status flags in the STATUS register. Errors that will set this flag are LENERR, SEXTTOUT, MEXTTOUT, LOWTOUT, ARBLOST, and BUSERR.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the flag.

- **Bits 6:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – SB: Slave on Bus**
  The Slave on Bus flag (SB) is set when a byte is successfully received in master read mode, i.e., no arbitration lost or bus error occurred during the operation. When this flag is set, the master forces the SCL line low, stretching the I^2C clock period. The SCL line will be released and SB will be cleared on one of the following actions:
  - Writing to ADDR.ADDR
  - Writing to DATA.DATA
  - Reading DATA.DATA when smart mode is enabled (CTRLB.SMEN)
  - Writing a valid command to CTRLB.CMD

  Writing a one to this bit location will clear the SB flag. The transaction will not continue or be terminated until one of the above actions is performed.
  Writing a zero to this bit has no effect.

- **Bit 0 – MB: Master on Bus**
  The Master on Bus flag (MB) is set when a byte is transmitted in master write mode. The flag is set regardless of the occurrence of a bus error or an arbitration lost condition. MB is also set when arbitration is lost during sending of NACK in master read mode, and when issuing a start condition if the bus state is unknown. When this flag is set and arbitration is not lost, the master forces the SCL line low, stretching the I^2C clock period. The SCL line will be released and MB will be cleared on one of the following actions:
  - Writing to ADDR.ADDR
  - Writing to DATA.DATA
  - Reading DATA.DATA when smart mode is enabled (CTRLB.SMEN)
  - Writing a valid command to CTRLB.CMD

  If arbitration is lost, writing a one to this bit location will clear the MB flag.

If arbitration is not lost, writing a one to this bit location will clear the MB flag. The transaction will not continue or be terminated until one of the above actions is performed.

Writing a zero to this bit has no effect.

27.8.2.7 Status

Name: STATUS

Offset: 0x1A

Reset: 0x0000

Property: Write-Synchronized

Bit	15	14	13	12	11	10	9	8
						LENERR	SEXTTOUT	MEXTTOUT
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLKHOLD	LOWTOUT	BUSSTATE[1:0]			RXNACK	ARBLOST	BUSERR
Access	R	R/W	R	R/W	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 10 – LENERR: Transaction Length Error**
  This bit is set when automatic length is used for a DMA transaction and the slave sends a NACK before ADDR.LEN bytes have been written by the master.
  Writing a one to this bit location will clear STATUS.LENERR. This flag is automatically cleared when writing to the ADDR register.
  Writing a zero to this bit has no effect.
  This bit is not write-synchronized.

- **Bit 9 – SEXTTOUT: Slave SCL Low Extend Time-Out**
  This bit is set if a slave SCL low extend time-out occurs.
  Writing a one to this bit location will clear STATUS.SEXTTOUT. Normal use of the I^2C interface does not require the STATUS.SEXTTOUT flag to be cleared by this method. This flag is automatically cleared when writing to the ADDR register.
  Writing a zero to this bit has no effect.
  This bit is not write-synchronized.

- **Bit 8 – MEXTTOUT: Master SCL Low Extend Time-Out**
  This bit is set if a master SCL low time-out occurs.
  Writing a one to this bit location will clear STATUS.MEXTTOUT. Normal use of the I^2C interface does not require the STATUS.MEXTTOUT flag to be cleared by this method. This flag is automatically cleared when writing to the ADDR register.
  Writing a zero to this bit has no effect.
  This bit is not write-synchronized.

- **Bit 7 – CLKHOLD: Clock Hold**
  The Master Clock Hold flag (STATUS.CLKHOLD) is set when the master is holding the SCL line low, stretching the I^2C clock. Software should consider this bit a read-only status flag that is set when INTFLAG.SB or INT-

FLAG.MB is set. When the corresponding interrupt flag is cleared and the next operation is given, this bit is automatically cleared.

Writing a zero to this bit has no effect.

Writing a one to this bit has no effect.

This bit is not write-synchronized.

- **Bit 6 – LOWTOUT: SCL Low Time-Out**

  This bit is set if an SCL low time-out occurs.

  Writing a one to this bit location will clear STATUS.LOWTOUT. Normal use of the I^2C interface does not require the LOWTOUT flag to be cleared by this method. This flag is automatically cleared when writing to the ADDR register.

  Writing a zero to this bit has no effect.

  This bit is not write-synchronized.

- **Bits 5:4 – BUSSTATE[1:0]: Bus State**

  These bits indicate the current I^2C bus state as defined in Table 27-13. After enabling the SERCOM as an I^2C master, the bus state will be unknown.

Table 27-13. Bus State

Value	Name	Description
0x0	Unknown	The bus state is unknown to the I^2C master and will wait for a stop condition to be detected or wait to be forced into an idle state by software
0x1	Idle	The bus state is waiting for a transaction to be initialized
0x2	Owner	The I^2C master is the current owner of the bus
0x3	Busy	Some other I^2C master owns the bus

When the master is disabled, the bus-state is unknown. When in the unknown state, writing 0x1 to BUSSTATE forces the bus state into the idle state. The bus state cannot be forced into any other state.

Writing STATUS.BUSSTATE to idle will set STATUS.SYNCBUSY.

- **Bit 3 – Reserved**

  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 2 – RXNACK: Received Not Acknowledge**

  This bit indicates whether the last address or data packet sent was acknowledged or not.

  0: Slave responded with ACK.

  1: Slave responded with NACK.

  Writing a zero to this bit has no effect.

  Writing a one to this bit has no effect.

  This bit is not write-synchronized.

- **Bit 1 – ARBLOST: Arbitration Lost**

  The Arbitration Lost flag (STATUS.ARBLOST) is set if arbitration is lost while transmitting a high data bit or a NACK bit, or while issuing a start or repeated start condition on the bus. The Master on Bus interrupt flag (INT-FLAG.MB) will be set when STATUS.ARBLOST is set.

  Writing the ADDR.ADDR register will automatically clear STATUS.ARBLOST.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear it.

This bit is not write-synchronized.

- **Bit 0 – BUSERR: Bus Error**
  The Bus Error bit (STATUS.BUSERR) indicates that an illegal bus condition has occurred on the bus, regardless of bus ownership. An illegal bus condition is detected if a protocol violating start, repeated start or stop is detected on the I^2C bus lines. A start condition directly followed by a stop condition is one example of a protocol violation. If a time-out occurs during a frame, this is also considered a protocol violation, and will set BUSERR.

  If the I^2C master is the bus owner at the time a bus error occurs, STATUS.ARBLOST and INTFLAG.MB will be set in addition to BUSERR.

  Writing the ADDR.ADDR register will automatically clear the BUSERR flag.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear it.

  This bit is not write-synchronized.

27.8.2.8 Syncbusy

Name: SYNCBUSY

Offset: 0x1C

Reset: 0x00000000

Property:

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						SYSOP	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2– SYSOP: System Operation Synchronization Busy**
  Writing CTRLB, STATUS.BUSSTATE, ADDR, or DATA when the SERCOM is enabled requires synchronization. When written, the SYNCBUSY.SYSOP bit will be set until synchronization is complete.

  0: System operation synchronization is not busy.

  1: System operation synchronization is busy.

- **Bit 1 – ENABLE: SERCOM Enable Synchronization Busy**
  Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. When written, the SYNCBUSY.ENABLE bit will be set until synchronization is complete.

  Writes to any register (except for CTRLA.SWRST) while enable synchronization is on-going will be discarded and an APB error will be generated.

  0: Enable synchronization is not busy.

  1: Enable synchronization is busy.

- **Bit 0 – SWRST: Software Reset Synchronization Busy**
  Resetting the SERCOM (CTRLA.SWRST) requires synchronization. When written, the SYNCBUSY.SWRST bit will be set until synchronization is complete.

  Writes to any register while synchronization is on-going will be discarded and an APB error will be generated.

  0: SWRST synchronization is not busy.

  1: SWRST synchronization is busy.

27.8.2.9 Address

Name: ADDR

Offset: 0x24

Reset: 0x0000

Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LEN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TENBITEN	HS	LENEN			ADDR[10:8]		
Access	R/W	R/W	R/W	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:24 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 23:16 – LEN[7:0]: Transaction Length**
  For DMA operation, this field represents the data length of the transaction from 0 to 255 bytes. The transaction length enable (ADDR.LENEN) must be written to 1 for automatic transaction length to be used. After ADDR.LEN bytes have been transmitted or received, a NACK (for master reads) and STOP are automatically generated.

- **Bit 15 – TENBITEN: Ten Bit Addressing Enable**
  This bit enables 10-bit addressing. This bit can be written simultaneously with ADDR to indicate a 10-bit or 7-bit address transmission.

  0: 10-bit addressing disabled.

  1: 10-bit addressing enabled.

- **Bit 14 – HS: High Speed**
  This bit enables High-speed mode for the current transfer from repeated START to STOP. This bit can be written simultaneously with ADDR for a high speed transfer.

0: High-speed transfer disabled.

1: High-speed transfer enabled.

- **Bit 13 – LENEN: Transfer Length Enable**
  This bit enables automatic transfer length.
  0: Automatic transfer length disabled.
  1: Automatic transfer length enabled.

- **Bits 12:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 10:0 – ADDR[10:0]: Address**
  When ADDR is written, the consecutive operation will depend on the bus state:
  Unknown: INTFLAG.MB and STATUS.BUSERR are set, and the operation is terminated.
  Busy: The I^2C master will await further operation until the bus becomes idle.
  Idle: The I^2C master will issue a start condition followed by the address written in ADDR. If the address is acknowledged, SCL is forced and held low, and STATUS.CLKHOLD and INTFLAG.MB are set.
  Owner: A repeated start sequence will be performed. If the previous transaction was a read, the acknowledge action is sent before the repeated start bus condition is issued on the bus. Writing ADDR to issue a repeated start is performed while INTFLAG.MB or INTFLAG.SB is set.
  Regardless of winning or loosing arbitration, the entire address will be sent. If arbitration is lost, only ones are transmitted from the point of loosing arbitration and the rest of the address length.
  STATUS.BUSERR, STATUS.ARBLOST, INTFLAG.MB and INTFLAG.SB will be cleared when ADDR is written.
  The ADDR register can be read at any time without interfering with ongoing bus activity, as a read access does not trigger the master logic to perform any bus protocol related operations.
  The I^2C master control logic uses bit 0 of ADDR as the bus protocol's read/write flag (R/W); 0 for write and 1 for read.

- **Bits 31:24 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 23:16 – LEN[7:0]: Transaction Length**
  For DMA operation, this field represents the data length of the transaction from 0 to 255 bytes. The transaction length enable (ADDR.LENEN) must be written to 1 for automatic transaction length to be used. After ADDR.LEN bytes have been transmitted or received, a NACK (for master reads) and STOP are automatically generated.

- **Bit 15 – TENBITEN: Ten Bit Addressing Enable**
  This bit enables 10-bit addressing. This bit can be written simultaneously with ADDR to indicate a 10-bit or 7-bit address transmission.
  0: 10-bit addressing disabled.
  1: 10-bit addressing enabled.

- **Bit 14 – HS: High Speed**
  This bit enables High-speed mode for the current transfer from repeated START to STOP. This bit can be written simultaneously with ADDR for a high speed transfer.
  0: .High-speed transfer disabled.
  1: High-speed transfer enabled.

- **Bit 13 – LENEN: Transfer Length Enable**
  This bit enables automatic transfer length.
  0: Automatic transfer length disabled.
  1: Automatic transfer length enabled.

- **Bits 12:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 10:0 – ADDR[10:0]: Address**
  When ADDR is written, the consecutive operation will depend on the bus state:
  Unknown: INTFLAG.MB and STATUS.BUSERR are set, and the operation is terminated.
  Busy: The I^2C master will await further operation until the bus becomes idle.
  Idle: The I^2C master will issue a start condition followed by the address written in ADDR. If the address is acknowledged, SCL is forced and held low, and STATUS.CLKHOLD and INTFLAG.MB are set.
  Owner: A repeated start sequence will be performed. If the previous transaction was a read, the acknowledge action is sent before the repeated start bus condition is issued on the bus. Writing ADDR to issue a repeated start is performed while INTFLAG.MB or INTFLAG.SB is set.
  Regardless of winning or loosing arbitration, the entire address will be sent. If arbitration is lost, only ones are transmitted from the point of loosing arbitration and the rest of the address length.
  STATUS.BUSERR, STATUS.ARBLOST, INTFLAG.MB and INTFLAG.SB will be cleared when ADDR is written.
  The ADDR register can be read at any time without interfering with ongoing bus activity, as a read access does not trigger the master logic to perform any bus protocol related operations.
  The I^2C master control logic uses bit 0 of ADDR as the bus protocol's read/write flag (R/W); 0 for write and 1 for read.

27.8.2.10 Data

Name: DATA

Offset: 0x18

Reset: 0x0000

Property: Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:8 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 7:0 – DATA[7:0]: Data**
  The master data register I/O location (DATA) provides access to the master transmit and receive data buffers. Reading valid data or writing data to be transmitted can be successfully done only when SCL is held low by the master (STATUS.CLKHOLD is set). An exception occurs when reading the last data byte after the stop condition has been sent.

  Accessing DATA.DATA auto-triggers I^2C bus operations. The operation performed depends on the state of CTRLB.ACKACT, CTRLB.SMEN and the type of access (read/write).

  Writing or reading DATA.DATA when not in smart mode does not require synchronization.

27.8.2.11 Debug Control

Name: DBGCTRL

Offset: 0x30

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
								DBGSTOP
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – DBGSTOP: Debug Stop Mode**
  This bit controls functionality when the CPU is halted by an external debugger.

  0: The baud-rate generator continues normal operation when the CPU is halted by an external debugger.

  1: The baud-rate generator is halted when the CPU is halted by an external debugger.

28. TC – Timer/Counter

28.1 Overview

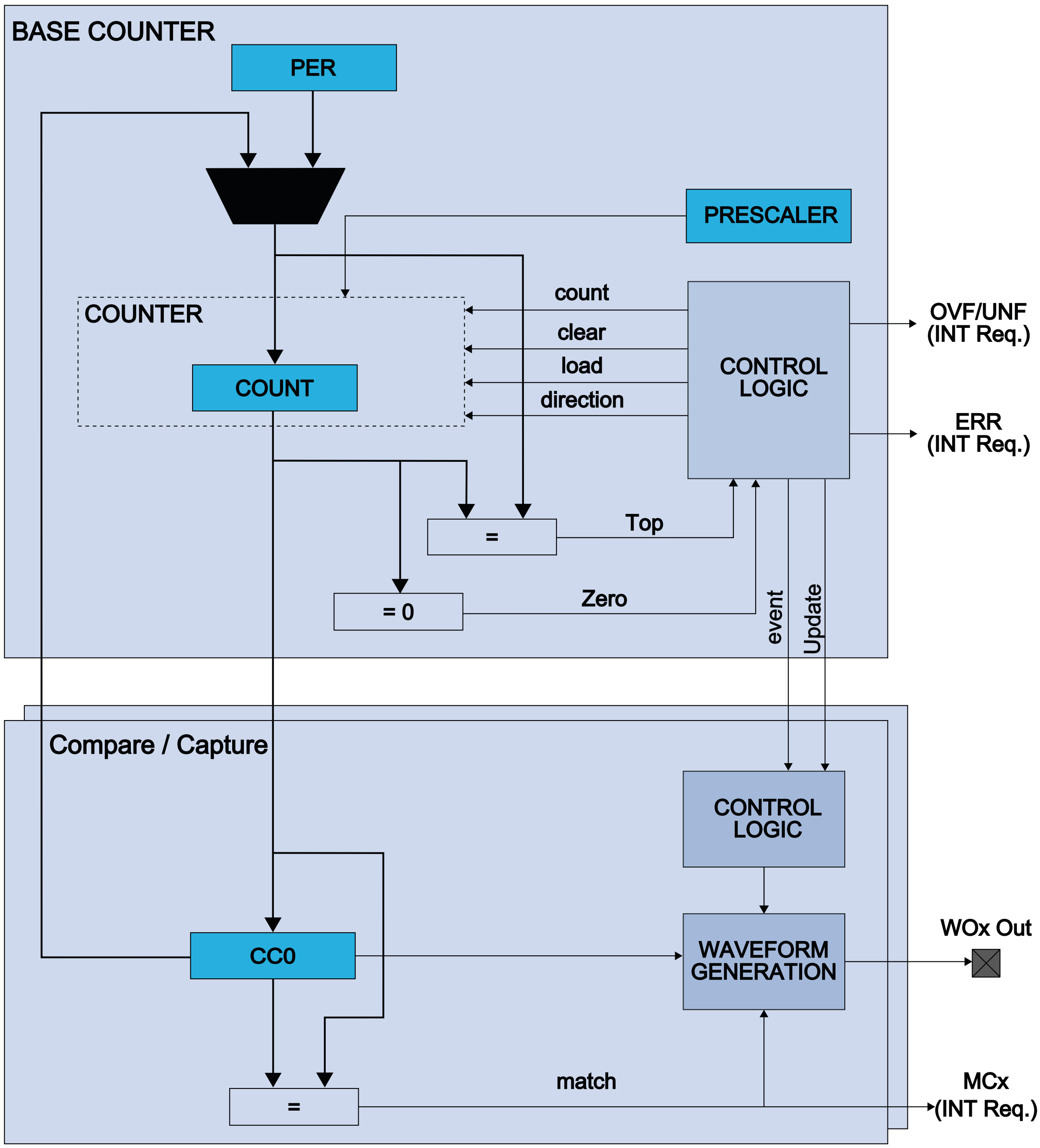
The TC consists of a counter, a prescaler, compare/capture channels and control logic. The counter can be set to count events, or it can be configured to count clock pulses. The counter, together with the compare/capture channels, can be configured to timestamp input events, allowing capture of frequency and pulse width. It can also perform waveform generation, such as frequency generation and pulse-width modulation (PWM).

28.2 Features

- Selectable configuration
  - 8-, 16- or 32-bit TC, with compare/capture channels
- Waveform generation
  - Frequency generation
  - Single-slope pulse-width modulation
- Input capture
  - Event capture
  - Frequency capture
  - Pulse-width capture
- One input event
- Interrupts/output events on:
  - Counter overflow/underflow
  - Compare match or capture
- Internal prescaler

28.3 Block Diagram

Figure 28-1. Timer/Counter Block Diagram



28.4 Signal Description

Signal Name	Type	Description
WO[1:0]	Digital output	Waveform output

Refer to "I/O Multiplexing and Considerations" on page 10 for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

28.5 Product Dependencies

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

28.5.1 I/O Lines

Using the TC's I/O lines requires the I/O pins to be configured. Refer to "PORT" on page 375 for details.

28.5.2 Power Management

The TC can continue to operate in any sleep mode where the selected source clock is running. The TC interrupts can be used to wake up the device from sleep modes. The events can trigger other operations in the system without exiting sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

28.5.3 Clocks

The TC bus clock (CLK_TCx_APB, where x represents the specific TC instance number) can be enabled and disabled in the Power Manager, and the default state of CLK_TCx_APB can be found in the Peripheral Clock Masking section in "PM – Power Manager" on page 107.

The different TC instances are paired, even and odd, starting from TC1, and use the same generic clock, GCLK_TCx. This means that the TC instances in a TC pair cannot be set up to use different GCLK_TCx clocks.

This generic clock is asynchronous to the user interface clock (CLK_TCx_APB). Due to this asynchronicity, accessing certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 581 for further details.

28.5.4 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the TC interrupts requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

28.5.5 Events

To use the TC event functionality, the corresponding events need to be configured in the event system. Refer to "EVSYS – Event System" on page 402 for details.

28.5.6 Debug Operation

When the CPU is halted in debug mode the TC will halt normal operation. The TC can be forced to continue operation during debugging. Refer to the DBGCTRL register for details.

28.5.7 Register Access Protection

All registers with write-access are optionally write-protected by the peripheral access controller (PAC), except the following registers:

- Interrupt Flag register (INTFLAG)

- Status register (STATUS)
- Read Request register (READREQ)
- Count register (COUNT), “Counter Value” on page 603
- Period register (PER), “ Period Value” on page 606
- Compare/Capture Value registers (CCx), “ Compare/Capture” on page 607

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode, all write-protection is automatically disabled.

Write-protection does not apply for accesses through an external debugger. Refer to “PAC – Peripheral Access Controller” on page 27 for details.

28.5.8 Analog Connections

Not applicable.

28.6 Functional Description

28.6.1 Principle of Operation

The counter in the TC can be set to count on events from the Event System, or on the GCLK_TCx frequency. The pulses from GCLK_TCx will go through the prescaler, where it is possible to divide the frequency down.

The value in the counter is passed to the compare/capture channels, where it can either be compared with user defined values or captured on a predefined event.

The TC can be configured as an 8-, 16- or 32-bit counter. Which mode is chosen will determine the maximum range of the counter. The counter range combined with the operating frequency will determine the maximum time resolution achievable with the TC peripheral.

The TC can be count up or down. By default, the counter will operate in a continuous mode and count up, where the counter will wrap to the zero when reaching the top value

When one of the compare/capture channels is used in compare mode, the TC can be used for waveform generation. Upon a match between the counter and the value in one or more of the Compare/Capture Value registers (CCx), one or more output pins on the device can be set to toggle. The CCx registers and the counter can thereby be used in frequency generation and PWM generation.

Capture mode can be used to automatically capture the period and pulse width of signals.

28.6.2 Basic Operation

28.6.2.1 Initialization

The following register is enable-protected, meaning that it can only be written when the TC is disabled (CTRLA.ENABLE is zero):

- Control A register (CTRLA), except the Run Standby (RUNSTDBY), Enable (ENABLE) and Software Reset (SWRST) bits

The following bits are enable-protected:

- Event Action bits in the Event Control register (EVCTRL.EVACT)

Enable-protected bits in the CTRLA register can be written at the same time as CTRLA.ENABLE is written to one, but not at the same time as CTRLA.ENABLE is written to zero.

Before the TC is enabled, it must be configured, as outlined by the following steps:

- The TC bus clock (CLK_TCx_APB) must be enabled
- The mode (8, 16 or 32 bits) of the TC must be selected in the TC Mode bit group in the Control A register (CTRLA.MODE). The default mode is 16 bits

- One of the wavegen modes must be selected in the Waveform Generation Operation bit group in the Control A register (CTRLA.WAVEGEN)
- If the GCLK_TCx frequency used should be prescaled, this can be selected in the Prescaler bit group in the Control A register (CTRLA.PRESCALER)
- If the prescaler is used, one of the presync modes must be chosen in the Prescaler and Counter Synchronization bit group in the Control A register (CTRLA.PRESYNC)
- One-shot mode can be selected by writing a one to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT)
- If the counter should count down from the top value, write a one to the Counter Direction bit in the Control B Set register (CTRLBSET.DIR)
- If capture operations are to be used, the individual channels must be enabled for capture in the Capture Channel x Enable bit group in the Control C register (CTRLC.CPTEN)
- The waveform output for individual channels can be inverted using the Waveform Output Invert Enable bit group in the Control C register (CTRLC.INVEN)

28.6.2.2 Enabling, Disabling and Resetting

The TC is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The TC is disabled by writing a zero to CTRLA.ENABLE.

The TC is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the TC, except DBGCTRL, will be reset to their initial state, and the TC will be disabled. Refer to the CTRLA register for details.

The TC should be disabled before the TC is reset to avoid undefined behavior.

28.6.2.3 Prescaler Selection

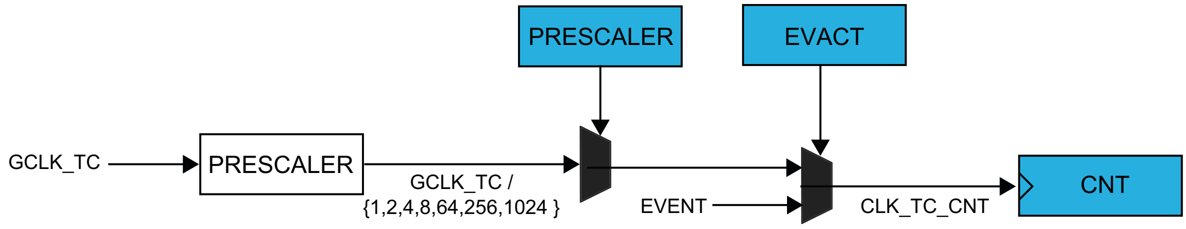
As seen in Figure 28-2, the GCLK_TC clock is fed into the internal prescaler. Prescaler output intervals from 1 to 1/1024 are available. For a complete list of available prescaler outputs, see the register description for the Prescaler bit group in the Control A register (CTRLA.PRESCALER).

The prescaler consists of a counter that counts to the selected prescaler value, whereupon the output of the prescaler toggles.

When the prescaler is set to a value greater than one, it is necessary to choose whether the prescaler should reset its value to zero or continue counting from its current value on the occurrence of an overflow or underflow. It is also necessary to choose whether the TC counter should wrap around on the next GCLK_TC clock pulse or the next prescaled clock pulse (CLK_TC_CNT of Figure 28-2). To do this, use the Prescaler and Counter Synchronization bit group in the Control A register (CTRLA.PRESYNC).

If the counter is set to count events from the event system, these will not pass through the prescaler, as seen in Figure 28-2.

Figure 28-2. Prescaler



28.6.2.4 TC Mode

The counter mode is selected with the TC Mode bit group in the Control A register (CTRLA.MODE). By default, the counter is enabled in the 16-bit counter mode.

Three counter modes are available:

- COUNT8: The 8-bit TC has its own Period register (PER). This register is used to store the period value that can be used as the top value for waveform generation.
- COUNT16: This is the default counter mode. There is no dedicated period register in this mode.
- COUNT32: This mode is achieved by pairing two 16-bit TC peripherals. This pairing is explained in "Clocks" on page 572. The even-numbered TC instance will act as master to the odd-numbered TC peripheral, which will act as a slave. The slave status of the slave is indicated by reading the Slave bit in the Status register (STATUS.SLAVE). The registers of the slave will not reflect the registers of the 32-bit counter. Writing to any of the slave registers will not affect the 32-bit counter. Normal access to the slave COUNT and CCx registers is not allowed.

28.6.2.5 Counter Operations

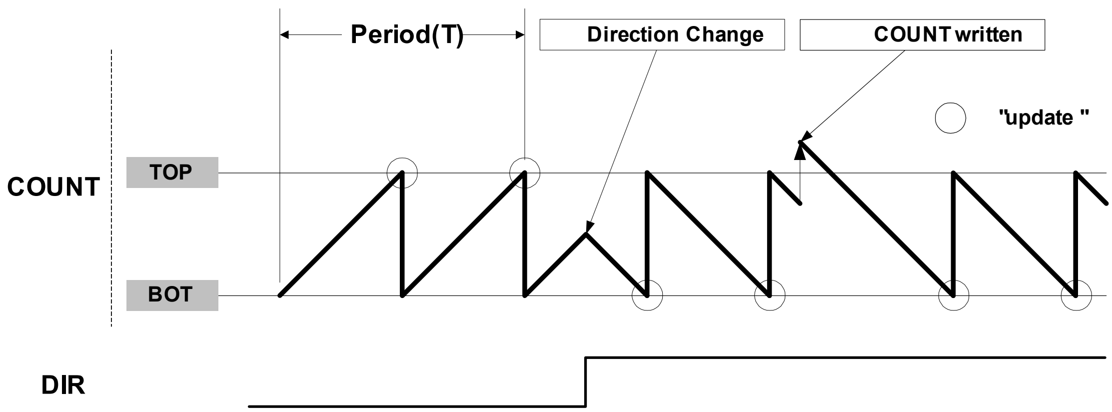
The counter can be set to count up or down. When the counter is counting up and the top value is reached, the counter will wrap around to zero on the next clock cycle. When counting down, the counter will wrap around to the top value when zero is reached. In one-shot mode, the counter will stop counting after a wraparound occurs.

To set the counter to count down, write a one to the Direction bit in the Control B Set register (CTRLBSET.DIR). To count up, write a one to the Direction bit in the Control B Clear register (CTRLBCLR.DIR).

Each time the counter reaches the top value or zero, it will set the Overflow Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF). It is also possible to generate an event on overflow or underflow when the Overflow/Underflow Event Output Enable bit in the Event Control register (EVCTRL.OVFEO) is one.

The counter value can be read from the Counter Value register (COUNT) or a new value can be written to the COUNT register. Figure 28-3 gives an example of writing a new counter value. The COUNT value will always be zero when starting the TC, unless some other value has been written to it or if the TC has been previously reloaded at TOP value, because stopped while TC was counting down.

Figure 28-3. Counter Operation



Stop Command

On the stop command, which can be evoked in the Command bit group in the Control B Set register (CTRLBSET.CMD), the counter will retain its current value. All waveforms are cleared. The counter stops counting, and the Stop bit in the Status register is set (STATUS.STOP).

Retrigger Command and Event Action

Retriggering can be evoked either as a software command, using the Retrigger command in the Control B Set register (CTRLBSET.CMD), or as a retrigger event action, using the Event Action bit group in the Event Control register (EVCTRL.EVACT).

When a retrigger is evoked while the counter is running, the counter will wrap to the top value or zero, depending on the counter direction.

When a retrigger is evoked with the counter stopped, the counter will continue counting from the value in the COUNT register.

Note: When retrigger event action is configured and enabled as an event action, enabling the counter will not start the counter. The counter will start at the next incoming event and restart on any following event.

Count Event Action

When the count event action is configured, every new incoming event will make the counter increment or decrement, depending on the state of the direction bit (CTRLBSET.DIR).

Start Event Action

When the TC is configured with a start event action in the EVCTRL.EVACT bit group, enabling the TC does not make the counter start; the start is postponed until the next input event or software retrigger action. When the counter is running, an input event has not effect on the counter.

28.6.2.6 Compare Operations

When using the TC with the Compare/Capture Value registers (CCx) configured for compare operation, the counter value is continuously compared to the values in the CCx registers. This can be used for timer or waveform operation.

Waveform Output Operations

The compare channels can be used for waveform generation on the corresponding I/O pins. To make the waveform visible on the connected pin, the following requirements must be fulfilled:

- Choose a waveform generation operation
- Optionally, invert the waveform output by writing the corresponding Waveform Output Invert Enable bit in the Control C register (CTRLC.INVx)
- Enable the corresponding multiplexor in the PORT

The counter value is continuously compared with each CCx available. When a compare match occurs, the Match or Capture Channel x interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.MCx) is set on the next zero-to-one transition of CLK_TC_CNT (see Figure 28-4). An interrupt and/or event can be generated on such a condition when INTENSET.MCx and/or EVCTRL.MCEOx is one.

One of four configurations in the Waveform Generation Operation bit group in the Control A register (CTRLA.WAVEGEN) must be chosen to perform waveform generation. This will influence how the waveform is generated and impose restrictions on the top value. The four configurations are:

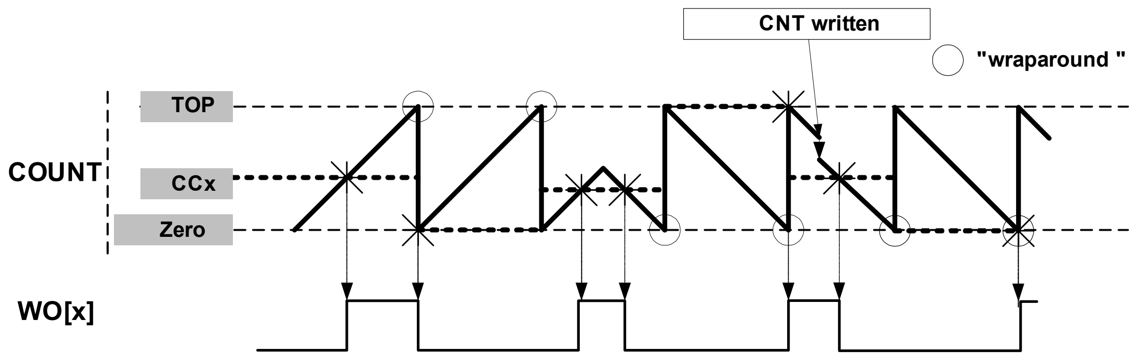
- Normal frequency (NFRQ)
- Match frequency (MFRQ)
- Normal PWM (NPWM)
- Match PWM (MPWM)

When using NPWM or NFRQ, the top value is determined by the counter mode. In 8-bit mode, the Period register (PER) is used as the top value and the top value can be changed by writing to the PER register. In 16- and 32-bit mode, the top value is fixed to the maximum value of the counter.

Frequency Operation

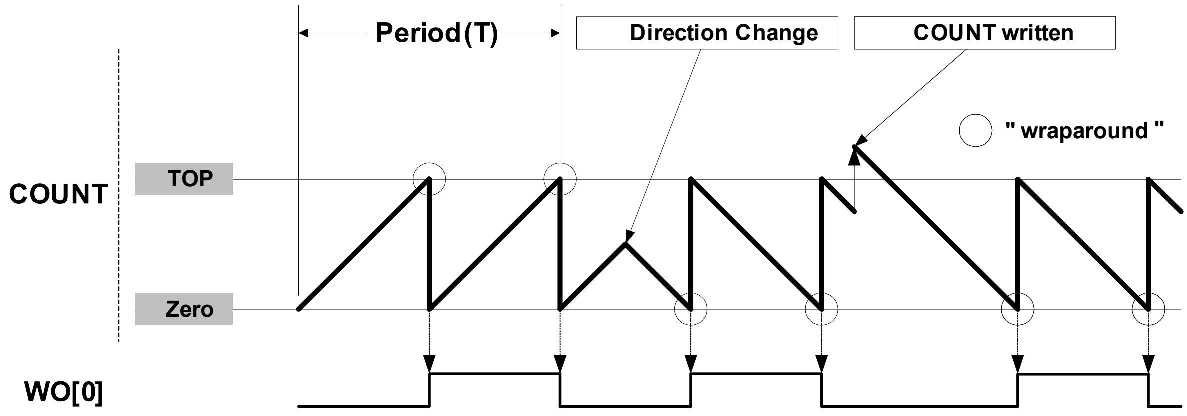
When NFRQ is used, the waveform output (WO[x]) toggles every time CCx and the counter are equal, and the interrupt flag corresponding to that channel will be set.

Figure 28-4. Normal Frequency Operation



When MFRQ is used, the value in CC0 will be used as the top value and WO[0] will toggle on every overflow/underflow.

Figure 28-5. Match Frequency Operation

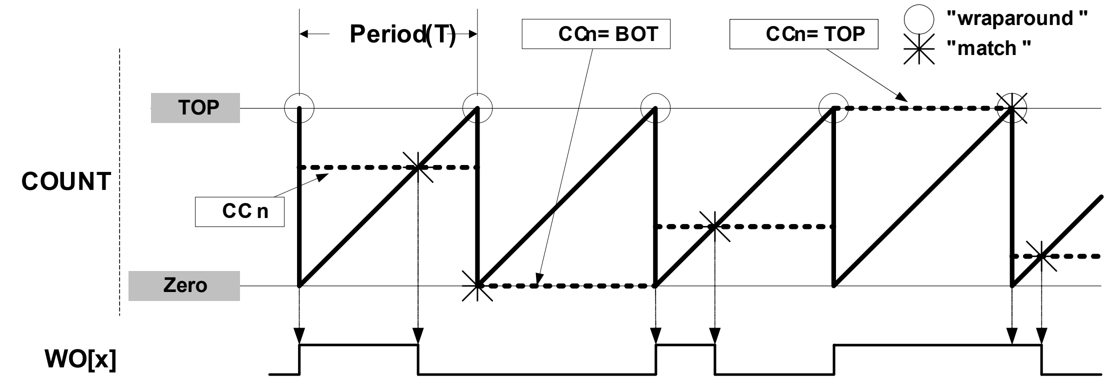


PWM Operation

In PWM operation, the CCx registers control the duty cycle of the waveform generator output. Figure 28-6 shows how in count-up the WO[x] output is set at a start or compare match between the COUNT value and the top value and cleared on the compare match between the COUNT value and CCx register value.

In count-down the WO[x] output is cleared at start or compare match between the COUNT value and the top value and set on the compare match between the COUNT value and CCx register value.

Figure 28-6. Normal PWM Operation



In match operation, Compare/Capture register CC0 is used as the top value, in this case a negative pulse will appear on WO[0] on every overflow/underflow.

The following equation is used to calculate the exact period for a single-slope PWM (R_{PWM_SS}) waveform:

$$R_{\text{PWM\_SS}} = \frac{\log(TOP + 1)}{\log(2)}$$

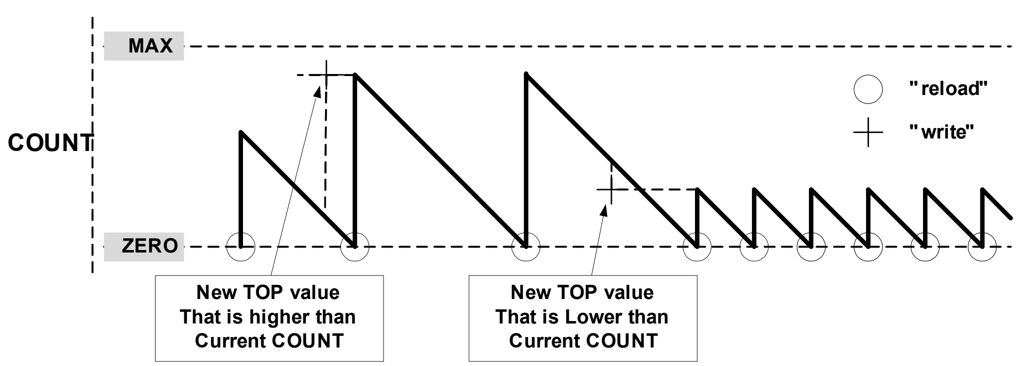
$$f_{\text{PWM\_SS}} = \frac{f_{\text{CLK\_TC}}}{N(TOP + 1)}$$

where N represent the prescaler divider used (1, 2, 4, 8, 16, 64, 256, 1024).

Changing the Top Value

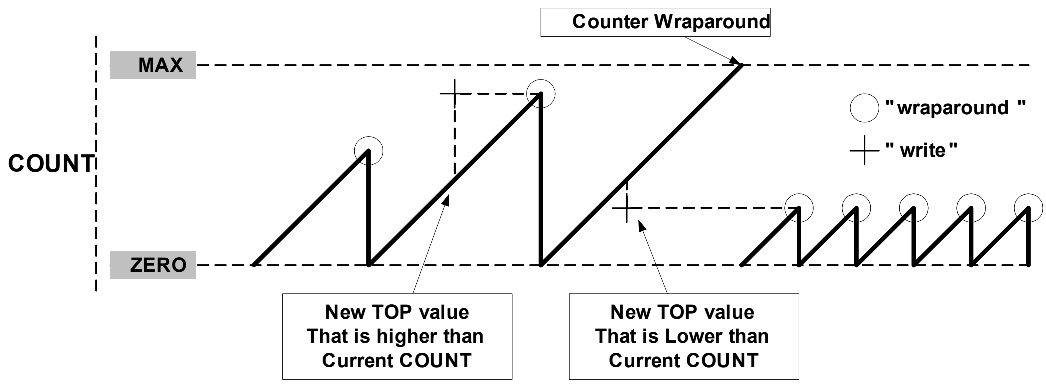
Changing the top value while the counter is running is possible. If a new top value is written when the counter value is close to zero and counting down, the counter can be reloaded with the previous top value, due to synchronization delays. If this happens, the counter will count one extra cycle before the new top value is used.

Figure 28-7. Changing the Top Value when Counting Down



When counting up a change from a top value that is lower relative to the old top value can make the counter miss this change if the counter value is larger than the new top value when the change occurred. This will make the counter count to the max value. An example of this can be seen in Figure 28-8.

Figure 28-8. Changing the Top Value when Counting Up



28.6.2.7 Capture Operations

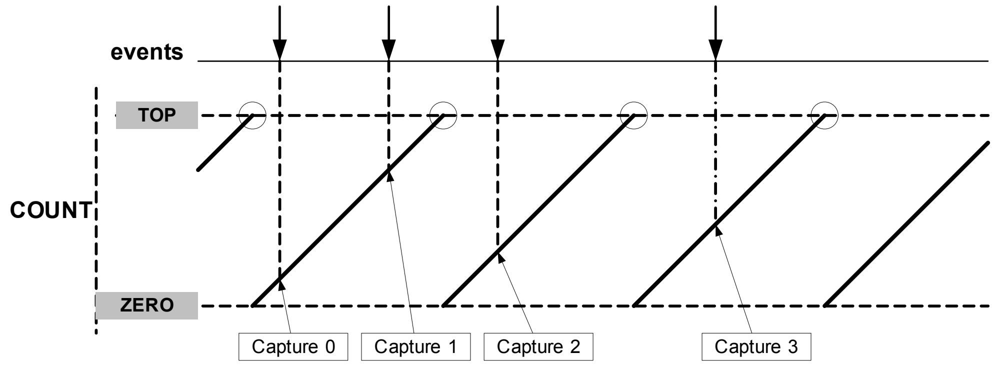
To enable and use capture operations, the event line into the TC must be enabled using the TC Event Input bit in the Event Control register (EVCTRL.TCEI). The capture channels to be used must also be enabled in the Capture Channel x Enable bit group in the Control C register (CTRLC.CPTENx) before capture can be performed.

Event Capture Action

The compare/capture channels can be used as input capture channels to capture any event from the Event System. Because all capture channels use the same event line, only one capture channel should be enabled at a time when performing event capture.

Figure 28-9 shows four capture events for one capture channel.

Figure 28-9. Input Capture Timing



When the Capture Interrupt flag is set and a new capture event is detected, there is nowhere to store the new timestamp. As a result, the Error Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.ERR) is set.

Period and Pulse-Width Capture Action

The TC can perform two input captures and restart the counter on one of the edges. This enables the TC to measure the pulse width and period. This can be used to characterize the frequency and duty cycle of an input signal:

$$f = \frac{1}{T}$$

$$dutyCycle = \frac{t_p}{T}$$

When using PPW event action, the period (T) will be captured into CC0 and the pulse width (t_p) in CC1. In PWP event action, the pulse width (t_p) will be captured in CC0 and the period (T) in CC1.

Selecting PWP (pulse-width, period) or PPW (period, pulse-width) in the Event Action bit group in the Event Control register (EVCTRL.EVACT) enables the TC to performs two capture actions, one on the rising edge and one on the falling edge.

The TC Inverted Event Input in the Event Control register (EVCTRL.TCINV) is used to select whether the wraparound should occur on the rising edge or the falling edge. If EVCTRL.TCINV is written to one, the wraparound will happen on the falling edge. The event source to be captured must be an asynchronous event.

To fully characterize the frequency and duty cycle of the input signal, activate capture on CC0 and CC1 by writing 0x3 to the Capture Channel x Enable bit group in the Control C register (CTRLC.CPTEN). When only one of these measurements is required, the second channel can be used for other purposes.

The TC can detect capture overflow of the input capture channels. When the Capture Interrupt flag is set and a new capture event is detected, there is nowhere to store the new timestamp. Asa result, INTFLAG.ERR is set.

28.6.3 Additional Features

28.6.3.1 One-Shot Operation

When one-shot operation is enabled, the counter automatically stops on the next counter overflow or underflow condition. When the counter is stopped, STATUS.STOP is automatically set by hardware and the waveform outputs are set to zero.

One-shot operation can be enabled by writing a one into the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) and disabled by writing a one to the One-Shot bit in the Control B Clear register (CTRLBCLR.ONESHOT). When enabled, it will count until an overflow or underflow occurs. The one-shot operation can be restarted with a retrigger command, a retrigger event or a start event.

When the counter restarts its operation, the Stop bit in the Status register (STATUS.STOP) is automatically cleared by hardware.

The TC has the following interrupt sources:

- Overflow/Underflow: OVF
- Compare or Capture Channels
- Capture Overflow Error: ERR
- Synchronization Ready: SYNCRDY

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the TC is reset. See the INTFLAG register for details on how to clear interrupt flags.

The TC has one common interrupt request line for all the interrupt sources. The user must read the INTFLAG register to determine which interrupt condition is present. Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to “Nested Vector Interrupt Controller” on page 23 for details.

The TC can generate the following output events:

- Overflow/Underflow (OVF)
- Match or Capture (MC)

Writing a one to an Event Output bit in the Event Control register (EVCTRL.MCEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event.

To enable one of the following event actions, write to the Event Action bit group (EVCTRL.EVACT).

- Start the counter
- Retrigger counter
- Increment or decrement counter (depends on counter direction)
- Capture event
- Capture period
- Capture pulse width

Writing a one to the TC Event Input bit in the Event Control register (EVCTRL.TCEI) enables input events to the TC. Writing a zero to this bit disables input events to the TC. Refer to “EVSYS – Event System” on page 402 for details on configuring the Event System.

28.6.4 Sleep Mode Operation

The TC can be configured to operate in any sleep mode. To be able to run in standby, the RUNSTDBY bit in the Control A register (CTRLA.RUNSTDBY) must be written to one. The TC can wake up the device using interrupts from any sleep mode or perform actions through the Event System.

28.6.5 Synchronization

Due to the asynchronicity between CLK_TCx_APB and GCLK_TCx some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set immediately, and cleared when synchronization is complete. The synchronization Ready interrupt can be used to signal when sync is complete. This can be accessed via the Synchronization Ready Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.SYNCRDY).

If an operation that requires synchronization is executed while STATUS.SYNCBUSY is one, the bus will be stalled. All operations will complete successfully, but the CPU will be stalled and interrupts will be pending as long as the bus is stalled.

The following bits need synchronization when written:

- Software Reset bit in the Control A register (CTRLA.SWRST)
- Enable bit in the Control A register (CTRLA.ENABLE)

Write-synchronization is denoted by the Write-Synchronized property in the register description.

The following registers need synchronization when written:

- Control B Clear register (CTRLBCLR)
- Control B Set register (CTRLBSET)
- Control C register (CTRLC)

- Count Value register (COUNT)
- Period Value register (PERIOD)
- Compare/Capture Value registers (CCx)

Write-synchronization is denoted by the Write-Synchronized property in the register description.

The following registers need synchronization when read:

- Control B Clear register (CTRLBCLR)
- Control B Set register (CTRLBSET)
- Control C register (CTRLC)
- Count Value register (COUNT)
- Period Value register (PERIOD)
- Compare/Capture Value registers (CCx)

Read-synchronization is denoted by the Read-Synchronized property in the register description.

28.7 Register Summary

Table 28-1. Register Summary – 8-Bit Mode Registers

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		WAVEGEN[1:0]			MODE[1:0]		ENABLE	SWRST
0x01		15:8			PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
0x02	READREQ	7:0				ADDR[4:0]				
0x03		15:8	RREQ	RCONT						
0x04	CTRLBCLR	7:0	CMD[1:0]					ONESHOT		DIR
0x05	CTRLBSET	7:0	CMD[1:0]					ONESHOT		DIR
0x06	CTRLC	7:0			CPTEN1	CPTEN0			INVEN1	INVEN0
0x07	Reserved									
0x08	DBGCTRL	7:0								DBGRUN
0x09	Reserved									
0x0A	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
0x0B		15:8			MCEO1	MCEO0				OVFEO
0x0C	INTENCLR	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0D	INTENSET	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0E	INTFLAG	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0F	STATUS	7:0	SYNCBUSY			SLAVE	STOP			
0x10	COUNT	7:0	COUNT[7:0]							
0x11	Reserved									
0x12	Reserved									
0x13	Reserved									
0x14	PER	7:0	PER[7:0]							
0x15	Reserved									
0x16	Reserved									
0x17	Reserved									
0x18	CC0	7:0	CC[7:0]							
0x19	CC1	7:0	CC[7:0]							
0x1A	Reserved									
0x1B	Reserved									
0x1C	Reserved									
0x1D	Reserved									
0x1E	Reserved									
0x1F	Reserved									

Table 28-2. Register Summary – 16-Bit Mode Registers

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		WAVEGEN[1:0]			MODE[1:0]		ENABLE	SWRST
0x01		15:8			PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
0x02	READREQ	7:0				ADDR[4:0]				
0x03		15:8	RREQ	RCONT						
0x04	CTRLBCLR	7:0	CMD[1:0]					ONESHOT		DIR
0x05	CTRLBSET	7:0	CMD[1:0]					ONESHOT		DIR
0x06	CTRLC	7:0			CPTEN1	CPTEN0			INVEN1	INVEN0
0x07	Reserved									
0x08	DBGCTRL	7:0								DBGRUN
0x09	Reserved									
0x0A	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
0x0B		15:8			MCEO1	MCEO0				OVFEO
0x0C	INTENCLR	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0D	INTENSET	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0E	INTFLAG	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0F	STATUS	7:0	SYNCBUSY			SLAVE	STOP			
0x10	COUNT	7:0	COUNT[7:0]							
0x11		15:8	COUNT[15:8]							
0x12	Reserved									
0x13	Reserved									
0x14	Reserved									
0x15	Reserved									
0x16	Reserved									
0x17	Reserved									
0x18	CC0	7:0	CC[7:0]							
0x19		15:8	CC[15:8]							
0x1A	CC1	7:0	CC[7:0]							
0x1B		15:8	CC[15:8]							
0x1C	Reserved									
0x1D	Reserved									
0x1E	Reserved									
0x1F	Reserved									

Table 28-3. Register Summary – 32-Bit Mode Registers

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		WAVEGEN[1:0]			MODE[1:0]		ENABLE	SWRST
0x01		15:8			PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
0x02	READREQ	7:0				ADDR[4:0]				
0x03		15:8	RREQ	RCONT						
0x04	CTRLBCLR	7:0	CMD[1:0]					ONESHOT		DIR
0x05	CTRLBSET	7:0	CMD[1:0]					ONESHOT		DIR
0x06	CTRLC	7:0			CPTEN1	CPTEN0			INVEN1	INVEN0
0x07	Reserved									
0x08	DBGCTRL	7:0								DBGRUN
0x09	Reserved									
0x0A	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
0x0B		15:8			MCEO1	MCEO0				OVFEO
0x0C	INTENCLR	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0D	INTENSET	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0E	INTFLAG	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0F	STATUS	7:0	SYNCBUSY			SLAVE	STOP			
0x10	COUNT	7:0	COUNT[7:0]							
0x11		15:8	COUNT[15:8]							
0x12		23:16	COUNT[23:16]							
0x13		31:24	COUNT[31:24]							
0x14	Reserved									
0x15	Reserved									
0x16	Reserved									
0x17	Reserved									
0x18	CC0	7:0	CC[7:0]							
0x19		15:8	CC[15:8]							
0x1A		23:16	CC[23:16]							
0x1B		31:24	CC[31:24]							
0x1C	CC1	7:0	CC[7:0]							
0x1D		15:8	CC[15:8]							
0x1E		23:16	CC[23:16]							
0x1F		31:24	CC[31:24]							

28.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to the "Register Access Protection" on page 572 and the "PAC – Peripheral Access Controller" on page 27 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Write-Synchronized or Read-Synchronized property in each individual register description. Refer to "Synchronization" on page 581 for details.

Some registers are enable-protected, meaning they can only be written when the TC is disabled. Enable-protection is denoted by the Enable-Protected property in each individual register description.

28.8.1 Control A

Name: CTRLA

Offset: 0x00

Reset: 0x0000

Property: Write-Protected, Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
			PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		WAVEGEN[1:0]			MODE[1:0]		ENABLE	SWRST
Access	R	R/W	R/W	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:14 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 13:12 – PRESCSYNC[1:0]: Prescaler and Counter Synchronization**
  These bits select whether on start or retrigger event the counter should wrap around on the next GCLK_TCx clock or the next prescaled GCLK_TCx clock. It's also possible to reset the prescaler.

  The options are as shown in Table 28-4.

  These bits are not synchronized.

Table 28-4. Prescaler and Counter Synchronization

Value	Name	Description
0x0	GCLK	Reload or reset the counter on next generic clock
0x1	PRESC	Reload or reset the counter on next prescaler clock
0x2	RESYNC	Reload or reset the counter on next generic clock. Reset the prescaler counter
0x3	-	Reserved

- **Bit 11 – RUNSTDBY: Run in Standby**
  This bit is used to keep the TC running in standby mode:

  0: The TC is halted in standby.

  1: The TC continues to run in standby.

  This bit is not synchronized.

- **Bits 10:8 – PRESCALER[2:0]: Prescaler**
  These bits select the counter prescaler factor, as shown in Table 28-5.

  These bits are not synchronized.

Table 28-5. Prescaler

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TC
0x1	DIV2	Prescaler: GCLK_TC/2
0x2	DIV4	Prescaler: GCLK_TC/4
0x3	DIV8	Prescaler: GCLK_TC/8
0x4	DIV16	Prescaler: GCLK_TC/16
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 6:5 – WAVEGEN[1:0]: Waveform Generation Operation**
  These bits select the waveform generation operation. They affect the top value, as shown in "Waveform Output Operations" on page 576. It also controls whether frequency or PWM waveform generation should be used. How these modes differ can also be seen from "Waveform Output Operations" on page 576.

  These bits are not synchronized.

Table 28-6. Waveform Generation Operation

Value	Name	Operation	Top Value	Waveform Output on Match	Waveform Output on Wraparound
0x0	NFRQ	Normal frequency	PER[(1)]/Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER[(1)]/Max	Clear when counting up Set when counting down	Set when counting up Clear when counting down
0x3	MPWM	Match PWM	CC0	Clear when counting up Set when counting down	Set when counting up Clear when counting down

Note: 1. This depends on the TC mode. In 8-bit mode, the top value is the Period Value register (PER). In 16- and 32-bit mode it is the maximum value.

- **Bit 4 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 3:2 – MODE[1:0]: TC Mode**
  These bits select the TC mode, as shown in Table 28-7.

  These bits are not synchronized.

Table 28-7. TC Mode

Value	Name	Description
0x0	COUNT16	Counter in 16-bit mode
0x1	COUNT8	Counter in 8-bit mode
0x2	COUNT32	Counter in 32-bit mode
0x3	-	Reserved

- **Bit 1 – ENABLE: Enable**

  0: The peripheral is disabled.

  1: The peripheral is enabled.

  Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately, and the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set. STATUS.SYNCBUSY will be cleared when the operation is complete.

- **Bit 0 – SWRST: Software Reset**

  0: There is no reset operation ongoing.

  1: The reset operation is ongoing.

  Writing a zero to this bit has no effect.

  Writing a one to this bit resets all registers in the TC, except DBGCTRL, to their initial state, and the TC will be disabled.

  Writing a one to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.

  Due to synchronization there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and STATUS.SYNCBUSY will both be cleared when the reset is complete.

28.8.2 Read Request

For a detailed description of this register and its use, refer to the "Synchronization" on page 581.

Name: READREQ

Offset: 0x02

Reset: 0x0000

Property: -

Bit	15	14	13	12	11	10	9	8
	RREQ	RCONT						
Access	W	R/W	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				ADDR[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 15 – RREQ: Read Request**
  Writing a zero to this bit has no effect.
  This bit will always read as zero.
  Writing a one to this bit requests synchronization of the register pointed to by the Address bit group (READREQ.ADDR) and sets the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY).
- **Bit 14 – RCONT: Read Continuously**
  0: Continuous synchronization is disabled.
  1: Continuous synchronization is enabled.
  When continuous synchronization is enabled, the register pointed to by the Address bit group (READREQ.ADDR) will be synchronized automatically every time the register is updated.
- **Bits 13:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 4:0 – ADDR[4:0]: Address**
  These bits select the offset of the register that needs read synchronization. In the TC, only COUNT and CCx are available for read synchronization.

28.8.3 Control B Clear

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBSET) register.

Name: CTRLBCLR

Offset: 0x04

Reset: 0x00

Property: Write-Protected, Write-Synchronized, Read-Synchronized

Bit	7	6	5	4	3	2	1	0
	CMD[1:0]					ONESHOT		DIR
Access	R/W	R/W	R	R	R	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – CMD[1:0]: Command**
  These bits are used for software control of retriggering and stopping the TC. When a command has been executed, the CMD bit group will read back as zero. The commands are executed on the next prescaled GCLK_TC clock cycle.
  Writing a zero to one of these bits has no effect.
  Writing a one to one of these bits will clear the pending command.

Table 28-8. Command

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	-	Reserved

- **Bits 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – ONESHOT: One-Shot**
  This bit controls one-shot operation of the TC. When in one-shot mode, the TC will stop counting on the next overflow/underflow condition or a stop command.
  0: The TC will wrap around and continue counting on an overflow/underflow condition.
  1: The TC will wrap around and stop on the next underflow/overflow condition.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will disable one-shot operation.

- **Bit 1 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 0 – DIR: Counter Direction**
  This bit is used to change the direction of the counter.
  0: The timer/counter is counting up (incrementing).

1: The timer/counter is counting down (decrementing).

Writing a zero to this bit has no effect.

Writing a one to this bit will make the counter count up.

28.8.4 Control B Set

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBCLR) register.

Name: CTRLBSET

Offset: 0x05

Reset: 0x00

Property: Write-Protected, Write-Synchronized, Read-Synchronized

Bit	7	6	5	4	3	2	1	0
	CMD[1:0]					**ONESHOT**		**DIR**
Access	R/W	R/W	R	R	R	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – CMD[1:0]: Command**
  These bits is used for software control of retriggering and stopping the TC. When a command has been executed, the CMD bit group will be read back as zero. The commands are executed on the next prescaled GCLK_TC clock cycle.

  Writing a zero to one of these bits has no effect.

  Writing a one to one of these bits will set a command.

Table 28-9. Command

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	-	Reserved

- **Bits 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – ONESHOT: One-Shot**
  This bit controls one-shot operation of the TC. When active, the TC will stop counting on the next overflow/underflow condition or a stop command.

  0: The TC will wrap around and continue counting on an overflow/underflow condition.

  1: The timer/counter will wrap around and stop on the next underflow/overflow condition.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will enable one-shot operation.

- **Bit 1 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 0 – DIR: Counter Direction**
  This bit is used to change the direction of the counter.

  0: The timer/counter is counting up (incrementing).

1: The timer/counter is counting down (decrementing).

Writing a zero to this bit has no effect

Writing a one to this bit will make the counter count down.

28.8.5 Control C

Name: CTRLC

Offset: 0x06

Reset: 0x00

Property: Write-Protected, Write-Synchronized, Read-Synchronized

Bit	7	6	5	4	3	2	1	0
			CPTEN1	CPTEN0			INVEN1	INVEN0
Access	R	R	R/W	R/W	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:4 – CPTENx: Capture Channel x Enable**
  These bits are used to select whether channel x is a capture or a compare channel.
  Writing a one to CPTENx enables capture on channel x.
  Writing a zero to CPTENx disables capture on channel x.

- **Bits 3:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 1:0 – INVENx: Waveform Output x Invert Enable**
  These bits are used to select inversion on the output of channel x.
  Writing a one to INVENx inverts the output from WO[x].
  Writing a zero to INVENx disables inversion of the output from WO[x].

28.8.6 Debug Control

Name: DBGCTRL

Offset: 0x08

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – DBGRUN: Debug Run Mode**
  This bit is not affected by a software reset, and should not be changed by software while the TC is enabled.
  0: The TC is halted when the device is halted in debug mode.
  1: The TC continues normal operation when the device is halted in debug mode.

28.8.7 Event Control

Name: EVCTRL

Offset: 0x0A

Reset: 0x0000

Property: Write-Protected, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEO1	MCEO0				OVFEO
Access	R	R	R/W	R/W	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			TCEI	TCINV		EVACT[2:0]		
Access	R	R	R/W	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:14 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 13:12 – MCEOx: Match or Capture Channel x Event Output Enable**
  These bits control whether event match or capture on channel x is enabled or not and generated for every match or capture.

  0: Match/Capture event on channel x is disabled and will not be generated.

  1: Match/Capture event on channel x is enabled and will be generated for every compare/capture.

  These bits are not enable-protected.

- **Bits 11:9 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 8 – OVFEO: Overflow/Underflow Event Output Enable**
  This bit is used to enable the Overflow/Underflow event. When enabled an event will be generated when the counter overflows/underflows.

  0: Overflow/Underflow event is disabled and will not be generated.

  1: Overflow/Underflow event is enabled and will be generated for every counter overflow/underflow.

  This bit is not enable-protected.

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 5 – TCEI: TC Event Input**
  This bit is used to enable input events to the TC.

  0: Incoming events are disabled.

  1: Incoming events are enabled.

  This bit is not enable-protected.

- **Bit 4 – TCINV: TC Inverted Event Input**
  This bit inverts the input event source when used in PWP or PPW measurement.
  0: Input event source is not inverted.
  1: Input event source is inverted.
  This bit is not enable-protected.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bits 2:0 – EVACT[2:0]: Event Action**
  These bits define the event action the TC will perform on an event,as shown in Table 28-10. The EVACT can only be changed while the TC is disabled.

Table 28-10. Event Action

Value	Name	Description
0x0	OFF	Event action disabled
0x1	RETRIGGER	Start, restart or retrigger TC on event
0x2	COUNT	Count on event
0x3	START	Start TC on event
0x4	-	Reserved
0x5	PPW	Period captured in CC0, pulse width in CC1
0x6	PWP	Period captured in CC1, pulse width in CC0
0x7	-	Reserved

28.8.8 Interrupt Enable Clear

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Name: INTENCLR

Offset: 0x0C

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
			MC1	MC0	SYNCRDY		ERR	OVF
Access	R	R/W	R/W	R/W	R/W	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:4 – MCx: Match or Capture Channel x Interrupt Enable**
  0: The Match or Capture Channel x interrupt is disabled.
  1: The Match or Capture Channel x interrupt is enabled.
  Writing a zero to MCx has no effect.
  Writing a one to MCx will clear the corresponding Match or Capture Channel x Interrupt Disable/Enable bit, which disables the Match or Capture Channel x interrupt.

- **Bit 3 – SYNCRDY: Synchronization Ready Interrupt Enable**
  0: The Synchronization Ready interrupt is disabled.
  1: The Synchronization Ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Synchronization Ready Interrupt Disable/Enable bit, which disables the Synchronization Ready interrupt.

- **Bit 2 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 1 – ERR: Error Interrupt Enable**
  0: The Error interrupt is disabled.
  1: The Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Error Interrupt Disable/Enable bit, which disables the Error interrupt.

- **Bit 0 – OVF: Overflow Interrupt Enable**
  0: The Overflow interrupt is disabled.
  1: The Overflow interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Overflow Interrupt Disable/Enable bit, which disables the Overflow interrupt.

28.8.9 Interrupt Enable Set

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Name: INTENSET

Offset: 0x0D

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
			MC1	MC0	SYNCRDY		ERR	OVF
Access	R	R	R/W	R/W	R/W	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:4 – MCx: Match or Capture Channel x Interrupt Enable**
  0: The Match or Capture Channel x interrupt is disabled.
  1: The Match or Capture Channel x interrupt is enabled.
  Writing a zero to MCx has no effect.
  Writing a one to MCx will set the corresponding Match or Capture Channel x Interrupt Enable bit, which enables the Match or Capture Channel x interrupt.

- **Bit 3 – SYNCRDY: Synchronization Ready Interrupt Enable**
  0: The Synchronization Ready interrupt is disabled.
  1: The Synchronization Ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Synchronization Ready Interrupt Disable/Enable bit, which enables the Synchronization Ready interrupt.

- **Bit 2 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 1 – ERR: Error Interrupt Enable**
  0: The Error interrupt is disabled.
  1: The Error interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Error Interrupt bit, which enables the Error interrupt.

- **Bit 0 – OVF: Overflow Interrupt Enable**
  0: The Overflow interrupt is disabled.
  1: The Overflow interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt.

28.8.10 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x0E

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
			MC1	MC0	SYNCRDY		ERR	OVF
Access	R	R	R/W	R/W	R/W	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:4 – MCx: Match or Capture Channel x**
  This flag is set on the next CLK_TC_CNT cycle after a match with the compare condition or once CCx register contain a valid capture value, and will generate an interrupt request if the corresponding Match or Capture Channel x Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.MCx) is one.

  Writing a zero to one of these bits has no effect.

  Writing a one to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag

  In capture mode, this flag is automatically cleared when CCx register is read.

- **Bit 3 – SYNCRDY: Synchronization Ready**
  This flag is set on a 1-to-0 transition of the Synchronization Busy bit in the Status register (STATUS.SYNC-BUSY), except when the transition is caused by an enable or software reset, and will generate an interrupt request if the Synchronization Ready Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.SYN-CRDY) is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will clear the Synchronization Ready interrupt flag

- **Bit 2 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 1 – ERR: Error**
  This flag is set if a new capture occurs on a channel when the corresponding Match or Capture Channel x interrupt flag is one, in which case there is nowhere to store the new capture.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the Error interrupt flag.

- **Bit 0 – OVF: Overflow**
  This flag is set on the next CLK_TC_CNT cycle after an overflow condition occurs, and will generate an interrupt if

  INTENCLR/SET.OVF is one.

  Writing a zero to this bit has no effect.

  Writing a one to this bit clears the Overflow interrupt flag.

28.8.11 Status

Name: STATUS

Offset: 0x0F

Reset: 0x08

Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCBUSY			SLAVE	STOP			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	0

- **Bit 7 – SYNCBUSY: Synchronization Busy**
  This bit is cleared when the synchronization of registers between the clock domains is complete.
  This bit is set when the synchronization of registers between clock domains is started.

- **Bits 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 4 – SLAVE: Slave**
  This bit is set when the even-numbered master TC is set to run in 32-bit mode. The odd-numbered TC will be the slave.

- **Bit 3 – STOP: Stop**
  This bit is set when the TC is disabled, on a Stop command or on an overflow or underflow condition when the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) is one.
  0: Counter is running.
  1: Counter is stopped.

- **Bits 2:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

28.8.12 Counter Value

28.8.12.1 8-Bit Mode

Name: COUNT

Offset: 0x10

Reset: 0x00

Property: Write-Synchronized, Read-Synchronized

Bit	7	6	5	4	3	2	1	0
	COUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:0 – COUNT[7:0]: Counter Value**
  These bits contain the current counter value.

28.8.12.2 16-Bit Mode

Name: COUNT

Offset: 0x10

Reset: 0x0000

Property: Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
	COUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – COUNT[15:0]: Counter Value**
  These bits contain the current counter value.

28.8.12.3 32-Bit Mode

Name: COUNT

Offset: 0x10

Reset: 0x00000000

Property: Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
	COUNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COUNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – COUNT[31:0]: Counter Value**
  These bits contain the current counter value.

28.8.13 Period Value

The Period Value register is available only in 8-bit TC mode. It is not available in 16-bit and 32-bit TC modes.

28.8.13.1 8-Bit Mode

Name: PER

Offset: 0x14

Reset: 0xFF

Property: Write-Synchronized, Read-Synchronized

Bit	7	6	5	4	3	2	1	0
	PER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

- **Bits 7:0 – PER[7:0]: Period Value**
  These bits contain the counter period value in 8-bitTC mode.

28.8.14 Compare/Capture

28.8.14.1 8-Bit Mode

Name: CCx

Offset: 0x18+i*0x1 [i=0..3]

Reset: 0x00

Property: Write-Synchronized, Read-Synchronized

Bit	7	6	5	4	3	2	1	0
	CC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:0 – CC[7:0]: Compare/Capture Value**
  These bits contain the compare/capture value in 8-bit TC mode. In frequency or PWM waveform match operation (CTRLA.WAVEGEN), the CC0 register is used as a period register.

28.8.14.2 16-Bit Mode

Name: CCx

Offset: 0x18+i*0x2 [i=0..3]

Reset: 0x0000

Property: Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
	CC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – CC[15:0]: Compare/Capture Value**
  These bits contain the compare/capture value in 16-bit TC mode. In frequency or PWM waveform match operation (CTRLA.WAVEGEN), the CC0 register is used as a period register.

28.8.14.3 32-Bit Mode

Name: CCx

Offset: 0x18+i*0x4 [i=0..3]

Reset: 0x00000000

Property: Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
	CC[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CC[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – CC[31:0]: Compare/Capture Value**

  These bits contain the compare/capture value in 32-bit TC mode. In frequency or PWM waveform match operation (CTRLA.WAVEGEN), the CC0 register is used as a period register.

29. ADC – Analog-to-Digital Converter

29.1 Overview

The Analog-to-Digital Converter (ADC) converts analog signals to digital values. The ADC has 12-bit resolution, and is capable of converting up to 350ksps. The input selection is flexible, and both differential and single-ended measurements can be performed. An optional gain stage is available to increase the dynamic range. In addition, several internal signal inputs are available. The ADC can provide both signed and unsigned results.

ADC measurements can be started by either application software or an incoming event from another peripheral in the device. ADC measurements can be started with predictable timing, and without software intervention.

Both internal and external reference voltages can be used.

An integrated temperature sensor is available for use with the ADC. The bandgap voltage as well as the scaled I/O and core voltages can also be measured by the ADC.

The ADC has a compare function for accurate monitoring of user-defined thresholds, with minimum software intervention required.

The ADC may be configured for 8-, 10- or 12-bit results, reducing the conversion time. ADC conversion results are provided left- or right-adjusted, which eases calculation when the result is represented as a signed value.

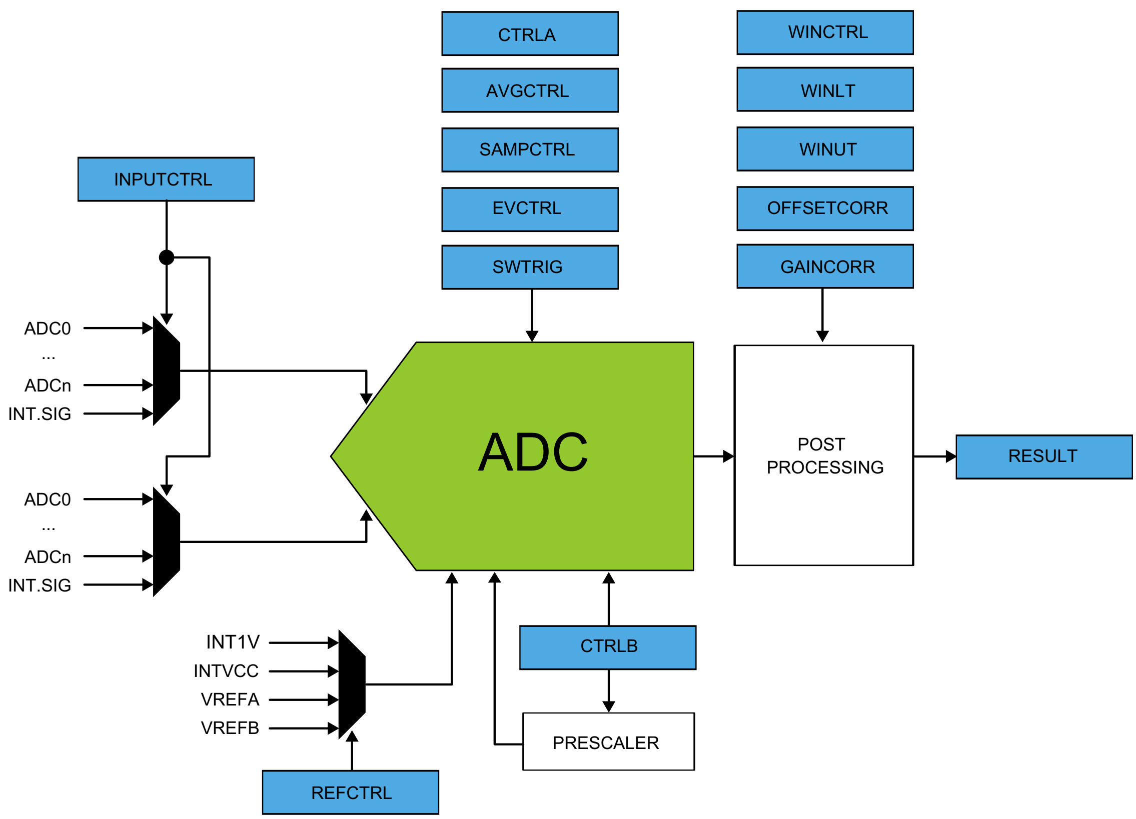
29.2 Features

- 8-, 10- or 12-bit resolution
- Up to 350,000 samples per second (350ksps)
- Differential and single-ended inputs
  - Up to 32 analog inputs
    - 25 positive and 10 negative, including internal and external
- Five internal inputs
  - Bandgap
  - Temperature sensor
  - Scaled core supply
  - Scaled I/O supply
- 1/2x to 16x gain
- Single, continuous and pin-scan conversion options
- Windowing monitor with selectable channel
- Conversion range:
  - V_{ref} [1v to V_{DDANA} - 0.6V]
  - ADCx * GAIN [0V to -V_{ref}]
- Built-in internal reference and external reference options
  - Four bits for reference selection

- Event-triggered conversion for accurate timing (one event input)
- Hardware gain and offset compensation
- Averaging and oversampling with decimation to support, up to 16-bit result
- Selectable sampling time

29.3 Block Diagram

Figure 29-1. ADC Block Diagram



29.4 Signal Description

Signal Name	Type	Description
VREFA	Analog input	External reference voltage A
VREFB	Analog input	External reference voltage B
ADC[19..0][(1)]	Analog input	Analog input channels

Note: 1. Refer to "Configuration Summary" on page 3 for details on exact number of analog input channels.

Refer to "I/O Multiplexing and Considerations" on page 10 for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

29.5 Product Dependencies

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

29.5.1 I/O Lines

Using the ADC's I/O lines requires the I/O pins to be configured using the port configuration (PORT).

Refer to "PORT" on page 375 for details.

29.5.2 Power Management

The ADC will continue to operate in any sleep mode where the selected source clock is running. The ADC's interrupts can be used to wake up the device from sleep modes. The events can trigger other operations in the system without exiting the sleep modes. Refer to "PM – Power Manager" on page 107 for details on the different sleep modes.

29.5.3 Clocks

The ADC bus clock (CLK_ADC_APB) can be enabled and disabled in the Power Manager, and the default state of CLK_ADC_APB can be found in the Table 15-1.

A generic clock (GCLK_ADC) is required to clock the ADC. This clock must be configured and enabled in the Generic Clock Controller (GCLK) before using the ADC. Refer "GCLK – Generic Clock Controller" on page 85 for details.

This generic clock is asynchronous to the bus clock (CLK_ADC_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 620 for further details.

29.5.4 Interrupts

The interrupt request line is connected to the interrupt controller. Using ADC interrupts requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 23 for details.

29.5.5 Events

Events are connected to the Event System. Refer to "EVSYS – Event System" on page 402 for details.

29.5.6 Debug Operation

When the CPU is halted in debug mode, the ADC will halt normal operation. The ADC can be forced to continue operation during debugging. Refer to the Debug Control register (DBGCTRL) for details.

29.5.7 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following register:

- Interrupt Flag Status and Clear register (INTFLAG)

Write-protection is denoted by the Write-Protection property in the register description.

When the CPU is halted in debug mode or the CPU reset is extended, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger. Refer to "PAC – Peripheral Access Controller" on page 27 for details.

29.5.8 Analog Connections

I/O-pins AIN0 to AIN19 as well as the VREFA/VREFB reference voltage pin are analog inputs to the ADC.

29.5.9 Calibration

The values BIAS_CAL and LINEARITY_CAL from the production test must be loaded from the NVM Software Calibration Area into the ADC Calibration register (CALIB) by software to achieve specified accuracy.

Refer to "NVM Software Calibration Row Mapping" on page 21 for more details.

29.6 Functional Description

29.6.1 Principle of Operation

By default, the ADC provides results with 12-bit resolution. 8-bit or 10-bit results can be selected in order to reduce the conversion time. The ADC has an oversampling with decimation option that can extend the resolution to 16 bits. The input values can be either internal (e.g., internal temperature sensor) or external (connected I/O pins). The user can also configure whether the conversion should be single-ended or differential.

29.6.2 Basic Operation

29.6.2.1 Initialization

Before enabling the ADC, the asynchronous clock source must be selected and enabled, and the ADC reference must be configured. The first conversion after the reference is changed must not be used. All other configuration registers must be stable during the conversion. The source for GCLK_ADC is selected and enabled in the System Controller (SYSCTRL). Refer to "SYSCTRL – System Controller" on page 140 for more details.

When GCLK_ADC is enabled, the ADC can be enabled by writing a one to the Enable bit in the Control Register A (CTRLA.ENABLE).

29.6.2.2 Enabling, Disabling and Reset

The ADC is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The ADC is disabled by writing a zero to CTRLA.ENABLE.

The ADC is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the ADC, except DBGCTRL, will be reset to their initial state, and the ADC will be disabled. Refer to the CTRLA register for details.

The ADC must be disabled before it is reset.

29.6.2.3 Basic Operation

In the most basic configuration, the ADC sample values from the configured internal or external sources (INPUTCTRL register). The rate of the conversion is dependent on the combination of the GCLK_ADC frequency and the clock prescaler.

To convert analog values to digital values, the ADC needs first to be initialized, as described in "Initialization" on page 613. Data conversion can be started either manually, by writing a one to the Start bit in the Software Trigger register (SWTRIG.START), or automatically, by configuring an automatic trigger to initiate the conversions. A free-running mode could be used to continuously convert an input channel. There is no need for a trigger to start the conversion. It will start automatically at the end of previous conversion.

The automatic trigger can be configured to trigger on many different conditions.

The result of the conversion is stored in the Result register (RESULT) as it becomes available, overwriting the result from the previous conversion.

To avoid data loss if more than one channel is enabled, the conversion result must be read as it becomes available (INTFLAG.RESRDY). Failing to do so will result in an overrun error condition, indicated by the OVERRUN bit in the Interrupt Flag Status and Clear register (INTFLAG.OVERRUN).

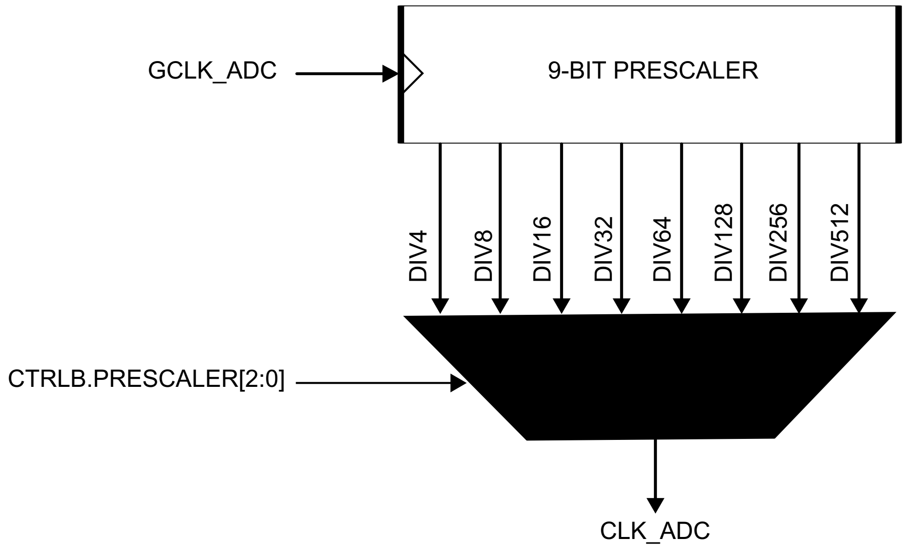
To use an interrupt handler, the corresponding bit in the Interrupt Enable Set register (INTENSET) must be written to one.

29.6.3 Prescaler

The ADC is clocked by GCLK_ADC. There is also a prescaler in the ADC to enable conversion at lower clock rates.

Refer to CTRLB for details on prescaler settings.

Figure 29-2. ADC Prescaler



The propagation delay of an ADC measurement depends on the selected mode and is given by:

- Single-shot mode:

$$PropagationDelay = \frac{1 + \frac{Resolution}{2} + DelayGain}{f_{CLK-ADC}}$$

- Free-running mode:

$$PropagationDelay = \frac{\frac{Resolution}{2} + DelayGain}{f_{CLK-ADC}}$$

Table 29-1. Delay Gain

		Delay Gain (in CLK_ADC Period)			
		Free-running mode		Single shot mode	
Name	INTPUTCTRL.GAIN[3:0]	Differential Mode	Single-Ended Mode	Differential mode	Single-Ended mode
1X	0x0	0	0	0	1
2X	0x1	0	1	0.5	1.5
4X	0x2	1	1	1	2
8X	0x3	1	2	1.5	2.5
16X	0x4	2	2	2	3
Reserved	0x5 ... 0xE	Reserved	Reserved	Reserved	Reserved
DIV2	0xF	0	1	0.5	1.5

29.6.4 ADC Resolution

The ADC supports 8-bit, 10-bit and 12-bit resolutions. Resolution can be changed by writing the Resolution bit group in the Control B register (CTRLB.RESSEL). After a reset, the resolution is set to 12 bits by default.

29.6.5 Differential and Single-Ended Conversions

The ADC has two conversion options: differential and single-ended. When measuring signals where the positive input is always at a higher voltage than the negative input, the single-ended conversion should be used in order to have full 12-bit resolution in the conversion, which has only positive values. The negative input must be connected to ground. This ground could be the internal GND, IOGND or an external ground connected to a pin. Refer to INPUTCTRL for selection details. If the positive input may go below the negative input, creating some negative results, the differential mode should be used in order to get correct results. The configuration of the conversion is done in the Differential Mode bit in the Control B register (CTRLB.DIFFMODE). These two types of conversion could be run in single mode or in free-running mode. When set up in free-running mode, an ADC input will continuously sample and do new conversions. The INTFLAG.RESRDY bit will be set at the end of each conversion.

29.6.5.1 Conversion Timing

Figure 29-3 shows the ADC timing for a single conversion without gain. The writing of the ADC Start Conversion bit (SWTRIG.START) or Start Conversion Event In bit (EVCTRL.STARTEI) must occur at least one CLK_ADC_APB cycle before the CLK_ADC cycle on which the conversion starts. The input channel is sampled in the first half CLK_ADC period. The sampling time can be increased by using the Sampling Time Length bit group in the Sampling Time Control register (SAMPCTRL.SAMPLEN). Refer to Figure 29-4 for example on increased sampling time.

Figure 29-3. ADC Timing for One Conversion in Differential Mode without Gain

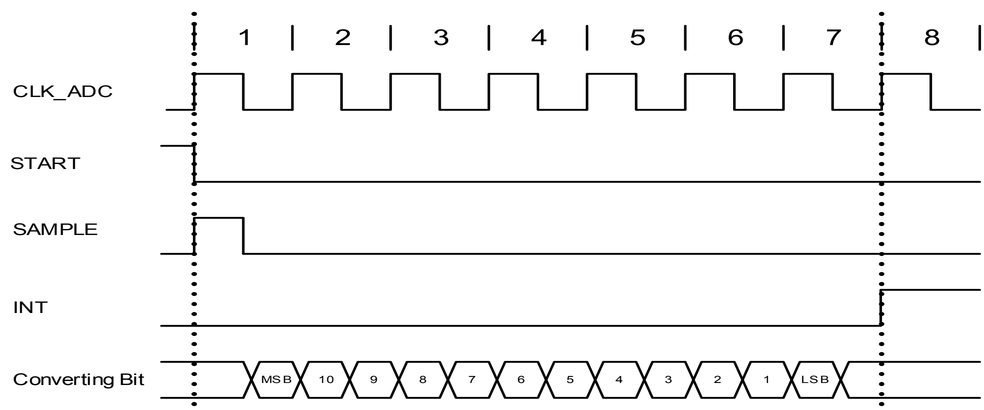


Figure 29-4. ADC Timing for One Conversion in Differential Mode without Gain, but with Increased Sampling Time

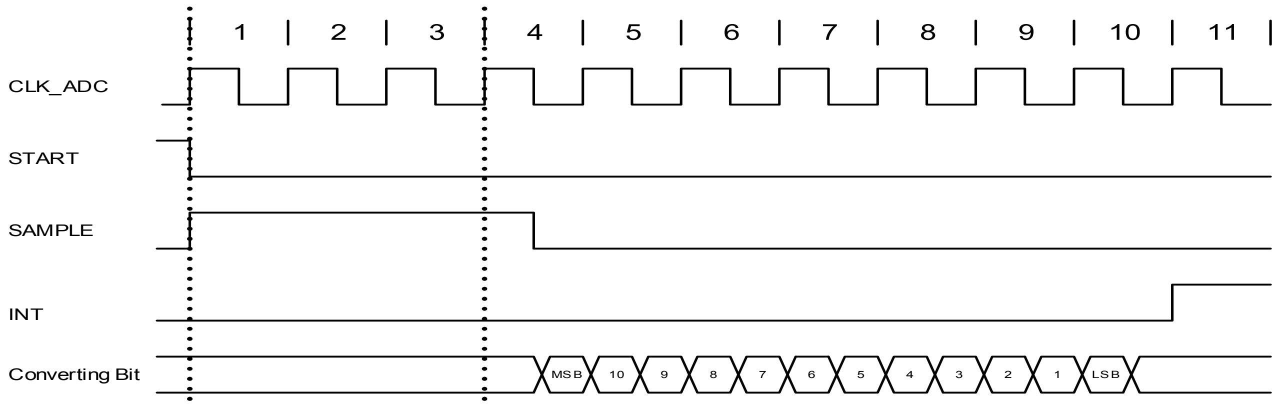


Figure 29-5. ADC Timing for Free Running in Differential Mode without Gain

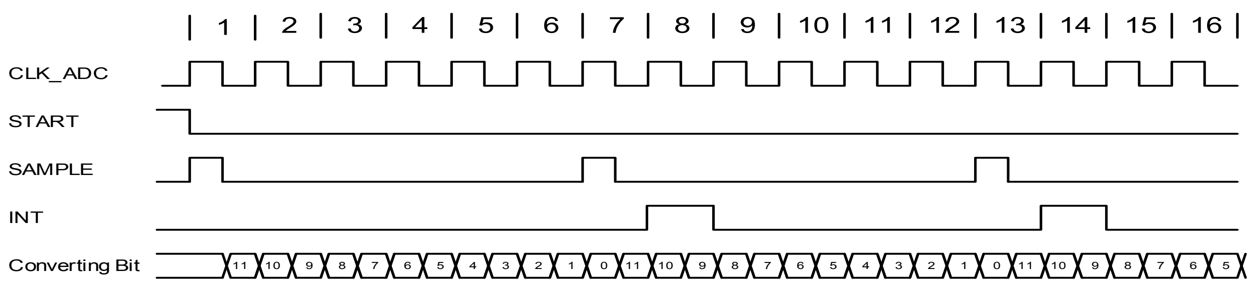


Figure 29-6. ADC Timing for One Conversion in Single-Ended Mode without Gain

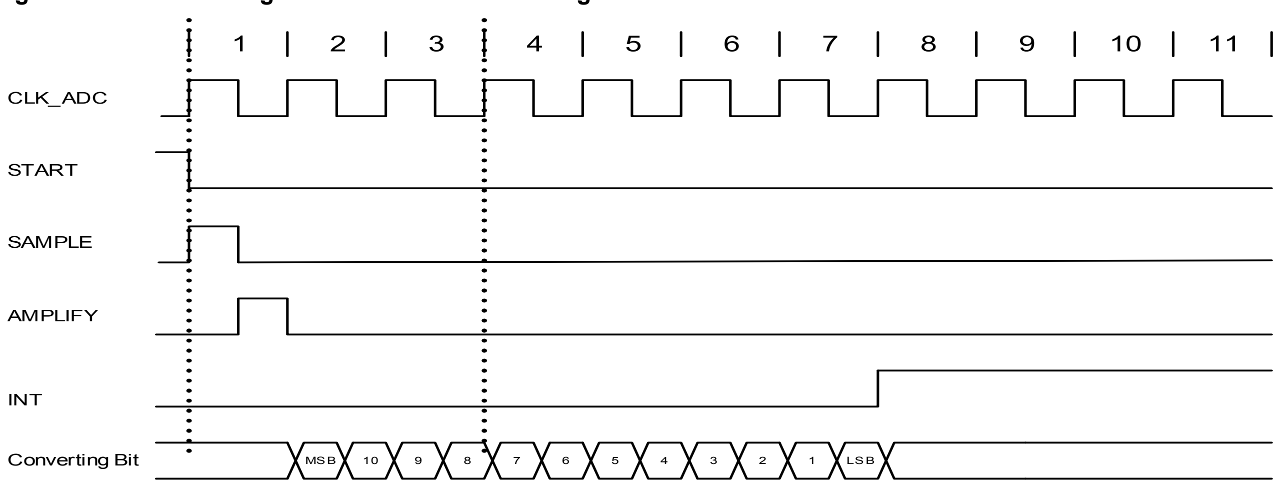
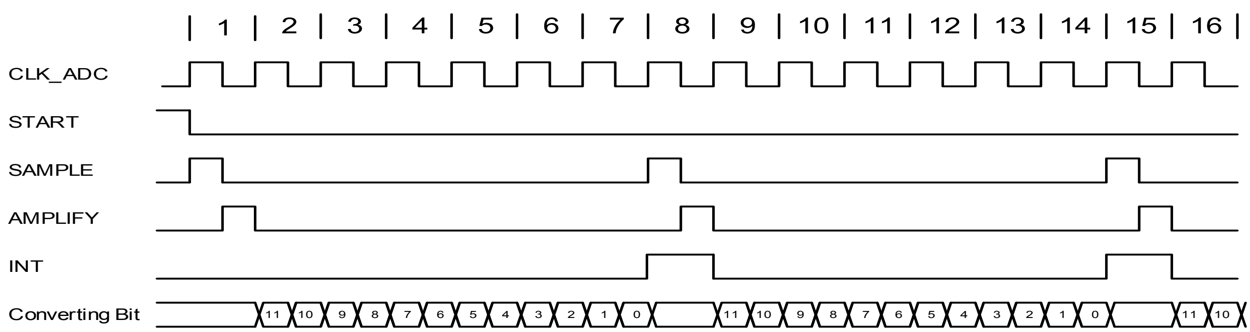


Figure 29-7. ADC Timing for Free Running in Single-Ended Mode without Gain



29.6.6 Accumulation

The result from multiple consecutive conversions can be accumulated. The number of samples to be accumulated is specified by writing to the Number of Samples to be Collected field in the Average Control register (AVGCTRL.SAMPLENUM) as described in Table . When accumulating more than 16 samples, the result will be too large for the 16-bit RESULT register. To avoid overflow, the result is shifted right automatically to fit within the 16 available bits. The number of automatic right shifts are specified in Table . Note that to be able to perform the accumulation of two or more samples, the Conversion Result Resolution field in the Control B register (CTRLB.RESSEL) must be written to one.

Table 29-2. Accumulation

Number of Accumulated Samples	AVGCTRL. SAMPLENUM	Intermediate Result Precision	Number of Automatic Right Shifts	Final Result Precision	Automatic Division Factor
1	0x0	12 bits	0	12 bits	0
2	0x1	13 bits	0	13 bits	0
4	0x2	14 bits	0	14 bits	0
8	0x3	15 bits	0	15 bits	0
16	0x4	16 bits	0	16 bits	0
32	0x5	17 bits	1	16 bits	2
64	0x6	18 bits	2	16 bits	4
128	0x7	19 bits	3	16 bits	8
256	0x8	20 bits	4	16 bits	16
512	0x9	21 bits	5	16 bits	32
1024	0xA	22 bits	6	16 bits	64
Reserved	0xB –0xF	12 bits		12 bits	0

29.6.7 Averaging

Averaging is a feature that increases the sample accuracy, though at the cost of reduced sample rate. This feature is suitable when operating in noisy conditions. Averaging is done by accumulating *m* samples, as described in

"Accumulation" on page 617, and divide the result by *m*. The averaged result is available in the RESULT register. The number of samples to be accumulated is specified by writing to AVGCTRL.SAMPLENUM as described in Table . The division is obtained by a combination of the automatic right shift described above, and an additional right shift that must be specified by writing to the Adjusting Result/Division Coefficient field in AVGCTRL (AVGCTRL.ADJRES) as described in Table . Note that to be able to perform the averaging of two or more samples, the Conversion Result Resolution field in the Control B register (CTRLB.RESSEL) must be written to one.

Averaging AVGCTRL.SAMPLENUM samples will reduce the effective sample rate by $\frac{1}{\text{AVGCTRL.SAMPLENUM}}$.

When the required average is reached, the INTFLAG.RESRDY bit is set.

Table 29-3. Averaging

Number of Accumulated Samples	AVGCTRL. SAMPLENUM	Intermediate Result Precision	Number of Automatic Right Shifts	Division Factor	AVGCTRL. ADJRES	Total Number of Right Shifts	Final Result Precision	Automatic Division Factor
1	0x0	12 bits	0	1	0x0		12 bits	0
2	0x1	13	0	2	0x1	1	12 bits	0
4	0x2	14	0	4	0x2	2	12 bits	0
8	0x3	15	0	8	0x3	3	12 bits	0
16	0x4	16	0	16	0x4	4	12 bits	0
32	0x5	17	1	16	0x4	5	12 bits	2
64	0x6	18	2	16	0x4	6	12 bits	4
128	0x7	19	3	16	0x4	7	12 bits	8
256	0x8	20	4	16	0x4	8	12 bits	16
512	0x9	21	5	16	0x4	9	12 bits	32
1024	0xA	22	6	16	0x4	10	12 bits	64
Reserved	0xB –0xF				0x0		12 bits	0

29.6.8 Oversampling and Decimation

By using oversampling and decimation, the ADC resolution can be increased from 12 bits to up to 16 bits. To increase the resolution by *n* bits, 4^n samples must be accumulated. The result must then be shifted right by *n* bits. This right shift is a combination of the automatic right shift and the value written to AVGCTRL.ADJRES. To obtain the correct resolution, the ADJRES must be configured as described in the table below. This method will result in *n* bit extra LSB resolution.

Table 29-4. Configuration Required for Oversampling and Decimation

Result Resolution	Number of Samples to Average	AVGCTRL.SAMPLENUM[3:0]	Number of Automatic Right Shifts	AVGCTRL.ADJRES[2:0]
13 bits	$4^1 = 4$	0x2	0	0x1
14 bits	$4^2 = 16$	0x4	0	0x2
15 bits	$4^3 = 64$	0x6	2	0x1
16 bits	$4^4 = 256$	0x8	4	0x0

29.6.9 Window Monitor

The window monitor allows the conversion result to be compared to some predefined threshold values. Supported modes are selected by writing the Window Monitor Mode bit group in the Window Monitor Control register (WINCTRL.WINMODE[2:0]). Thresholds are given by writing the Window Monitor Lower Threshold register (WINLT) and Window Monitor Upper Threshold register (WINUT).

If differential input is selected, the WINLT and WINUT are evaluated as signed values. Otherwise they are evaluated as unsigned values.

Another important point is that the significant WINLT and WINUT bits are given by the precision selected in the Conversion Result Resolution bit group in the Control B register (CTRLB.RESSEL). This means that if 8-bit mode is selected, only the eight lower bits will be considered. In addition, in differential mode, the eighth bit will be considered as the sign bit even if the ninth bit is zero.

The INTFLAG.WINMON interrupt flag will be set if the conversion result matches the window monitor condition.

29.6.10 Offset and Gain Correction

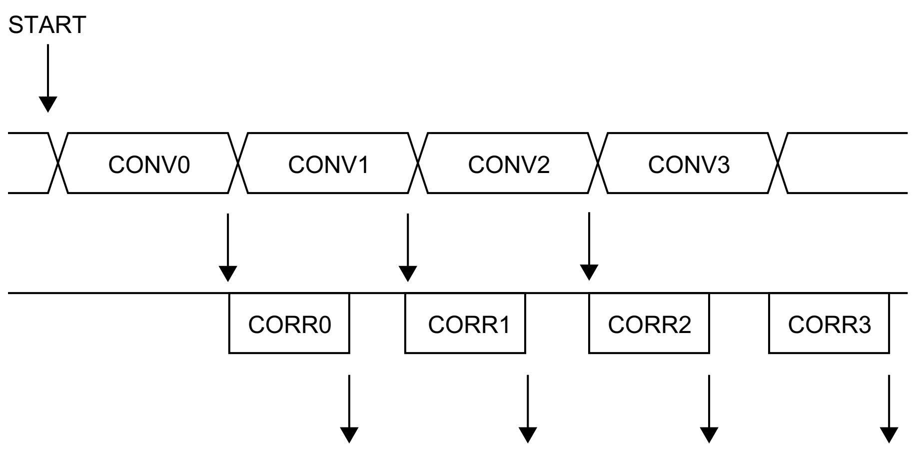
Inherent gain and offset errors affect the absolute accuracy of the ADC. The offset error is defined as the deviation of the actual ADC's transfer function from an ideal straight line at zero input voltage. The offset error cancellation is handled by the Offset Correction register (OFFSETCORR). The offset correction value is subtracted from the converted data before writing the Result register (RESULT). The gain error is defined as the deviation of the last output step's midpoint from the ideal straight line, after compensating for offset error. The gain error cancellation is handled by the Gain Correction register (GAINCORR). To correct these two errors, the Digital Correction Logic Enabled bit in the Control B register (CTRLB.CORREN) must be written to one.

Offset and gain error compensation results are both calculated according to:

$$\text{Result} = (\text{Conversion value} - \text{OFFSETCORR}) \cdot \text{GAINCORR}$$

In single conversion, a latency of 13 GCLK_ADC is added to the availability of the final result. Since the correction time is always less than the propagation delay, this latency appears in free-running mode only during the first conversion. After that, a new conversion will be initialized when a conversion completes. All other conversion results are available at the defined sampling rate.

Figure 29-8. ADC Timing Correction Enabled



The ADC has the following interrupt sources:

- Result Conversion Ready: RESRDY. This is an asynchronous interrupt and can be used to wake-up the device from any sleep mode.
- Overrun: OVERRUN
- Window Monitor: WINMON. This is an asynchronous interrupt and can be used to wake-up the device from any sleep mode.
- Synchronization Ready: SYNCRDY. This is an asynchronous interrupt and can be used to wake-up the device from any sleep mode.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR) register. An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the peripheral is reset. An interrupt flag is cleared by writing a one to the corresponding bit in the INTFLAG register. Each peripheral can have one interrupt request line per interrupt source or one common interrupt request line for all the interrupt sources. This is device dependent.

Refer to "Nested Vector Interrupt Controller" on page 23 for details. If the peripheral has one common interrupt request line for all the interrupt sources, the user must read the INTFLAG register to determine which interrupt condition is present.

The peripheral can generate the following output events:

- Result Ready (RESRDY)
- Window Monitor (WINMON)

Output events must be enabled to be generated. Writing a one to an Event Output bit in the Event Control register (EVCTRL.xxEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. The events must be correctly routed in the Event System. Refer to "EVSYS – Event System" on page 402 for details.

The peripheral can take the following actions on an input event:

- ADC start conversion (START)
- ADC conversion flush (FLUSH)

Input events must be enabled for the corresponding action to be taken on any input event. Writing a one to an Event Input bit in the Event Control register (EVCTRL.xxEI) enables the corresponding action on the input event. Writing a zero to this bit disables the corresponding action on the input event. Note that if several events are connected to the peripheral, the enabled action will be taken on any of the incoming events. The events must be correctly routed in the Event System. Refer to "EVSYS – Event System" on page 402 for details.

29.6.11 Sleep Mode Operation

The Run in Standby bit in the Control A register (CTRLA.RUNSTDBY) controls the behavior of the ADC during standby sleep mode. When the bit is zero, the ADC is disabled during sleep, but maintains its current configuration. When
the bit is one, the ADC continues to operate during sleep. Note that when RUNSTDBY is zero, the analog blocks are powered off for the lowest power consumption. This necessitates a start-up time delay when the system returns from sleep.
When RUNSTDBY is one, any enabled ADC interrupt source can wake up the CPU. While the CPU is sleeping, ADC conversion can only be triggered by events.

29.6.12 Synchronization

Due to the asynchronicity between CLK_ADC_APB and GCLK_ADC, some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read

- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set immediately, and cleared when synchronization is complete. The Synchronization Ready interrupt can be used to signal when synchronization is complete.

If an operation that requires synchronization is executed while STATUS.SYNCBUSY is one, the bus will be stalled. All operations will complete successfully, but the CPU will be stalled and interrupts will be pending as long as the bus is stalled.

The following bits need synchronization when written:

- Software Reset bit in the Control A register (CTRLA.SWRST)
- Enable bit in the Control A register (CTRLA.ENABLE)

The following registers need synchronization when written:

- Control B (CTRLB)
- Software Trigger (SWTRIG)
- Window Monitor Control (WINCTRL)
- Input Control (INPUTCTRL)
- Window Upper/Lower Threshold (WINUT/WINLT)

Write-synchronization is denoted by the Write-Synchronized property in the register description.

The following registers need synchronization when read:

- Software Trigger (SWTRIG)
- Input Control (INPUTCTRL)
- Result (RESULT)

Read-synchronization is denoted by the Read-Synchronized property in the register description.

29.7 Register Summary

Table 29-5. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0						RUNSTDBY	ENABLE	SWRST
0x01	REFCTRL	7:0	REFCOMP				REFSEL[3:0]			
0x02	AVGCTRL	7:0		ADJRES[2:0]			SAMPLENUM[3:0]			
0x03	SAMPCTRL	7:0			SAMPLEN[5:0]					
0x04	CTRLB	7:0			RESSEL[1:0]		CORREN	FREERUN	LEFTADJ	DIFFMODE
0x05		15:8						PRESCALER[2:0]		
0x06	Reserved									
0x07	Reserved									
0x08	WINCTRL	7:0						WINMODE[2:0]		
0x09 ... 0x0B	Reserved									
0x0C	SWTRIG	7:0							START	FLUSH
0x0D ... 0x0F	Reserved									
0x10	INPUTCTRL	7:0				MUXPOS[4:0]				
0x11		15:8				MUXNEG[4:0]				
0x12		23:16	INPUTOFFSET[3:0]				INPUTSCAN[3:0]			
0x13		31:24					GAIN[3:0]			
0x14	EVCTRL	7:0			WINMONEO	RESRDYEO			SYNCEI	STARTEI
0x15	Reserved									
0x16	INTENCLR	7:0					SYNCRDY	WINMON	OVERRUN	RESRDY
0x17	INTENSET	7:0					SYNCRDY	WINMON	OVERRUN	RESRDY
0x18	INTFLAG	7:0					SYNCRDY	WINMON	OVERRUN	RESRDY
0x19	STATUS	7:0	SYNCBUSY							
0x1A	RESULT	7:0	RESULT[7:0]							
0x1B		15:8	RESULT[15:8]							
0x1C	WINLT	7:0	WINLT[7:0]							
0x1D		15:8	WINLT[15:8]							
0x1E	Reserved									
0x1F	Reserved									
0x20	WINUT	7:0	WINUT[7:0]							
0x21		15:8	WINUT[15:8]							
0x22	Reserved									
0x23	Reserved									
0x24	GAINCORR	7:0	GAINCORR[7:0]							
0x25		15:8					GAINCORR[11:8]			
0x26	OFFSETCORR	7:0	OFFSETCORR[7:0]							
0x27		15:8					OFFSETCORR[11:8]			

Offset	Name	Bit Pos.								
0x28	CALIB	7:0	LINEARITY_CAL[7:0]							
0x29		15:8						BIAS_CAL[2:0]		
0x2A	DBGCTRL	7:0								DBGRUN

29.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the Write-Protected property in each individual register description. Refer to "Register Access Protection" on page 612 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Write-Synchronized or the Read-Synchronized property in each individual register description. Refer to "Synchronization" on page 620 for details.

Some registers are enable-protected, meaning they can be written only when the ADC is disabled. Enable-protection is denoted by the Enable-Protected property in each individual register description.

29.8.1 Control A

Name: CTRLA

Offset: 0x00

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
						RUNSTDBY	ENABLE	SWRST
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 2 – RUNSTDBY: Run in Standby**
  This bit indicates whether the ADC will continue running in standby sleep mode or not:
  0: The ADC is halted during standby sleep mode.
  1: The ADC continues normal operation during standby sleep mode.

- **Bit 1 – ENABLE: Enable**
  0: The ADC is disabled.
  1: The ADC is enabled.
  Due to synchronization, there is a delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately and the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set. STATUS.SYNCBUSY will be cleared when the operation is complete.

- **Bit 0 – SWRST: Software Reset**
  0: There is no reset operation ongoing.
  1: The reset operation is ongoing.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all registers in the ADC, except DBGCTRL, to their initial state, and the ADC will be disabled.
  Writing a one to CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.
  Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and STATUS.SYNCBUSY will both be cleared when the reset is complete.

29.8.2 Reference Control

Name: REFCTRL

Offset: 0x01

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	REFCOMP				REFSEL[3:0]			
Access	R/W	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – REFCOMP: Reference Buffer Offset Compensation Enable**
  The accuracy of the gain stage can be increased by enabling the reference buffer offset compensation. This will decrease the input impedance and thus increase the start-up time of the reference.
  0: Reference buffer offset compensation is disabled.
  1: Reference buffer offset compensation is enabled.

- **Bits 6:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 3:0 – REFSEL[3:0]: Reference Selection**
  These bits select the reference for the ADC according to Table 29-6.

Table 29-6. Reference Selection

REFSEL[3:0]	Name	Description
0x0	INT1V	1.0V voltage reference
0x1	INTVCC0	1/1.48 VDDANA
0x2	INTVCC1	1/2 VDDANA (only for VDDANA > 2.0V)
0x3	VREFA	External reference
0x4	VREFB	External reference
0x5-0xF		Reserved

29.8.3 Average Control

Name: AVGCTRL

Offset: 0x02

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
		ADJRES[2:0]			SAMPLENUM[3:0]			
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bits 6:4 – ADJRES[2:0]: Adjusting Result / Division Coefficient**
  These bits define the division coefficient in 2^n steps.
- **Bits 3:0 – SAMPLENUM[3:0]: Number of Samples to be Collected**
  These bits define how many samples should be added together.The result will be available in the Result register (RESULT). Note: if the result width increases, CTRLB.RESSEL must be changed.

Table 29-7. Number of Samples to be Collected

SAMPLENUM[3:0]	Name	Description
0x0	1	1 sample
0x1	2	2 samples
0x2	4	4 samples
0x3	8	8 samples
0x4	16	16 samples
0x5	32	32 samples
0x6	64	64 samples
0x7	128	128 samples
0x8	256	256 samples
0x9	512	512 samples
0xA	1024	1024 samples
0xB-0xF		Reserved

29.8.4 Sampling Time Control

Name: SAMPCTRL

Offset: 0x03

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
			SAMPLEN[5:0]					
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:0 – SAMPLEN[5:0]: Sampling Time Length**
  These bits control the ADC sampling time in number of half CLK_ADC cycles, depending of the prescaler value, thus controlling the ADC input impedance. Sampling time is set according to the equation:

$$\text{Sampling time} = (SAMPLEN + 1) \cdot \left(\frac{CLK_{ADC}}{2}\right)$$

29.8.5 Control B

Name: CTRLB

Offset: 0x04

Reset: 0x0000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
						PRESCALER[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			RESSEL[1:0]		CORREN	FREERUN	LEFTADJ	DIFFMODE
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 10:8 – PRESCALER[2:0]: Prescaler Configuration**
  These bits define the ADC clock relative to the peripheral clock according to Table 29-8. These bits can only be written while the ADC is disabled.

Table 29-8. Prescaler Configuration

PRESCALER[2:0]	Name	Description
0x0	DIV4	Peripheral clock divided by 4
0x1	DIV8	Peripheral clock divided by 8
0x2	DIV16	Peripheral clock divided by 16
0x3	DIV32	Peripheral clock divided by 32
0x4	DIV64	Peripheral clock divided by 64
0x5	DIV128	Peripheral clock divided by 128
0x6	DIV256	Peripheral clock divided by 256
0x7	DIV512	Peripheral clock divided by 512

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 5:4 – RESSEL[1:0]: Conversion Result Resolution**
  These bits define whether the ADC completes the conversion at 12-, 10- or 8-bit result resolution. These bits can be written only while the ADC is disabled.

Table 29-9. Conversion Result Resolution

RESSEL[1:0]	Name	Description
0x0	12BIT	12-bit result
0x1	16BIT	For averaging mode output
0x2	10BIT	10-bit result
0x3	8BIT	8-bit result

- **Bit 3 – CORREN: Digital Correction Logic Enabled**

  0: Disable the digital result correction.

  1: Enable the digital result correction. The ADC conversion result in the RESULT register is then corrected for gain and offset based on the values in the GAINCAL and OFFSETCAL registers. Conversion time will be increased by X cycles according to the value in the Offset Correction Value bit group in the Offset Correction register.

  This bit can be changed only while the ADC is disabled.

- **Bit 2 – FREERUN: Free Running Mode**

  0: The ADC run is single conversion mode.

  1: The ADC is in free running mode and a new conversion will be initiated when a previous conversion completes.

  This bit can be changed only while the ADC is disabled.

- **Bit 1 – LEFTADJ: Left-Adjusted Result**

  0: The ADC conversion result is right-adjusted in the RESULT register.

  1: The ADC conversion result is left-adjusted in the RESULT register. The high byte of the 12-bit result will be present in the upper part of the result register. Writing this bit to zero (default) will right-adjust the value in the RESULT register.

  This bit can be changed only while the ADC is disabled.

- **Bit 0 – DIFFMODE: Differential Mode**

  0: The ADC is running in singled-ended mode.

  1: The ADC is running in differential mode. In this mode, the voltage difference between the MUXPOS and MUXNEG inputs will be converted by the ADC.

  This bit can be changed only while the ADC is disabled.

29.8.6 Window Monitor Control

Name: WINCTRL

Offset: 0x08

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
						WINMODE[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:0 – WINMODE[2:0]: Window Monitor Mode**
  These bits enable and define the window monitor mode. Table 29-10 shows the mode selections.

Table 29-10. Window Monitor Mode

WINMODE[2:0]	Name	Description
0x0	DISABLE	No window mode (default)
0x1	MODE1	Mode 1: RESULT > WINLT
0x2	MODE2	Mode 2: RESULT < WINUT
0x3	MODE3	Mode 3: WINLT < RESULT < WINUT
0x4	MODE4	Mode 4: !(WINLT < RESULT < WINUT)
0x5-0x7		Reserved

29.8.7 Software Trigger

Name: SWTRIG

Offset: 0x0C

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
							START	FLUSH
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 1 – START: ADC Start Conversion**
  0: The ADC will not start a conversion.
  1: The ADC will start a conversion. The bit is cleared by hardware when the conversion has started. Setting this bit when it is already set has no effect.
  Writing this bit to zero will have no effect.
- **Bit 0 – FLUSH: ADC Conversion Flush**
  0: No flush action.
  1: The ADC pipeline will be flushed. A flush will restart the ADC clock on the next peripheral clock edge, and all conversions in progress will be aborted and lost. This bit is cleared until the ADC has been flushed.
  After the flush, the ADC will resume where it left off; i.e., if a conversion was pending, the ADC will start a new conversion.
  Writing this bit to zero will have no effect.

29.8.8 Input Control

Name: INPUTCTRL

Offset: 0x10

Reset: 0x00000000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
					GAIN[3:0]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INPUTOFFSET[3:0]				INPUTSCAN[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				MUXNEG[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				MUXPOS[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:28 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 27:24 – GAIN[3:0]: Gain Factor Selection**
  These bits set the gain factor of the ADC gain stage according to the values shown in Table 29-11.

Table 29-11. Gain Factor Selection

GAIN[3:0]	Name	Description
0x0	1X	1x
0x1	2X	2x
0x2	4X	4x
0x3	8X	8x

GAIN[3:0]	Name	Description
0x4	16X	16x
0x5-0xe		Reserved
0xF	DIV2	1/2x

- **Bits 23:20 – INPUTOFFSET[3:0]: Positive Mux Setting Offset**
  The pin scan is enabled when INPUTSCAN != 0. Writing these bits to a value other than zero causes the first conversion triggered to be converted using a positive input equal to MUXPOS + INPUTOFFSET. Setting this register to zero causes the first conversion to use a positive input equal to MUXPOS.

  After a conversion, the INPUTOFFSET register will be incremented by one, causing the next conversion to be done with the positive input equal to MUXPOS + INPUTOFFSET. The sum of MUXPOS and INPUTOFFSET gives the input that is actually converted.

- **Bits 19:16 – INPUTSCAN[3:0]: Number of Input Channels Included in Scan**
  This register gives the number of input sources included in the pin scan. The number of input sources included is INPUTSCAN + 1. The input channels included are in the range from MUXPOS + INPUTOFFSET to MUXPOS + INPUTOFFSET + INPUTSCAN.

  The range of the scan mode must not exceed the number of input channels available on the device.

- **Bits 15:13 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 12:8 – MUXNEG[4:0]: Negative Mux Input Selection**
  These bits define the Mux selection for the negative ADC input. Table 29-12 shows the possible input selections.

Table 29-12. Negative Mux Input Selection

MUXNEG[4:0]	Name	Description
0x00	PIN0	ADC AIN0 pin
0x01	PIN1	ADC AIN1 pin
0x02	PIN2	ADC AIN2 pin
0x03	PIN3	ADC AIN3 pin
0x04	PIN4	ADC AIN4 pin
0x05	PIN5	ADC AIN5 pin
0x06	PIN6	ADC AIN6 pin
0x07	PIN7	ADC AIN7 pin
0x08 - 0x17		Reserved
0x18	GND	Internal ground
0x19	IOGND	I/O ground
0x1A - 0x1F		Reserved

- **Bits 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:0 – MUXPOS[4:0]: Positive Mux Input Selection**
  These bits define the Mux selection for the positive ADC input. Table 29-13 shows the possible input selections. If the internal bandgap voltage or temperature sensor input channel is selected, then the Sampling Time Length bit group in the Sampling Control register must be written with a corresponding value.

Table 29-13. Positive Mux Input Selection

MUXPOS[4:0]	Group configuration	Description
0x00	PIN0	ADC AIN0 pin
0x01	PIN1	ADC AIN1 pin
0x02	PIN2	ADC AIN2 pin
0x03	PIN3	ADC AIN3 pin
0x04	PIN4	ADC AIN4 pin
0x05	PIN5	ADC AIN5 pin
0x06	PIN6	ADC AIN6 pin
0x07	PIN7	ADC AIN7 pin
0x08	PIN8	ADC AIN8 pin
0x09	PIN9	ADC AIN9 pin
0x0A-0x17		Reserved
0x18	TEMP	Temperature reference
0x19	BANDGAP	Bandgap voltage
0x1A	SCALEDCOREVCC	1/4 scaled core supply
0x1B	SCALEDIOVCC	1/4 scaled I/O supply
0x1C-0x1F		Reserved

29.8.9 Event Control

Name: EVCTRL

Offset: 0x14

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
			WINMONEO	RESRDYEO			SYNCEI	STARTEI
Access	R	R	R/W	R/W	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 5 – WINMONEO: Window Monitor Event Out**
  This bit indicates whether the Window Monitor event output is enabled or not and an output event will be generated when the window monitor detects something.
  0: Window Monitor event output is disabled and an event will not be generated.
  1: Window Monitor event output is enabled and an event will be generated.

- **Bit 4 – RESRDYEO: Result Ready Event Out**
  This bit indicates whether the Result Ready event output is enabled or not and an output event will be generated when the conversion result is available.
  0: Result Ready event output is disabled and an event will not be generated.
  1: Result Ready event output is enabled and an event will be generated.

- **Bits 3:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 1 – SYNCEI: Synchronization Event In**
  0: A flush and new conversion will not be triggered on any incoming event.
  1: A flush and new conversion will be triggered on any incoming event.

- **Bit 0 – STARTEI: Start Conversion Event In**
  0: A new conversion will not be triggered on any incoming event.
  1: A new conversion will be triggered on any incoming event.

29.8.10 Interrupt Enable Clear

Name: INTENCLR

Offset: 0x16

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
					SYNCRDY	WINMON	OVERRUN	RESRDY
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – SYNCRDY: Synchronization Ready Interrupt Enable**
  0: The Synchronization Ready interrupt is disabled.
  1: The Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the Synchronization Ready interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Synchronization Ready Interrupt Enable bit and the corresponding interrupt request.

- **Bit 2 – WINMON: Window Monitor Interrupt Enable**
  0: The window monitor interrupt is disabled.
  1: The window monitor interrupt is enabled, and an interrupt request will be generated when the Window Monitor interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Window Monitor Interrupt Enable bit and the corresponding interrupt request.

- **Bit 1 – OVERRUN: Overrun Interrupt Enable**
  0: The Overrun interrupt is disabled.
  1: The Overrun interrupt is enabled, and an interrupt request will be generated when the Overrun interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Overrun Interrupt Enable bit and the corresponding interrupt request.

- **Bit 0 – RESRDY: Result Ready Interrupt Enable**
  0: The Result Ready interrupt is disabled.
  1: The Result Ready interrupt is enabled, and an interrupt request will be generated when the Result Ready interrupt flag is set.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the Result Ready Interrupt Enable bit and the corresponding interrupt request.

29.8.11 Interrupt Enable Set

Name: INTENSET

Offset: 0x17

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
					SYNCRDY	WINMON	OVERRUN	RESRDY
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – SYNCRDY: Synchronization Ready Interrupt Enable**
  0: The Synchronization Ready interrupt is disabled.
  1: The Synchronization Ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Synchronization Ready Interrupt Enable bit, which enables the Synchronization Ready interrupt.

- **Bit 2 – WINMON: Window Monitor Interrupt Enable**
  0: The Window Monitor interrupt is disabled.
  1: The Window Monitor interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Window Monitor Interrupt bit and enable the Window Monitor interrupt.

- **Bit 1 – OVERRUN: Overrun Interrupt Enable**
  0: The Overrun interrupt is disabled.
  1: The Overrun interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Overrun Interrupt bit and enable the Overrun interrupt.

- **Bit 0 – RESRDY: Result Ready Interrupt Enable**
  0: The Result Ready interrupt is disabled.
  1: The Result Ready interrupt is enabled.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will set the Result Ready Interrupt bit and enable the Result Ready interrupt.

29.8.12 Interrupt Flag Status and Clear

Name: INTFLAG

Offset: 0x18

Reset: 0x00

Access: Read-Write

Property: -

Bit	7	6	5	4	3	2	1	0
					SYNCRDY	WINMON	OVERRUN	RESRDY
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 3 – SYNCRDY: Synchronization Ready**
  This flag is cleared by writing a one to the flag.
  This flag is set on a one-to-zero transition of the Synchronization Busy bit in the Status register (STATUS.SYNC-BUSY), except when caused by an enable or software reset, and will generate an interrupt request if INTENCLR/SET.SYNCRDY is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Synchronization Ready interrupt flag.

- **Bit 2 – WINMON: Window Monitor**
  This flag is cleared by writing a one to the flag or by reading the RESULT register.
  This flag is set on the next GCLK_ADC cycle after a match with the window monitor condition, and an interrupt request will be generated if INTENCLR/SET.WINMON is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Window Monitor interrupt flag.

- **Bit 1 – OVERRUN: Overrun**
  This flag is cleared by writing a one to the flag.
  This flag is set if RESULT is written before the previous value has been read by CPU, and an interrupt request will be generated if INTENCLR/SET.OVERRUN is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Overrun interrupt flag.

- **Bit 0 – RESRDY: Result Ready**
  This flag is cleared by writing a one to the flag or by reading the RESULT register.
  This flag is set when the conversion result is available, and an interrupt will be generated if INTEN-CLR/SET.RESRDY is one.
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the Result Ready interrupt flag.

29.8.13 Status

Name: STATUS

Offset: 0x19

Reset: 0x00

Access: Read-Only

Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCBUSY							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – SYNCBUSY: Synchronization Busy**
  This bit is cleared when the synchronization of registers between the clock domains is complete.
  This bit is set when the synchronization of registers between clock domains is started.
- **Bits 6:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

29.8.14 Result

Name: RESULT

Offset: 0x1A

Reset: 0x0000

Access: Read-Only

Property: Read-Synchronized

Bit	15	14	13	12	11	10	9	8
	RESULT[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RESULT[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – RESULT[15:0]: Result Conversion Value**

  These bits will hold up to a 16-bit ADC result, depending on the configuration.

  In single-ended without averaging mode, the ADC conversion will produce a 12-bit result, which can be left- or right-shifted, depending on the setting of CTRLB.LEFTADJ.

  If the result is left-adjusted (CTRLB.LEFTADJ), the high byte of the result will be in bit position [15:8], while the remaining 4 bits of the result will be placed in bit locations [7:4]. This can be used only if an 8-bit result is required; i.e., one can read only the high byte of the entire 16-bit register.

  If the result is not left-adjusted (CTRLB.LEFTADJ) and no oversampling is used, the result will be available in bit locations [11:0], and the result is then 12 bits long.

  If oversampling is used, the result will be located in bit locations [15:0], depending on the settings of the Average Control register (AVGCTRL).

29.8.15 Window Monitor Lower Threshold

Name: WINLT

Offset: 0x1C

Reset: 0x0000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	WINLT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WINLT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – WINLT[15:0]: Window Lower Threshold**
  If the window monitor is enabled, these bits define the lower threshold value.

29.8.16 Window Monitor Upper Threshold

Name: WINUT

Offset: 0x20

Reset: 0x0000

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	WINUT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WINUT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:0 – WINUT[15:0]: Window Upper Threshold**
  If the window monitor is enabled, these bits define the upper threshold value.

29.8.17 Gain Correction

Name: GAINCORR

Offset: 0x24

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
					GAINCORR[11:8]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	GAINCORR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:0 – GAINCORR[11:0]: Gain Correction Value**
  If the CTRLB.CORREN bit is one, these bits define how the ADC conversion result is compensated for gain error before being written to the result register. The gaincorrection is a fractional value, a 1-bit integer plusan 11-bit fraction, and therefore 1/2 <= GAINCORR < 2. GAINCORR values range from 0.10000000000 to 1.11111111111.

29.8.18 Offset Correction

Name: OFFSETCORR

Offset: 0x26

Reset: 0x0000

Access: Read-Write

Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
					OFFSETCORR[11:8]			
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OFFSETCORR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 11:0 – OFFSETCORR[11:0]: Offset Correction Value**
  If the CTRLB.CORREN bit is one, these bits define how the ADC conversion result is compensated for offset error before being written to the Result register. This OFFSETCORR value is in two's complement format.

29.8.19 Calibration

Name: CALIB
Offset: 0x28
Reset: 0x0000
Access: Read-Write
Property: Write-Protected

Bit	15	14	13	12	11	10	9	8
						BIAS_CAL[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LINEARITY_CAL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 15:11 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 10:8 – BIAS_CAL[2:0]: Bias Calibration Value**
  This value from production test must be loaded from the NVM software calibration area into the CALIB register by software to achieve the specified accuracy.
  The value must be copied only, and must not be changed.
- **Bits 7:0 – LINEARITY_CAL[7:0]: Linearity Calibration Value**
  This value from production test must be loaded from the NVM software calibration area into the CALIB register by software to achieve the specified accuracy.
  The value must be copied only, and must not be changed.

29.8.20 Debug Control

Name: DBGCTRL

Offset: 0x2A

Reset: 0x00

Access: Read-Write

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:1 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bit 0 – DBGRUN: Debug Run**
  0: The ADC is halted during debug mode.
  1: The ADC continues normal operation during debug mode.
  This bit can be changed only while the ADC is disabled.
  This bit should be written only while a conversion is not ongoing.

30. Electrical Characteristics

30.1 Disclaimer

All typical values are measured at T = 25°C unless otherwise specified. All minimum and maximum values are valid across operating temperature and voltage unless otherwise specified.

30.2 Absolute Maximum Ratings

Stresses beyond those listed in Table 30-1 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.

Table 30-1. Absolute maximum ratings

Symbol	Parameter	Min.	Max.	Units
V_{DD}	Power supply voltage	0	3.8	V
I_{VDD}	Current into a V_{DD} pin	-	92[(1)]	mA
I_{GND}	Current out of a GND pin	-	130[(1)]	mA
V_{PIN}	Pin voltage with respect to GND and V_{DD}	GND-0.6V	V_{DD}+0.6V	V
$T_{storage}$	Storage temperature	-60	150	°C

Note: 1. Maximum source current is 46mA and maximum sink current is 65mA per cluster. A cluster is a group of GPIOs as shown in . Also note that each V_{DD}/GND pair is connected to 2 clusters so current consumption through the pair will be a sum of the clusters source/sink currents.

30.3 General Operating Ratings

The device must operate within the ratings listed in Table 30-2 in order for all other electrical characteristics and typical characteristics of the device to be valid.

Table 30-2. General operating conditions[(2)]

Symbol	Parameter	Min.	Typ.	Max.	Units
V_{DD}	Power supply voltage	2.4[(1)]	3.3	3.63	V
T_A	Temperature range	-40	25	85	°C
T_J	Junction temperature	-	-	100	°C

Notes: 1. With BOD33 disabled. If the BOD33 is enabled, check Table 30-16
2. **CAUTION:** In debugger cold-plugging mode, NVM erase operations are not protected by the BOD33 and BOD12. NVM erase operation at supply voltages below specified minimum can cause corruption of NVM areas that are mandatory for correct device behavior.

30.4 Supply Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified and are valid for a junction temperature up to T_J = 100°C. Refer to "Power Supply and Start-Up Considerations" on page 14.

Table 30-3. Supply voltage Characteristics

Symbol	Conditions	Min.	Max.	Units
V_{DDIO}	Full Voltage Range	1.62	3.63	V
V_{DDANA}		1.62	3.63	

Table 30-4. Supply Slew Rates

Symbol	Parameter	Fall Rate Max	Rise Rate Max.	Units
V_{DDIO}	DC supply peripheral I/Os, internal regulator and analog supply voltage	0.05	0.1	V/μs
V_{DDANA}		0.05	0.1	

30.5 Maximum Clock Frequencies

Table 30-5. Maximum GCLK Generator Output Frequencies

Symbol	Description	Undivided	Divided	Units
$f_{GCLKGEN0}$ / f_{GCLK_MAIN} $f_{GCLKGEN1}$ $f_{GCLKGEN2}$ $f_{GCLKGEN3}$ $f_{GCLKGEN4}$ $f_{GCLKGEN5}$	GCLK Generator Output Frequency	96	48	MHz

Table 30-6. Maximum Peripheral Clock Frequencies

Symbol	Description	Max.	Units
f_{CPU}	CPU clock frequency	48	MHz
f_{AHB}	AHB clock frequency	48	MHz
f_{APBA}	APBA clock frequency	48	MHz
f_{APBB}	APBB clock frequency	48	MHz
f_{APBC}	APBC clock frequency	48	MHz
$f_{GCLK\_DFLL48M\_REF}$	DFLL48M Reference clock frequency	33	kHz
f_{GCLK_DPLL}	FDPLL96M Reference clock frequency	2	MHz
$f_{GCLK\_DPLL\_32K}$	FDPLL96M 32k Reference clock frequency	32	kHz
f_{GCLK_WDT}	WDT input clock frequency	48	MHz

Table 30-6. **Maximum Peripheral Clock Frequencies (Continued)**

Symbol	Description	Max.	Units
f_{GCLK_RTC}	RTC input clock frequency	48	MHz
f_{GCLK_EIC}	EIC input clock frequency	48	MHz
$f_{GCLK\_EVSYS\_CHANNEL\_0}$	EVSYS channel 0 input clock frequency	48	MHz
$f_{GCLK\_EVSYS\_CHANNEL\_1}$	EVSYS channel 1 input clock frequency	48	MHz
$f_{GCLK\_EVSYS\_CHANNEL\_2}$	EVSYS channel 2 input clock frequency	48	MHz
$f_{GCLK\_EVSYS\_CHANNEL\_3}$	EVSYS channel 3 input clock frequency	48	MHz
$f_{GCLK\_EVSYS\_CHANNEL\_4}$	EVSYS channel 4 input clock frequency	48	MHz
$f_{GCLK\_EVSYS\_CHANNEL\_5}$	EVSYS channel 5 input clock frequency	48	MHz
$f_{GCLK\_SERCOMx\_SLOW}$	Common SERCOM slow input clock frequency	48	MHz
$f_{GCLK\_SERCOM0\_CORE}$	SERCOM0 input clock frequency	48	MHz
$f_{GCLK\_SERCOM1\_CORE}$	SERCOM1 input clock frequency	48	MHz
$f_{GCLK\_TC1, GCLK\_TC2}$	TC1,TC2 input clock frequency	48	MHz
f_{GCLK_ADC}	ADC input clock frequency	48	MHz

30.6 Power Consumption

The values in Table 30-7 are measured values of power consumption under the following conditions, except where noted:

- Operating conditions
  - V_{VDDIN} = 3.3V
  - V_{DDIN} = 1.8V, CPU is running on Flash with 3 wait state
  - Wake up time from sleep mode is measured from the edge of the wakeup signal to the execution of the first instruction fetched in flash.
- Oscillators
  - XOSC (crystal oscillator) stopped
  - XOSC32K (32kHz crystal oscillator) running with external 32kHz crystal
  - DFLL48M using XOSC32K as reference and running at 48MHz
- Clocks
  - DFLL48M used as main clock source, except otherwise specified.
  - CPU, AHB clocks undivided
  - APBA clock divided by 4
  - APBB and APBC bridges off
  - The following AHB module clocks are running: NVMCTRL, APBA bridge
    - All other AHB clocks stopped
  - The following peripheral clocks running: PM, SYSCTRL, RTC
    - All other peripheral clocks stopped
- I/Os are inactive with internal pull-up
- CPU is running on flash with 1 wait states
- NVMCTRL cache enabled
- BOD33 disabled

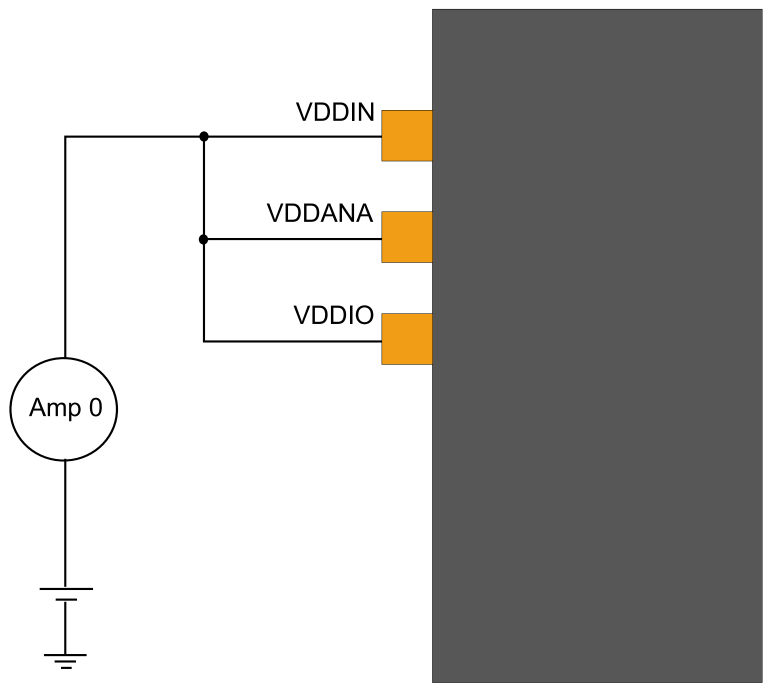
Table 30-7. **Current Consumption**

Mode	Conditions	T_A	V_{CC}	Min.	Typ.	Max.	Units
ACTIVE	CPU running a While(1) algorithm	25°C	3.3V		2.95	3.01	mA
		85°C	3.3V		2.96	3.02	
	CPU running a While(1) algorithm V_{DDIN}=1.8V, CPU is running on Flash with 3 wait states	25°C	1.8V		2.89	2.95	
		85°C	1.8V		2.94	3.02	
	CPU running a While(1) algorithm, CPU is running on Flash with 3 wait states with GCLKIN as reference	25°C	3.3V		53*freq + 282	53*freq + 330	µA (with freq in MHz)
		85°C	3.3V		54*freq + 278	54*freq + 354	
	CPU running a Fibonacci algorithm	25°C	3.3V		3.25	3.39	mA
		85°C	3.3V		3.24	3.40	
	CPU running a Fibonacci algorithm V_{DDIN}=1.8V, CPU is running on flash with 3 wait states	25°C	1.8V		3.19	3.33	
		85°C	1.8V		3.25	3.40	
	CPU running a Fibonacci algorithm, CPU is running on Flash with 3 wait states with GCLKIN as reference	25°C	3.3V		60*freq + 281	61*freq + 335	µA (with freq in MHz)
		85°C	3.3V		60*freq + 281	61*freq + 350	
	CPU running a CoreMark algorithm	25°C	3.3V		3.95	4.04	mA
		85°C	3.3V		3.98	4.07	
	CPU running a CoreMark algorithm V_{DDIN}=1.8V, CPU is running on flash with 3 wait states	25°C	1.8V		3.52	3.63	
		85°C	1.8V		3.59	3.69	
	CPU running a CoreMark algorithm, CPU is running on Flash with 3 wait states with GCLKIN as reference	25°C	3.3V		75*freq + 284	74*freq + 284	µA (with freq in MHz)
		85°C	3.3V		76*freq + 273	75*freq + 273	
IDLE0		25°C	3.3V		1.81	1.89	mA
		85°C	3.3V		1.82	1.90	
IDLE1		25°C	3.3V		1.28	1.36	
		85°C	3.3V		1.29	1.38	
IDLE2		25°C	3.3V		1.06	1.17	
		85°C	3.3V		1.06	1.18	
STANDBY	XOSC32K running RTC running at 1kHz	25°C	3.3V		3.40	10.40	µA
		85°C	3.3V		23.00	57.00	
	XOSC32K and RTC stopped	25°C	3.3V		2.20	9.20	
		85°C	3.3V		21.00	55.000	

Table 30-8. Wake-up Time

Mode	Conditions	T_A	Min.	Typ.	Max.	Units
IDLE0	OSC8M used as main clock source, cache disabled	25°C		4.0		μs
		85°C		4.0		
IDLE1	OSC8M used as main clock source, cache disabled	25°C		12.1		
		85°C		13.6		
IDLE2	OSC8M used as main clock source, cache disabled	25°C		13.0		
		85°C		14.5		
STANDBY	OSC8M used as main clock source, cache disabled	25°C		19.6		
		85°C		19.7		

Figure 30-1. Measurement Schematic



30.7 Peripheral Power Consumption

30.7.1 All peripheral

Default conditions, except where noted:

- Operating conditions
  - V_{VDDIN} = 3.3V
- Oscillators
  - XOSC (crystal oscillator) stopped
  - XOSC32K (32kHz crystal oscillator) running with external 32kHz crystal
  - OSC8M at 8MHz
- Clocks
  - OSC8M used as main clock source
  - CPU, AHB and APBn clocks undivided
- The following AHB module clocks are running: NVMCTRL, HPB2 bridge, HPB1 bridge, HPB0 bridge
  - All other AHB clocks stopped
- The following peripheral clocks running: PM, SYSCTRL
  - All other peripheral clocks stopped
- I/Os are inactive with internal pull-up
- CPU in IDLE0 mode
- Cache enabled
- BOD33 disabled

In this default conditions, the power consumption $I_{default}$ is measured.

Operating mode for each peripheral in turn:

- Configure and enable the peripheral GCLK (When relevant, see conditions)
- Unmask the peripheral clock
- Enable the peripheral (when relevant)
- Set CPU in IDLE0 mode
- Measurement I_{periph}
- Wake-up CPU via EIC (async: level detection, filtering disabled)
- Disable the peripheral (when relevant)
- Mask the peripheral clock
- Disable the peripheral GCLK (when relevant, see conditions)

Each peripheral power consumption provided in the table is the value (I_{periph} - $I_{default}$), using the same measurement method as for global power consumption measurement.

Table 30-9. Typical Peripheral Power Consumption

Peripheral	Conditions	Typ.	Units
RTC	f_{GCLK_RTC} = 32kHz, 32bit counter mode	5.20	µA
WDT	f_{GCLK_WDT}=32kHz, normal mode with EW	2.70	µA
TCx[(1)]	f_{GCLK}=8MHz, Enable + COUNTER in 8bit mode	38.9	µA
SERCOMx.I2CM[(2)]	f_{GCLK}=8MHz, Enable	62.9	µA
SERCOMx.I2CS[(2)]	f_{GCLK}=8MHz, Enable	25.1	µA
SERCOMx.SPI[(2)]	f_{GCLK}=8MHz, Enable	58.7	µA
SERCOMx.USART[(2)]	f_{GCLK}=8MHz, Enable	70.7	µA
DMAC[(3)]	RAM to RAM transfer	178.3	µA

Notes:
1. All TCs share the same power consumption values.
2. All SERCOMs share the same power consumption values.
3. The value includes the power consumption of the R/W access to the RAM

30.8 I/O Pin Characteristics

30.8.1 Normal I/O Pins

Table 30-10. Normal I/O Pins Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
R_{PULL}	Pull-up - Pull-down resistance		20	40	60	kΩ
V_{IL}	Input low-level voltage	V_{DD}=1.62V-2.7V	-	-	0.25*V_{DD}	V
		V_{DD}=2.7V-3.63V	-	-	0.3*V_{DD}	
V_{IH}	Input high-level voltage	V_{DD}=1.62V-2.7V	0.7*V_{DD}	-	-	
		V_{DD}=2.7V-3.63V	0.55*V_{DD}	-	-	
V_{OL}	Output low-level voltage	V_{DD}>1.6V, I_{OL} max	-	0.1*V_{DD}	0.2*V_{DD}	
V_{OH}	Output high-level voltage	V_{DD}>1.6V, I_{OH} max	0.8*V_{DD}	0.9*V_{DD}	-	
I_{OL}	Output low-level current	V_{DD}=1.62V-3V, PORT.PINCFG.DRVSTR=0	-	-	1	mA
		V_{DD}=3V-3.63V, PORT.PINCFG.DRVSTR=0	-	-	2.5	
		V_{DD}=1.62V-3V, PORT.PINCFG.DRVSTR=1	-	-	3	
		V_{DD}=3V-3.63V, PORT.PINCFG.DRVSTR=1	-	-	10	
I_{OH}	Output high-level current	V_{DD}=1.62V-3V, PORT.PINCFG.DRVSTR=0	-	-	0.7	
		V_{DD}=3V-3.63V, PORT.PINCFG.DRVSTR=0	-	-	2	
		V_{DD}=1.62V-3V, PORT.PINCFG.DRVSTR=1	-	-	2	
		V_{DD}=3V-3.63V, PORT.PINCFG.DRVSTR=1	-	-	7	
t_{RISE}	Rise time[1]	PORT.PINCFG.DRVSTR=0 load = 5pF, V_{DD} = 3.3V		-	15	nS
		PORT.PINCFG.DRVSTR=1 load = 20pF, Vdd = 3.3V		-	15	
t_{FALL}	Fall time[1]	PORT.PINCFG.DRVSTR=0 load = 5pF, V_{DD} = 3.3V		-	15	
		PORT.PINCFG.DRVSTR=1 load = 20pF, Vdd = 3.3V		-	15	
I_{LEAK}	Input leakage current	Pull-up resistors disabled	-1	+/-0.015	1	µA

Note: 1. These values are based on simulation. These values are not covered by test limits in production or characterization.

30.8.2 I²C Pins

Refer to "I/O Multiplexing and Considerations" on page 10 to get the list of I²C pins.

Table 30-11. I²C Pins Characteristics in I²C configuration

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
R_{PULL}	Pull-up - Pull-down resistance		20	40	60	kΩ
V_{IL}	Input low-level voltage	V_{DD}=1.62V-2.7V	-	-	0.25*V_{DD}	V
		V_{DD}=2.7V-3.63V	-	-	0.3*V_{DD}	
V_{IH}	Input high-level voltage	V_{DD}=1.62V-2.7V	0.7*V_{DD}	-	-	
		V_{DD}=2.7V-3.63V	0.55*V_{DD}	-	-	
V_{HYS}	Hysteresis of Schmitt trigger inputs		0.08*V_{DD}	-	-	
V_{OL}	Output low-level voltage	V_{DD}> 2.0V I_{OL}=3mA	-	-	0.4	
		V_{DD}≤2.0V I_{OL}=2mA	-	-	0.2*V_{DD}	
C_I	Capacitance for each I/O Pin					pF
I_{OL}	Output low-level current	V_{OL} =0.4V Standard, Fast and HS Modes	3	-	-	mA
		V_{OL} =0.4V Fast Mode+	20	-	-	
		V_{OL} =0.6V	6	-	-	
f_{SCL}	SCL clock frequency		-	-	3.4	MHz
R_P	Value of pull-up resistor	f_{SCL} ≤ 100kHz				Ω
		f_{SCL} > 100kHz				

I²C pins timing characteristics can be found in "SERCOM in I2C Mode Timing" on page 680.

Table 30-12. I²C Pins Characteristics in I/O Configuration

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
R_{PULL}	Pull-up - Pull-down resistance		20	40	60	kΩ
V_{IL}	Input low-level voltage	V_{DD}=1.62V-2.7V	-	-	0.25*V_{DD}	V
		V_{DD}=2.7V-3.63V	-	-	0.3*V_{DD}	
V_{IH}	Input high-level voltage	V_{DD}=1.62V-2.7V	0.7*V_{DD}	-	-	
		V_{DD}=2.7V-3.63V	0.55*V_{DD}	-	-	
V_{OL}	Output low-level voltage	V_{DD}>1.6V, I_{OL} max	-	0.1*V_{DD}	0.2*V_{DD}	
V_{OH}	Output high-level voltage	V_{DD}>1.6V, I_{OH} max	0.8*V_{DD}	0.9*V_{DD}	-	

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
I_{OL}	Output low-level current	V_{DD}=1.62V-3V, PORT.PINCFG.DRVSTR=0	-	-	1	mA
		V_{DD}=3V-3.63V, PORT.PINCFG.DRVSTR=0	-	-	2.5	
		V_{DD}=1.62V-3V, PORT.PINCFG.DRVSTR=1	-	-	3	
		V_{DD}=3V-3.63V, PORT.PINCFG.DRVSTR=1	-	-	10	
I_{OH}	Output high-level current	V_{DD}=1.62V-3V, PORT.PINCFG.DRVSTR=0	-	-	0.7	
		V_{DD}=3V-3.63V, PORT.PINCFG.DRVSTR=0	-	-	2	
		V_{DD}=1.62V-3V, PORT.PINCFG.DRVSTR=1	-	-	2	
		V_{DD}=3V-3.63V, PORT.PINCFG.DRVSTR=1	-	-	7	
t_{RISE}	Rise time[(1)]	PORT.PINCFG.DRVSTR=0 load = 5pF, V_{DD} = 3.3V		-	15	nS
		PORT.PINCFG.DRVSTR=1 load = 20pF, V_{DD} = 3.3V		-	15	
t_{FALL}	Fall time[(1)]	PORT.PINCFG.DRVSTR=0 load = 5pF, V_{DD} = 3.3V		-	15	
		PORT.PINCFG.DRVSTR=1 load = 20pF, V_{DD} = 3.3V		-	15	
I_{LEAK}	Input leakage current	Pull-up resistors disabled	-1	+/-0.015	1	µA

Note: 1. These values are based on simulation. These values are not covered by test limits in production or characterization.

30.8.3 XOSC Pin

XOSC pins behave as normal pins when used as normal I/Os. Refer to Table 30-10.

30.8.4 XOSC32 Pin

XOSC32 pins behave as normal pins when used as normal I/Os. Refer to Table 30-10.

30.8.5 External Reset Pin

Reset pin has the same electrical characteristics as normal I/O pins. Refer to Table 30-10.

30.9 Injection Current

Stresses beyond those listed in the table below 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.

Table 30-13. Injection Current[(1)]

Symbol	Parameter	Min.	Max.	Units
I_{inj1} [(2)]	IO pin injection current	-1	+1	mA
T_{inj2} [(3)]	IO pin injection current	-15	+15	
$I_{injtotal}$	Sum of IO pins injection current	-45	+45	

Notes:
1. Injecting current may have an effect on the accuracy of Analog blocks
2. Conditions for Vpin: Vpin < GND-0.3V or 3.6V < Vpin ≤ 4.2V
   Conditions for VDD: 3V < VDD ≤ 3.6V.
   If V_{pin} is lower than GND-0.6V, then a current limiting resistor is required. The negative DC injection current limiting resistor R is calculated as R = |(GND-0.6V . Vpin)/Iinj1|.
   If Vpin is greater than VDD+0.6V, a current limiting resistor is required. The positive DC injection current limiting resistor R is calculated as R = (Vpin-(VDD+0.6V))/Iinj1
3. Conditions for Vpin: Vpin < GND-0.6V or Vpin ≤ 3.6V.
   Conditions for VDD: VDD ≤ 3V.
   If Vpin is lower than GND-0.6V, a current limiting resistor is required. The negative DC injection current limiting resistor R is calculated as R = |(GND-0.6V . Vpin)/Iinj2|.
   If Vpin is greater than VDD+0.6V, a current limiting resistor is required. The positive DC injection current limiting resistor R is calculated as R = (Vpin-(VDD+0.6V))/Iinj2.

30.10 Analog Characteristics

30.10.1 Voltage Regulator Characteristics

Table 30-14. Decoupling requirements

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
C_{IN}	Input regulator capacitor, between V_{DDIN} and GND		-	4.7	-	µF

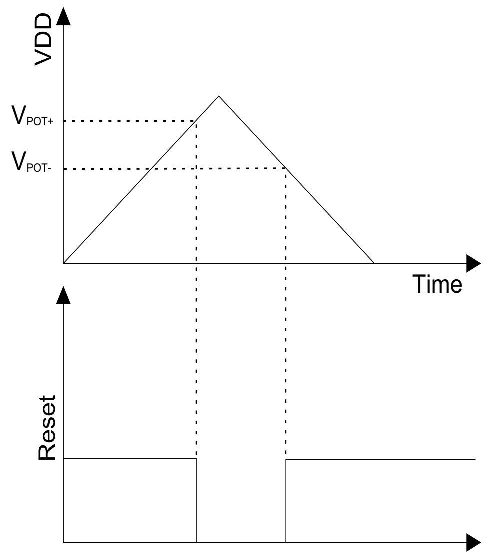
Note: Supplying any external components using VDDCORE pin is not allowed to assure the integrity of the core supply voltage.

30.10.2 Power-On Reset (POR) Characteristics

Table 30-15. POR Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
V_{POT+}	Voltage threshold on V_{DDIN} rising	V_{DD} falls at 1V/ms or slower	1.27	1.43	1.58	V
V_{POT-}	Voltage threshold on V_{DDIN} falling		0.72	1.02	1.32	

Figure 30-2. POR Operating Principle



30.10.3 Brown-Out Detectors Characteristics

30.10.3.1 BOD33

Figure 30-3. BOD33 Hysteresis OFF

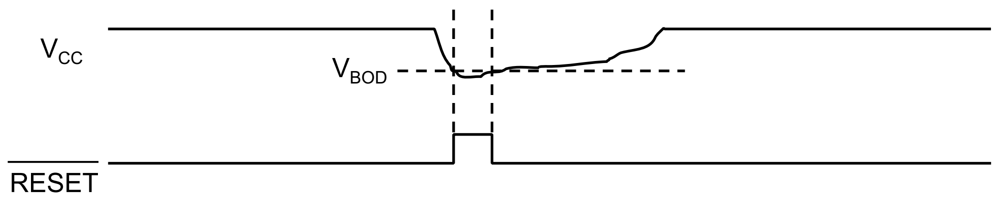


Figure 30-4. BOD33 Hysteresis ON

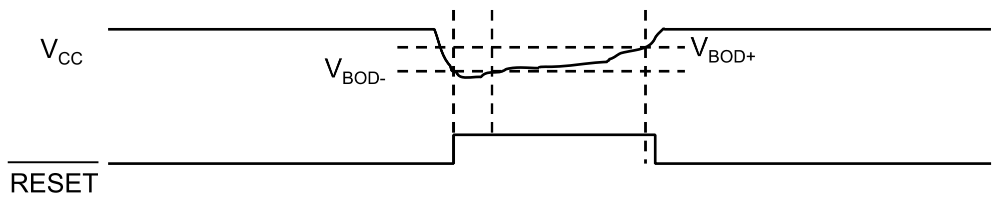


Table 30-16. BOD33 LEVEL Value

Symbol	BOD33.LEVEL	Conditions	Min.	Typ.	Max.	Units
V_{BOD+}	6	Hysteresis ON	-	1.67	1.71	V
	7		-	1.70	1.75	
	39		-	2.81	2.83	
	48		-	3.09	3.20	
V_{BOD-} or V_{BOD}	6	Hysteresis ON or Hysteresis OFF	1.61	1.64	1.65	
	7		1.64	1.67	1.69	
	39		2.72	2.76	2.79	
	48		3.00	3.07	3.10	

Note: See chapter Memories table "NVM User Row Mapping" on page 20 for the BOD33 default value settings.

Table 30-17. BOD33 Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
	Step size, between adjacent values in BOD33.LEVEL		-	34	-	mV
V_{HYST}	V_{BOD+} - V_{BOD-}	Hysteresis ON	35	-	100	
t_{DET}	Detection time	Time with V_{DDANA} < V_{TH} necessary to generate a reset signal	-	0.9[1]	-	μs
$t_{STARTUP}$	Startup time		-	2.2[1]	-	

Note: 1. These values are based on simulation. These values are not covered by test limits in production or characterization.

Table 30-18. BOD33 Mode Characteristics

Symbol	Parameter	Conditions	T_A	V_{CC}	Typ.	Max.	Units
I_{BOD33}	Current consumption	IDLE2, Continuous mode	25°C	3.3V	25	48	μA
			-40 to 85°C		-	50	
		IDLE2, Sampling mode	25°C	1.8V	0.034	0.21	
			-40 to 85°C		-	1.62	
		STDBY, Sampling mode	25°C	3.3V	0.132	0.38	
			-40 to 85°C		-	1	

30.10.4 Analog-to-Digital (ADC) Characteristics

Table 30-19. Operating Conditions

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
RES	Resolution		8	-	12	bits
f_{CLK_ADC}	ADC Clock frequency		30	-	2100	kHz
	Conversion speed		10		1000	
	Sample rate[1]	Single shot	5	-	300	ksps
		Free running	5	-	350[3]	
	Sampling time[1]		0.5	-	-	cycles
	Conversion time[1]	1x Gain	6	-	-	cycles
V_{REF}	Voltage reference range		1.0	-	V_{DDANA}-0.6	V
$V_{REFINT1V}$	Internal 1V reference [2]		-	1.0	-	V
$V_{REFINTVCC0}$	Internal ratiometric reference 0[2]		-	V_{DDANA}/1.48	-	V

Table 30-19. Operating Conditions (Continued)

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
$V_{REFINTVCC0}$ Voltage Error	Internal ratiometric reference 0[2] error	$2.0V<V_{DDANA}<3.063V$	-1	-	1	%
$V_{REFINTVCC1}$	Internal ratiometric reference 1[2]	$V_{DDANA}>2.0V$	-	$V_{DDANA}/2$	-	V
$V_{REFINTVCC1}$ Voltage Error	Internal ratiometric reference 1[2] error	$2.0V<V_{DDANA}<3.063V$	-1	-	1	%
	Conversion range[1]	Differential mode	$-V_{REF}/GAIN$	-	$+V_{REF}/GAIN$	V
		Single-ended mode	0.0	-	$+V_{REF}/GAIN$	V
C_{SAMPLE}	Sampling capacitance[2]		-	3.5	-	pF
R_{SAMPLE}	Input channel source resistance[2]		-	-	3.5	kΩ
I_{DD}	DC supply current[1]	f_{CLK_ADC} = 2.1MHz[3]	-	3.8	4.5	mA

Notes:
1. These values are based on characterization. These values are not covered by test limits in production.
2. These values are based on simulation. These values are not covered by test limits in production or characterization.
3. In this condition and for a sample rate of 350ksps, a conversion takes 6 clock cycles of the ADC clock (conditions: 1X gain, 12-bit resolution, differential mode, free-running).

Table 30-20. Differential Mode

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
ENOB	Effective Number Of Bits	With gain compensation	-	10.5	10.7	bits
TUE	Total Unadjusted Error	1x Gain	3.1	4.3	20	LSB
INL	Integral Non Linearity	1x Gain	1.0	1.3	6.3	LSB
DNL	Differential Non Linearity	1x Gain	+/-0.3	+/-0.5	+/-0.98	LSB
	Gain Error	Ext. Ref 1x	-25.0	2.5	+25.0	mV
		$V_{REF}=V_{DDANA}/1.48$	-30.0	-1.5	+30.0	mV
		Bandgap	-15.0	-5.0	+10.0	mV
	Gain Accuracy[4]	Ext. Ref. 0.5x	+/-0.04	+/-0.2	+/-1.5	%
		Ext. Ref. 2x to 16x	+/-0.1	+/-0.4	+/-2.0	%
	Offset Error	Ext. Ref. 1x	-10.0	-1.5	+10.0	mV
		$V_{REF}=V_{DDANA}/1.48$	-10.0	0.5	+10.0	mV
		Bandgap	-10.0	3.0	+10.0	mV
SFDR	Spurious Free Dynamic Range	1x Gain F_{CLK_ADC} = 2.1MHz F_{IN} = 40kHz A_{IN} = 95%FSR	62.7	70.4	75.8	dB
SINAD	Signal-to-Noise and Distortion		54.1	61.4	62.7	dB
SNR	Signal-to-Noise Ratio		54.5	63.6	65.6	dB
THD	Total Harmonic Distortion		-74	-70.2	-63	dB
	Noise RMS	T=25°C	0.6	1.0	2	mV

Notes: 1. Maximum numbers are based on characterization and not tested in production, and valid for 5% to 95% of the input voltage range.
2. Dynamic parameter numbers are based on characterization and not tested in production.
3. Respect the input common mode voltage through the following equations (where V_{CM_IN} is the Input channel common mode voltage):
   c. If $|V_{IN}| > V_{REF}/4$
      - $V_{CM\_IN} < 0.95*V_{DDANA} + V_{REF}/4 – 0.75V$
      - $V_{CM\_IN} > V_{REF}/4 -0.05*V_{DDANA} -0.1V$
   d. If $|V_{IN}| < V_{REF}/4$
      - $V_{CM\_IN} < 1.2*V_{DDANA} - 0.75V$
      - $V_{CM\_IN} > 0.2*V_{DDANA} - 0.1V$
4. The gain accuracy represents the gain error expressed in percent. Gain accuracy (%) = (Gain Error in V x 100) / (2*Vref/GAIN)

Table 30-21. Single-Ended Mode

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
ENOB	Effective Number of Bits	With gain compensation		9.5	9.8	Bits
TUE	Total Unadjusted Error	1x gain		10.5	40.0	LSB
INL	Integral Non-Linearity	1x gain	1.0	1.6	10.0	LSB
DNL	Differential Non-Linearity	1x gain	+/-0.5	+/-0.6	+/-0.98	LSB
	Gain Error	Ext. Ref. 1x	-5.0	0.7	+5.0	mV
	Gain Accuracy(3)	Ext. Ref. 0.5x	+/-0.09	+/-0.59	+/-3.5	%
		Ext. Ref. 2x to 16X	+/-0.03	+/-0.2	+/-4.0	%
	Offset Error	Ext. Ref. 1x	-5	2.0	10	mV
SFDR	Spurious Free Dynamic Range	1x Gain F_{CLK_ADC} = 2.1MHz F_{IN} = 40kHz A_{IN} = 95%FSR	54.2	65.0	67.0	dB
SINAD	Signal-to-Noise and Distortion		47.5	59.5	61	dB
SNR	Signal-to-Noise Ratio		48	60.0	64	dB
THD	Total Harmonic Distortion		-74	-68.8	-62.1	dB
	Noise RMS	T = 25°C	-	1.0	-	mV

Notes: 1. Maximum numbers are based on characterization and not tested in production, and for 5% to 95% of the input voltage range.
2. Respect the input common mode voltage through the following equations (where V_{CM_IN} is the Input channel common mode voltage) for all V_{IN}:
   - $V_{CM\_IN} < 0.7*V_{DDANA} + V_{REF}/4 – 0.75V$
   - $V_{CM\_IN} > V_{REF}/4 – 0.3*V_{DDANA} - 0.1V$
3. The gain accuracy represents the gain error expressed in percent. Gain accuracy (%) = (Gain Error in V x 100) / (V_{REF}/GAIN)

30.10.4.1 Performance with the Averaging Digital Feature

Averaging is a feature which increases the sample accuracy. ADC automatically computes an average value of multiple consecutive conversions. The numbers of samples to be averaged is specified by the Number-of-Samples-to-be-collected bit group in the Average Control register (AVGCTRL.SAMPLENUM[3:0]) and the averaged output is available in the Result register (RESULT).

Table 30-22. Averaging feature

Average Number	Conditions	SNR (dB)	SINAD (dB)	SFDR (dB)	ENOB (bits)
1	In differential mode, 1x gain, V_{DDANA}=3.0V, V_{REF}=1.0V, 350kSps T= 25°C	66.0	65.0	72.8	9.75
8		67.6	65.8	75.1	10.62
32		69.7	67.1	75.3	10.85
128		70.4	67.5	75.5	10.91

30.10.4.2 Performance with the hardware offset and gain correction

Inherent gain and offset errors affect the absolute accuracy of the ADC. The offset error cancellation is handled by the Offset Correction register (OFFSETCORR) and the gain error cancellation, by the Gain Correction register (GAINCORR). The offset and gain correction value is subtracted from the converted data before writing the Result register (RESULT).

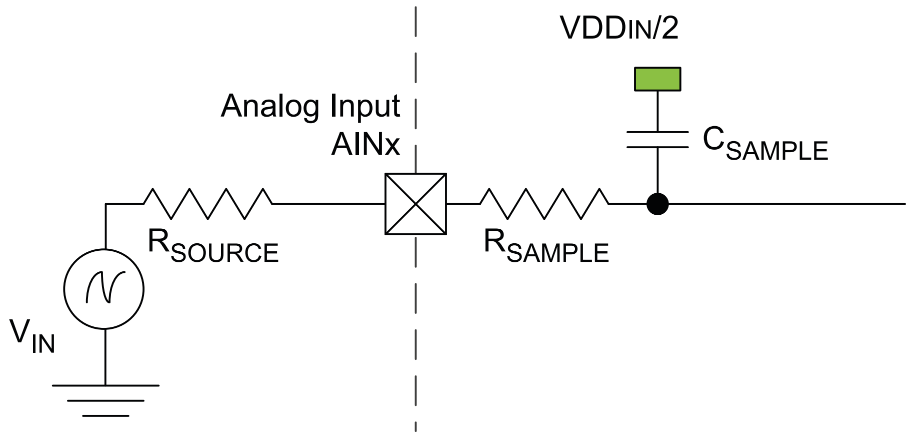
Table 30-23. Offset and Gain correction feature

Gain Factor	Conditions	Offset Error (mV)	Gain Error (mV)	Total Unadjusted Error (LSB)
0.5x	In differential mode, 1x gain, V_{DDANA}=3.0V, V_{REF}=1.0V, 350ksps T= 25°C	0.25	1.0	2.4
1x		0.20	0.10	1.5
2x		0.15	-0.15	2.7
8x		-0.05	0.05	3.2
16x		0.10	-0.05	6.1

30.10.4.3 Inputs and Sample and Hold Acquisition Times

The analog voltage source must be able to charge the sample and hold (S/H) capacitor in the ADC in order to achieve maximum accuracy. Seen externally the ADC input consists of a resistor (R_{SAMPLE}) and a capacitor (C_{SAMPLE}). In addition, the source resistance (R_{SOURCE}) must be taken into account when calculating the required sample and hold time. Figure 30-5 shows the ADC input channel equivalent circuit.

Figure 30-5. ADC Input



To achieve *n* bits of accuracy, the C_{SAMPLE} capacitor must be charged at least to a voltage of

$$V_{CSAMPLE} \geq V_{IN} \times (1 - 2^{-(n+1)})$$

The minimum sampling time $t_{SAMPLEHOLD}$ for a given R_{SOURCE} can be found using this formula:

$$t_{SAMPLEHOLD} \geq (R_{SAMPLE} + R_{SOURCE}) \times (C_{SAMPLE}) \times (n+1) \times \ln(2)$$

for a 12 bits accuracy: $t_{SAMPLEHOLD} \geq (R_{SAMPLE} + R_{SOURCE}) \times (C_{SAMPLE}) \times 9.02$

where

$$t_{SAMPLEHOLD} = \frac{1}{2 \times f_{ADC}}$$

30.10.5 Bandgap Reference Characteristics

Table 30-24. Internal 1.1V Bandgap reference characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
INT1V	Internal 1.1V Bandgap reference	Over voltage and [-40°C, +85°C]	1.08	1.1	1.12	V
		Over voltage at 25°C	1.07	1.1	1.11	

30.10.6 Temperature Sensor Characteristics

30.10.6.1 Temperature Sensor Characteristics

Table 30-25. Temperature Sensor Characteristics[(1)]

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
	Temperature sensor output voltage	T= 25°C, V_{DDANA} = 3.3V		0.688		V
	Temperature sensor slope		2.06	2.16	2.26	mV/°C
	Variation over V_{DDANA} voltage	V_{DDANA}=1.62V to 3.6V	-0.4	1.4	3.0	mV/V
	Temperature Sensor accuracy	Using the method described in Section 30.10.6.2	-10	-	10	°C

Note: 1. These values are based on characterization. These values are not covered by test limits in production.
2. See also rev C errata concerning the temperature sensor.

30.10.6.2 Software-based Refinement of the Actual Temperature

The temperature sensor behavior is linear but it depends on several parameters such as the internal voltage reference which itself depends on the temperature. To take this into account, each device contains a Temperature Log row with data measured and written during the production tests. These calibration values should be read by software to infer the most accurate temperature readings possible.

This Software Temperature Log row can be read at address 0x00806030. The Software Temperature Log row cannot be written.

This section specifies the Temperature Log row content and explains how to refine the temperature sensor output using the values in the Temperature Log row.

Temperature Log Row

All values in this row were measured in the following conditions:

- $V_{DDIN} = V_{DDIO} = V_{DDANA} = 3.3V$
- ADC Clock frequency = 1.0MHz
- ADC sample rate: 125ksps
- ADC sampling time: 57µs
- ADC mode: Free running mode, ADC averaging mode with 4 averaged samples
- Data computed on the average of 10 ADC conversions
- ADC voltage reference= 1.0V internal reference (INT1V)
- ADC input = temperature sensor

Table 30-26. Temperature Log Row Content

Bit Position	Name	Description
07:00	ROOM_TEMP_VAL_INT	Integer part of room temperature in °C
11:08	ROOM_TEMP_VAL_DEC	Decimal part of room temperature
19:12	HOT_TEMP_VAL_INT	Integer part of hot temperature in °C
23:20	HOT_TEMP_VAL_DEC	Decimal part of hot temperature

Table 30-26. Temperature Log Row Content (Continued)

Bit Position	Name	Description
31:24:00	ROOM_INT1V_VAL	2's complement of the internal 1V reference drift at room temperature (versus a 1.0 centered value)
39:32:00	HOT_INT1V_VAL	2's complement of the internal 1V reference drift at hot temperature (versus a 1.0 centered value)
51:40:00	ROOM_ADC_VAL	12bit ADC conversion at room temperature
63:52:00	HOT_ADC_VAL	12bit ADC conversion at hot temperature

The temperature sensor values are logged during test production flow for Room and Hot insertions:

- ROOM_TEMP_VAL_INT and ROOM_TEMP_VAL_DEC contains the measured temperature at room insertion (e.g. for ROOM_TEMP_VAL_INT=25 and ROOM_TEMP_VAL_DEC=2, the measured temperature at room insertion is 25.2°C).
- HOT_TEMP_VAL_INT and HOT_TEMP_VAL_DEC contains the measured temperature at hot insertion (e.g. for HOT_TEMP_VAL_INT=83 and HOT_TEMP_VAL_DEC=3, the measured temperature at room insertion is 83.3°C).

The temperature log row also contains the corresponding 12bit ADC conversions of both Room and Hot temperatures:

- ROOM_ADC_VAL contains the 12bit ADC value corresponding to (ROOM_TEMP_VAL_INT, ROOM_TEMP_VAL_DEC)
- HOT_ADC_VAL contains the 12bit ADC value corresponding to (HOT_TEMP_VAL_INT, HOT_TEMP_VAL_DEC)

The temperature log row also contains the corresponding 1V internal reference of both Room and Hot temperatures:

- ROOM_INT1V_VAL is the 2's complement of the internal 1V reference value corresponding to (ROOM_TEMP_VAL_INT, ROOM_TEMP_VAL_DEC)
- HOT_INT1V_VAL is the 2's complement of the internal 1V reference value corresponding to (HOT_TEMP_VAL_INT, HOT_TEMP_VAL_DEC)
- ROOM_INT1V_VAL and HOT_INT1V_VAL values are centered around 1V with a 0.001V step. In other words, the range of values [0,127] corresponds to [1V, 0.873V] and the range of values [-1, -127] corresponds to [1.001V, 1.127V]. INT1V == 1 - (VAL/1000) is valid for both ranges.

Using Linear Interpolation

For concise equations, we'll use the following notations:

- (ROOM_TEMP_VAL_INT, ROOM_TEMP_VAL_DEC) is denoted $temp_R$
- (HOT_TEMP_VAL_INT, HOT_TEMP_VAL_DEC) is denoted $temp_H$
- ROOM_ADC_VAL is denoted ADC_R, its conversion to Volt is denoted V_{ADCR}
- HOT_ADC_VAL is denoted ADC_H, its conversion to Volt is denoted V_{ADCH}
- ROOM_INT1V_VAL is denoted $INT1V_R$
- HOT_INT1V_VAL is denoted $INT1V_H$

Using the ($temp_R$, ADC_R) and ($temp_H$, ADC_H) points, using a linear interpolation we have the following equation:

$$\left(\frac{V_{ADC} - V_{ADCR}}{temp - temp_R}\right) = \left(\frac{V_{ADCH} - V_{ADCR}}{temp_H - temp_R}\right)$$

Given a temperature sensor ADC conversion value ADC_m, we can infer a coarse value of the temperature $temp_C$ as:

[Equation 1]

$$temp_C = temp_R + \left[\frac{\left\{\left(ADC_m \cdot \frac{1}{(2^{12}-1)}\right) - \left(ADC_R \cdot \frac{INT1V_R}{(2^{12}-1)}\right)\right\} \cdot (temp_H - temp_R)}{\left\{\left(ADC_H \cdot \frac{INT1V_H}{(2^{12}-1)}\right) - \left(ADC_R \cdot \frac{INT1V_R}{(2^{12}-1)}\right)\right\}}\right]$$

Note 1: in the previous expression, we've added the conversion of the ADC register value to be expressed in V

Note 2: this is a coarse value because we assume INT1V=1V for this ADC conversion.

Using the ($temp_R$, $INT1V_R$) and ($temp_H$, $INT1V_H$) points, using a linear interpolation we have the following equation:

$$\left(\frac{INT1V - INT1V_R}{temp - temp_R}\right) = \left(\frac{INT1V_H - INT1V_R}{temp_H - temp_R}\right)$$

Then using the coarse temperature value, we can infer a closer to reality INT1V value during the ADC conversion as:

$$INT1V_m = INT1V_R + \left(\frac{(INT1V_H - INT1V_R) \cdot (temp_C - temp_R)}{(temp_H - temp_R)}\right)$$

Back to *[Equation 1]*, we replace INT1V=1V by INT1V = $INT1V_m$, we can then deduce a finer temperature value as:

[Equation 1bis]

$$temp_f = temp_R + \left[\frac{\left\{\left(ADC_m \cdot \frac{INT1V_m}{(2^{12}-1)}\right) - \left(ADC_R \cdot \frac{INT1V_R}{(2^{12}-1)}\right)\right\} \cdot (temp_H - temp_R)}{\left\{\left(ADC_H \cdot \frac{INT1V_H}{(2^{12}-1)}\right) - \left(ADC_R \cdot \frac{INT1V_R}{(2^{12}-1)}\right)\right\}}\right]$$

30.11 NVM Characteristics

Table 30-27. Maximum Operating Frequency

V_{DD} range	NVM Wait States	Maximum Operating Frequency	Units
1.62V to 2.7V	0	14	MHz
	1	28	
	2	42	
	3	48	
2.7V to 3.63V	0	24	
	1	67	

Note that on this flash technology, a max number of 8 consecutive write is allowed per row. Once this number is reached, a row erase is mandatory.

Table 30-28. Flash Endurance and Data Retention

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
Ret_{NVM25k}	Retention after up to 25k	Average ambient 55°C	10	50	-	Years
$Ret_{NVM2.5k}$	Retention after up to 2.5k	Average ambient 55°C	20	100	-	Years
Ret_{NVM100}	Retention after up to 100	Average ambient 55°C	25	>100	-	Years
Cyc_{NVM}	Cycling Endurance[(1)]	-40°C < Ta < 85°C	25k	150k	-	Cycles

Note: 1. An endurance cycle is a write and an erase operation.

Table 30-29. EEPROM Emulation[(1)] Endurance and Data Retention

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
$Ret_{EEPROM100k}$	Retention after up to 100k	Average ambient 55°C	10	50	-	Years
$Ret_{EEPROM10k}$	Retention after up to 10k	Average ambient 55°C	20	100	-	Years
Cyc_{EEPROM}	Cycling Endurance[(2)]	-40°C < Ta < 85°C	100k	600k	-	Cycles

Notes: 1. The EEPROM emulation is a software emulation described in the App note AT03265.
2. An endurance cycle is a write and an erase operation.

Table 30-30. NVM Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
t_{FPP}	Page programming time	-	-	-	2.5	ms
t_{FRE}	Row erase time	-	-	-	6	ms
t_{FCE}	DSU chip erase time (CHIP_ERASE)	-	-	-	240	ms

Table 30-31. Flash Erase and Programming Current

Symbol	Parameter	Min.	Typ.	Max.	Units
IDD_{NVM}	Maximum current (peak) during whole programming or erase operation	-	10	-	mA

30.12 Oscillators Characteristics

30.12.1 Crystal Oscillator (XOSC) Characteristics

30.12.1.1 Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN.

Table 30-32. Digital Clock Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{CPXIN}	XIN clock frequency	Digital mode	-	-	32	MHz

30.12.1.2 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 30-6. 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:

$$C_{LEXT} = 2(C_L - C_{STRAY} - C_{SHUNT})$$

where C_{STRAY} is the capacitance of the pins and PCB, C_{SHUNT} is the shunt capacitance of the crystal.

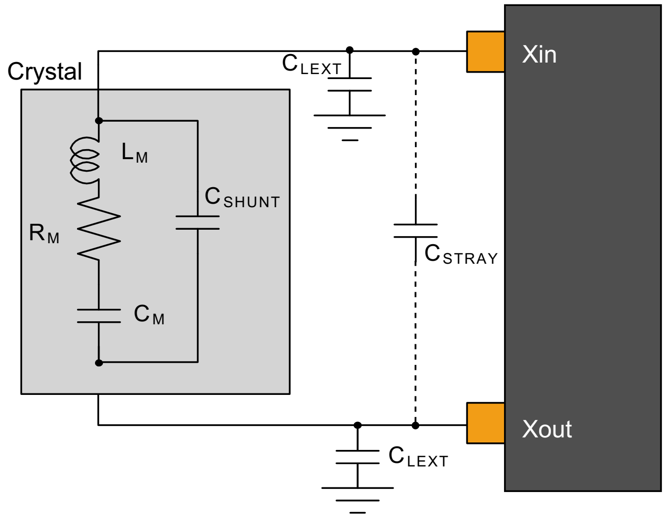
Table 30-33. Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{OUT}	Crystal oscillator frequency		0.4	-	32	MHz
ESR	Crystal Equivalent Series Resistance Safety Factor = 3	f = 0.455MHz, C_L = 100pF XOSC.GAIN = 0	-	-	5.6K	Ω
		f = 2MHz, C_L = 20pF XOSC.GAIN = 0	-	-	416	
		f = 4MHz, C_L = 20pF XOSC.GAIN = 1	-	-	243	
		f = 8MHz, C_L = 20pF XOSC.GAIN = 2	-	-	138	
		f = 16MHz, C_L = 20pF XOSC.GAIN = 3	-	-	66	
		f = 32MHz, C_L = 18pF XOSC.GAIN = 4	-	-	56	
C_{XIN}	Parasitic capacitor load		-	6.5	-	pF
C_{XOUT}	Parasitic capacitor load		-	4.3	-	pF

Table 30-33. Crystal Oscillator Characteristics (Continued)

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
I_{XOSC}	Current Consumption	f = 2MHz, C_L = 20pF, AGC off		65	85	µA
		f = 2MHz, C_L = 20pF, AGC on		52	73	
		f = 4MHz, C_L = 20pF, AGC off		117	150	
		f = 4MHz, C_L = 20pF, AGC on		74	100	
		f = 8MHz, C_L = 20pF, AGC off		226	296	
		f = 8MHz, C_L = 20pF, AGC on		128	172	
		f = 16MHz, C_L = 20pF, AGC off		502	687	
		f = 16MHz, C_L = 20pF, AGC on		307	552	
		f = 32MHz, C_L = 18pF, AGC off		1622	2200	
		f = 32MHz, C_L = 18pF, AGC on		615	1200	
$t_{STARTUP}$	Startup time	f = 2MHz, C_L = 20pF, XOSC.GAIN = 0, ESR = 600Ω		14K	48K	cycles
		f = 4MHz, C_L = 20pF, XOSC.GAIN = 1, ESR = 100Ω		6800	19.5K	
		f = 8MHz, C_L = 20pF, XOSC.GAIN = 2, ESR = 35Ω		5550	13K	
		f = 16MHz, C_L = 20pF, XOSC.GAIN = 3, ESR = 25Ω		6750	14.5K	
		f = 32MHz, C_L = 18pF, XOSC.GAIN = 4, ESR = 40Ω		5.3K	9.6K	

Figure 30-6. Oscillator Connection



30.12.2 External 32kHz Crystal Oscillator (XOSC32K) Characteristics

30.12.2.1 Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN32 pin.

Table 30-34. Digital Clock Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
$f_{CPXIN32}$	XIN32 clock frequency	Digital mode	-	32.768	-	kHz
	XIN32 clock duty cycle	Digital mode	-	50	-	%

30.12.2.2 Crystal Oscillator Characteristics

Figure 30-6 and the equation in "Crystal Oscillator Characteristics" on page 672 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 be found in the crystal datasheet.

Table 30-35. 32kHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{OUT}	Crystal oscillator frequency		-	32768		Hz
$t_{STARTUP}$	Startup time	ESR_{XTAL} = 39.9kΩ, C_L = 12.5pF	-	28K	30K	cycles
C_L	Crystal load capacitance		-	-	12.5	pF
C_{SHUNT}	Crystal shunt capacitance		-	0.1		
C_{XIN32}	Parasitic capacitor load	SOIC20/14 packages	-	6.5		
C_{XOUT32}	Parasitic capacitor load		-	4.3		
$I_{XOSC32K}$	Current consumption		-	1.22	2.19	µA
ESR	Crystal equivalent series resistance f=32.768kHz Safety Factor = 3	C_L=12.5pF	-	-	100	kΩ

30.12.3 Digital Frequency Locked Loop (DFLL48M) Characteristics

Table 30-36. DFLL48M Characteristics - Open Loop Mode[(1)]

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{OUT}	Output frequency	DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512	47	48	49	MHz
I_{DFLL}	Power consumption on V_{DDIN}	DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512	-	403	453	μA
$t_{STARTUP}$	Startup time	DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512 f_{OUT} within 90% of final value		8	9	μs

Note: 1. DFLL48M in Open loop after calibration at room temperature.

Table 30-37. DFLL48M Characteristics - Close Loop Mode[(1)]

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{OUT}	Average Output frequency	f_{REF} = 32.768kHz	47.76	48	48.24	MHz
f_{REF}	Reference frequency		0.732	32.768	33	kHz
Jitter	Cycle to Cycle jitter	f_{REF} = 32.768kHz	-	-	0.42	ns
I_{DFLL}	Power consumption on V_{DDIN}	f_{REF} = 32.768kHz	-	403	453	μA
t_{LOCK}	Lock time	f_{REF} = 32.768kHz DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512 DFLLCTRL.BPLCKC = 1 DFLLCTRL.QLDIS = 0 DFLLCTRL.CCDIS = 1 DFLLMUL.FSTEP = 10		200	500	μs

Note: 1. To insure that the device stays within the maximum allowed clock frequency, any reference clock for DFLL in close loop must be within a 2% error accuracy.

30.12.4 32.768kHz Internal oscillator (OSC32K) Characteristics

Table 30-38. 32kHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{OUT}	Output frequency	Calibrated against a 32.768kHz reference at 25°C, over [-40, +85]°C, over [1.62, 3.63]V	30.3	32.768	34.2	kHz
		Calibrated against a 32.768kHz reference at 25°C, at V_{DD}=3.3V	32.6	32.768	32.8	
		Calibrated against a 32.768kHz reference at 25°C, over [1.62, 3.63]V	32.4	32.768	33.1	

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
I_{OSC32K}	Current consumption			0.67	1.31	µA
$t_{STARTUP}$	Startup time			1	2	cycle
Duty	Duty Cycle			50		%

30.12.5 Ultra Low Power Internal 32kHz RC Oscillator (OSCULP32K) Characteristics

Table 30-39. Ultra Low Power Internal 32kHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{OUT}	Output frequency	Calibrated against a 32.768kHz reference at 25°C, over [-40, +85]C, over [1.62, 3.63]V	27.8	32.768	37.8	kHz
		Calibrated against a 32.768kHz reference at 25°C, at V_{DD}=3.3V	32.85	32.768	32.8	
		Calibrated against a 32.768kHz reference at 25°C, over [1.62, 3.63]V	31.9	32.768	33.1	
Duty	Duty Cycle			50		%

30.12.6 Multi RC Oscillator (OSC8M) Characteristics

Table 30-40. Multi RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{OUT}	Output frequency	Calibrated against a 8MHz reference at 25°C, over [-40, +85]C, over [1.62, 3.63]V	7.8	8	8.14	MHz
		Calibrated against a 8MHz reference at 25°C, at V_{DD}=3.3V	7.94	8	8.06	
		Calibrated against a 8MHz reference at 25°C, over [1.62, 3.63]V	7.92	8	8.08	
TempCo	Freq vs. temperature drift		-2.3		1.6	%
SupplyCo	Freq vs supply drift		-1		1	%
I_{OSC8M}	Current consumption	IDLE2 on OSC32K versus IDLE2 on calibrated OSC8M enabled at 8MHz (FRANGE=1, PRESC=0)		64	96	µA
$t_{STARTUP}$	Startup time			2.4	3.3	µs
Duty	Duty cycle			50		%

30.12.7 Fractional Digital Phase Locked Loop (FDPLL96M) Characteristics

Table 30-41. FDPLL96M Characteristics[(1)]

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
f_{IN}	Input frequency		32	-	2000	KHz
f_{OUT}	Output frequency		48	-	96	MHz
$I_{FDPLL96M}$	Current consumption	f_{IN}= 32 kHz, f_{OUT}= 48 MHz		400	700	µA
		f_{IN}= 32 kHz, f_{OUT}= 96 MHz		650	900	
Jp	Period jitter	f_{IN}= 32 kHz, f_{OUT}= 48 MHz		1.5	2	%
		f_{IN}= 32 kHz, f_{OUT}= 96 MHz		3.0	10	
		f_{IN}= 2 MHz, f_{OUT}= 48 MHz		1.3	2	
		f_{IN}= 2 MHz, f_{OUT}= 96 MHz		3.0	7	
t_{LOCK}	Lock Time	After startup, time to get lock signal. f_{IN}= 32 kHz, f_{OUT}= 96 MHz		1.3	2	ms
		f_{IN}= 2 MHz, f_{OUT}= 96 MHz		25	50	µs
Duty	Duty cycle		40	50	60	%

Note: 1. All values have been characterized with FILTSEL[1/0] as default value.

30.13 Timing Characteristics

30.13.1 External Reset

Table 30-42. External reset characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{EXT}	Minimum reset pulse width		10	-	-	ns

30.13.2 SERCOM in SPI Mode Timing

Figure 30-7. SPI timing requirements in master mode

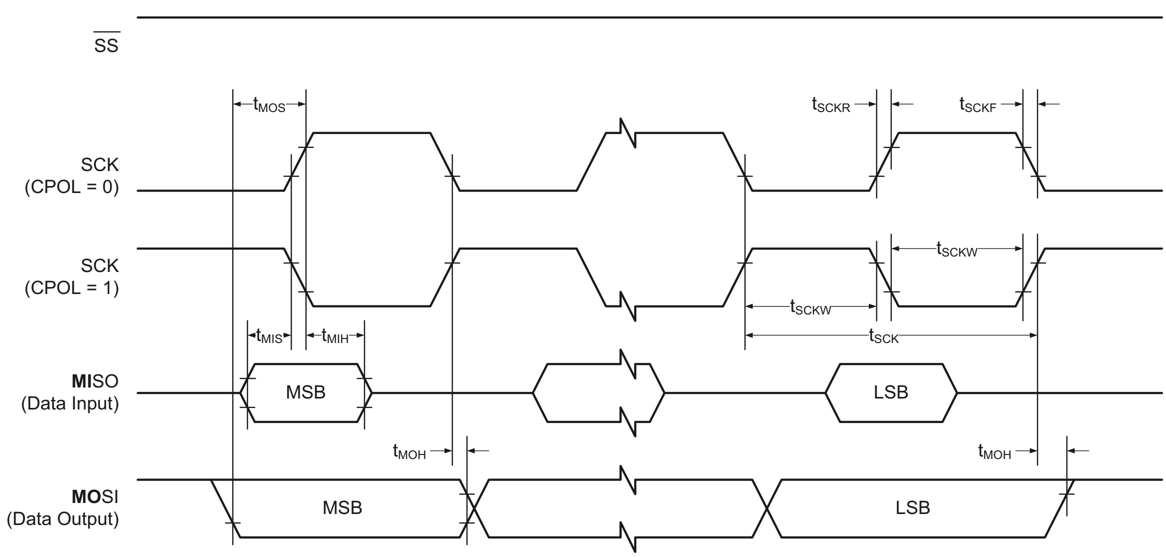


Figure 30-8. SPI timing requirements in slave mode

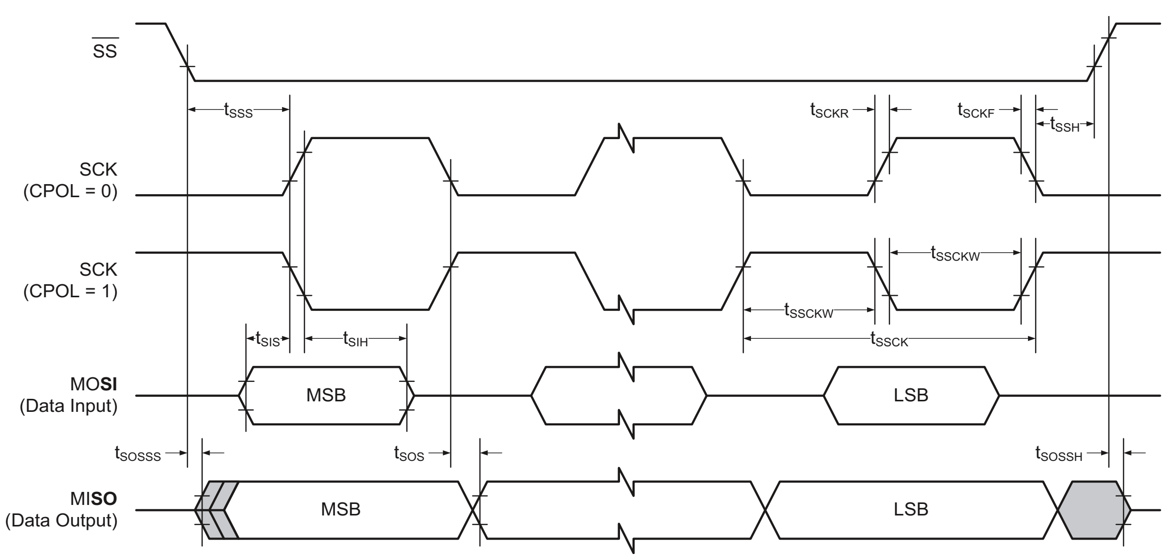


Table 30-43. SPI timing characteristics and requirements[1]

Symbol	Parameter	Conditions		Min.	Typ.	Max.	Units
t_{SCK}	SCK period	Master			84		ns
t_{SCKW}	SCK high/low width	Master		-	$0.5*t_{SCK}$	-	
t_{SCKR}	SCK rise time[2]	Master		-	-	-	
t_{SCKF}	SCK fall time[2]	Master		-	-	-	
t_{MIS}	MISO setup to SCK	Master		-	21	-	
t_{MIH}	MISO hold after SCK	Master		-	13	-	
t_{MOS}	MOSI setup SCK	Master		-	$t_{SCK}/2 - 3$	-	
t_{MOH}	MOSI hold after SCK	Master		-	3	-	
t_{SSCK}	Slave SCK Period	Slave		$1*t_{CLK\_APB}$	-	-	
t_{SSCKW}	SCK high/low width	Slave		$0.5*t_{SSCK}$	-	-	
t_{SSCKR}	SCK rise time[2]	Slave		-	-	-	
t_{SSCKF}	SCK fall time[2]	Slave		-	-	-	
t_{SIS}	MOSI setup to SCK	Slave		$t_{SSCK}/2 - 9$	-	-	
t_{SIH}	MOSI hold after SCK	Slave		$t_{SSCK}/2 - 3$	-	-	
t_{SSS}	$\overline{SS}$ setup to SCK	Slave	PRELOADEN=1	$2*t_{CLK\_APB} + t_{SOS}$	-	-	
			PRELOADEN=0	$t_{SOS}+7$	-	-	
t_{SSH}	SS hold after SCK	Slave		$t_{SIH} - 4$	-	-	
t_{SOS}	MISO setup SCK	Slave		-	$t_{SSCK}/2 - 18$	-	
t_{SOH}	MISO hold after SCK	Slave		-	18	-	
t_{SOSS}	MISO setup after $\overline{SS}$ low	Slave		-	18	-	
t_{SOSH}	MISO hold after $\overline{SS}$ high	Slave		-	10	-	

Notes: 1. These values are based on simulation. These values are not covered by test limits in production.
2. See "I/O Pin Characteristics" on page 656

30.13.3 SERCOM in I²C Mode Timing

Table 30-44 describes the requirements for devices connected to the I²C Interface Bus. Timing symbols refer to Figure 30-9.

Figure 30-9. I²C Interface Bus Timing

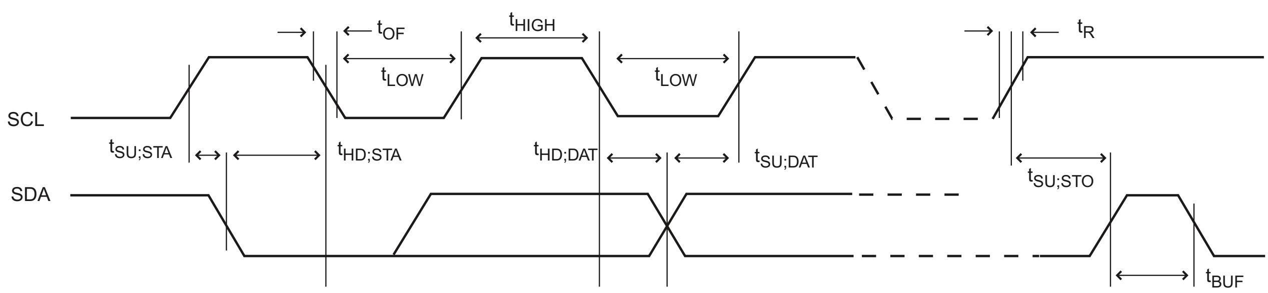


Table 30-44. I²C Interface Timing[1]

Symbol	Parameter	Conditions		Min.	Typ.	Max.	Units
t_R	Rise time for both SDA and SCL[3]	Standard / Fast mode	C_b[2] = 400pF	-	230	350	ns
		Fast mode +	C_b[2] = 550pF	-	60	100	
		High speed mode	C_b[2] = 100pF	-	30	60	
t_{OF}	Output fall time from V_{IHmin} to V_{ILmax}[3]	Standard / Fast mode	10pF < C_b[2] < 400pF	-	25	50	
		Fast mode +	10pF < C_b[2] < 550pF	-	20	30	
		High speed mode	10pF < C_b[2] < 100pF	-	10	20	
$t_{HD;STA}$	Hold time (repeated) START condition		f_{SCL} > 100kHz, Master	t_{LOW}-9	-	-	
t_{LOW}	Low period of SCL Clock		f_{SCL} > 100kHz	113	-	-	
t_{BUF}	Bus free time between a STOP and a START condition		f_{SCL} > 100kHz	t_{LOW}	-	-	
$t_{SU;STA}$	Setup time for a repeated START condition		f_{SCL} > 100kHz, Master	t_{LOW}+7	-	-	
$t_{HD;DAT}$	Data hold time		f_{SCL} > 100kHz, Master	9	-	12	
$t_{SU;DAT}$	Data setup time		f_{SCL} > 100kHz, Master	104	-	-	
$t_{SU;STO}$	Setup time for STOP condition		f_{SCL} > 100kHz, Master	t_{LOW}+9	-	-	
$t_{SU;DAT;Rx}$	Data setup time (receive mode)		f_{SCL} > 100kHz, Slave	51	-	56	
$t_{HD;DAT;Tx}$	Data hold time (send mode)		f_{SCL} > 100kHz, Slave	71	90	138	

Notes:
1. These values are based on simulation. These values are not covered by test limits in production.
2. Cb = Capacitive load on each bus line. Otherwise noted, value of C_b set to 20pF.
3. These values are based on characterization. These values are not covered by test limits in production.

30.13.4 SWD Timing

Figure 30-10.SWD Interface Signals

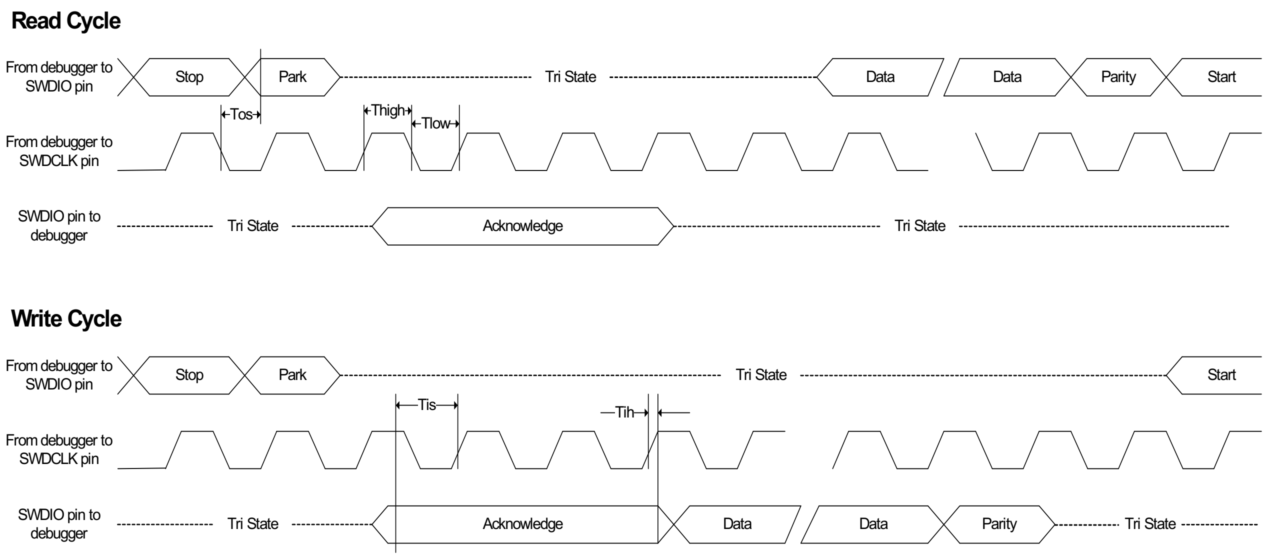


Table 30-45. SWD Interface Timings[1]

Symbol	Parameter	Conditions	Min.	Max.	Units
T_{HIGH}	SWDCLK High period	V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40pF	10	500000	ns
T_{LOW}	SWDCLK Low period		10	500000	
T_{OS}	SWDIO output skew to falling edge SWDCLK		-5	5	
T_{IS}	Input Setup time required between SWDIO		4	-	
T_{IH}	Input Hold time required between SWDIO and rising edge SWDCLK		1	-	

Note: 1. These values are based on simulation. These values are not covered by test limits in production or characterization.

31. Packaging Information

31.1 Thermal Considerations

31.1.1 Thermal Resistance Data

Table 6-1 on page 13 summarizes the thermal resistance data depending on the package.

Table 31-1. Thermal Resistance Data

Package Type	θ_{JA}	θ_{JC}	Units
24-pin QFN	61.7	25.4	°C/W
14-pin SOIC	58.5	26.3	°C/W

31.1.2 Junction Temperature

The average chip-junction temperature, T_J, in °C can be obtained from the following:

Equation 1

$$T_J = T_A + (P_D \times \theta_{JA})$$

Equation 2

$$T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$$

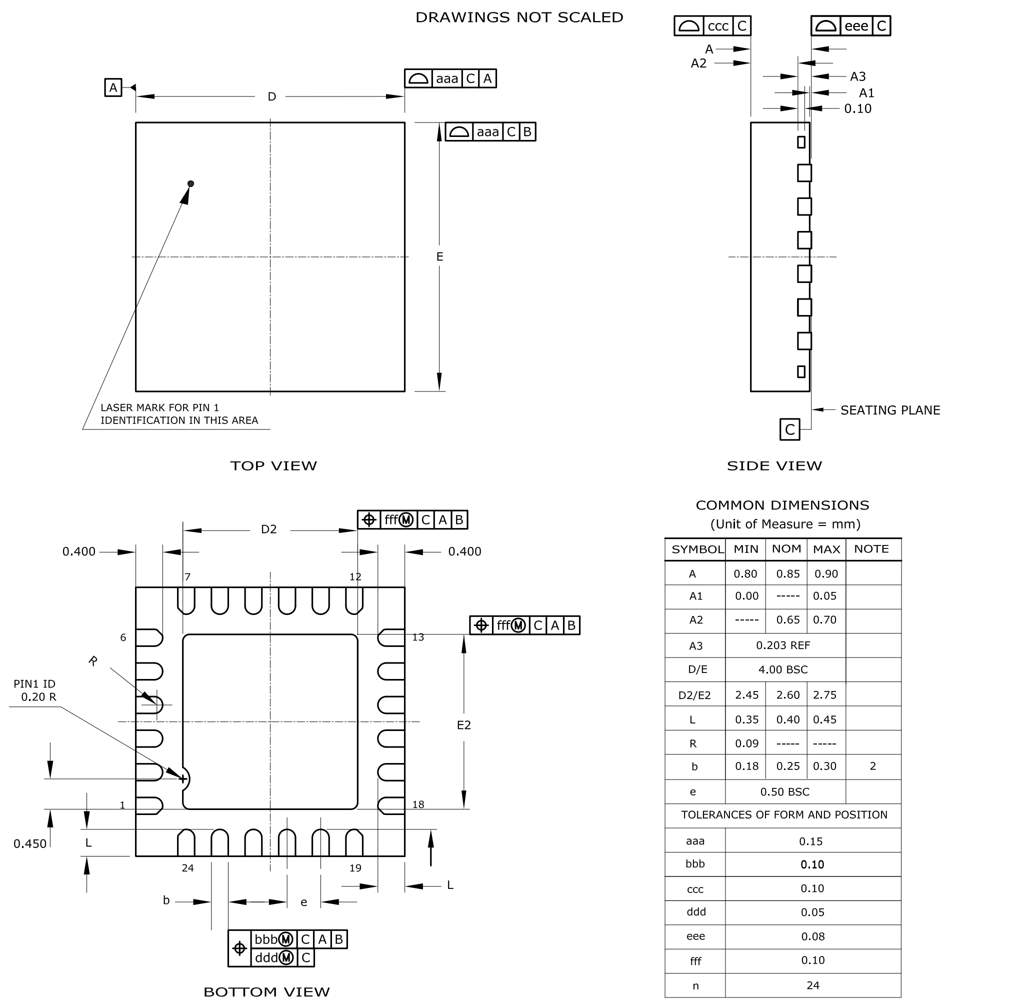
where:

- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 6-1 on page 13.
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 6-1 on page 13.
- $\theta_{HEATSINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W).
- T_A = ambient temperature (°C).

From the *Equation 1*, 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.

31.2 Package Drawings

31.2.1 24-pin QFN



SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.85	0.90	
A1	0.00	-----	0.05	
A2	-----	0.65	0.70	
A3	0.203 REF			
D/E	4.00 BSC			
D2/E2	2.45	2.60	2.75	
L	0.35	0.40	0.45	
R	0.09	-----	-----	
b	0.18	0.25	0.30	2
e	0.50 BSC			
TOLERANCES OF FORM AND POSITION				
aaa	0.15			
bbb	0.10			
ccc	0.10			
ddd	0.05			
eee	0.08			
fff	0.10			
n	24			

Notes :
1. This drawing is for general information only. Refer to JEDEC Drawing MO-220, Variation VGGD-6 for proper dimensions, tolerances, datums, etc. (excepted D2/E2 Min et Nom).
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.

02/18/2014

Table 31-2. Device and Package Maximum Weight

44	mg

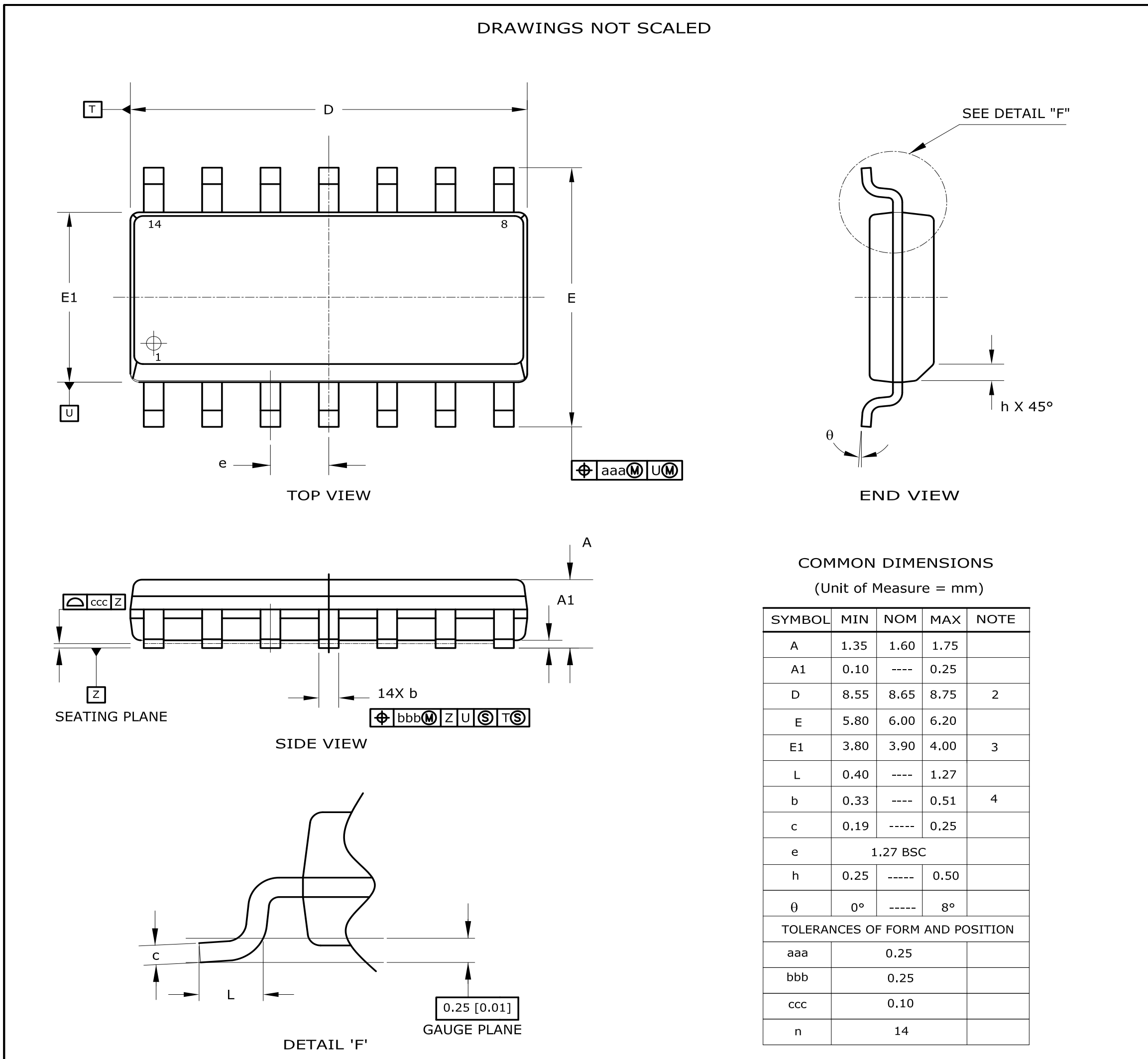
Table 31-3. Package Characteristics

Moisture Sensitivity Level	MSL3

Table 31-4. Package Reference

JEDEC Drawing Reference	MO-220
JESD97 Classification	E3

31.2.2 14-pin SOIC



COMMON DIMENSIONS

(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	1.35	1.60	1.75	
A1	0.10	----	0.25	
D	8.55	8.65	8.75	2
E	5.80	6.00	6.20	
E1	3.80	3.90	4.00	3
L	0.40	----	1.27	
b	0.33	----	0.51	4
c	0.19	-----	0.25	
e	1.27 BSC			
h	0.25	-----	0.50	
θ	0°	-----	8°	
TOLERANCES OF FORM AND POSITION				
aaa	0.25			
bbb	0.25			
ccc	0.10			
n	14			

Notes :
1. This drawing is for general information only. Refer to JEDEC Drawing MS-012, Variation AB.
2. Dimensions D does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusions and gate burrs shal not exceed 0.15mm (0.006 in) per side.
3. Dimensions E1 does not include inter-lead Flash or protrusions. Inter-lead Flash and protrusions shall not exceed 0.25mm (0.010 in) per side.
4. Dimensions b does not include Dambar protrusion. Allowable Dambar protrusion shall be 0.10mm total in excess of the b dimension at maximum material condition. Dambar cannot be located on the lower radius of the foot.

01/31/2014

Table 31-6. Device and Package Maximum Weight

230	mg

Table 31-7. Package Characteristics

Moisture Sensitivity Level	MSL3

Table 31-8. Package Reference

JEDEC Drawing Reference	MS-012
JESD97 Classification	E3

31.3 Soldering Profile

The following table gives the recommended soldering profile from J-STD-20.

Profile Feature	Green Package
Average Ramp-up Rate (217°C to peak)	3°C/s max
Preheat Temperature 175°C +/-25°C	150-200°C
Time Maintained Above 217°C	60-150s
Time within 5°C of Actual Peak Temperature	30s
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.

32. Schematic Checklist

32.1 Introduction

A good hardware design comes from a proper schematic. This chapter describes a common checklist which should be used when starting and reviewing the schematics for the design of the device. This chapter illustrates a recommended power supply connection, how to connect external analog references, programmer, debugger, oscillator,crystal.

32.1.1 Operation in Noisy Environment

If the microcontroller is operating in an environment with much electromagnetic noise it must be protected from this noise to ensure reliable operation. In addition to following best practice EMC design guidelines, the recommendations listed in the schematic checklist sections must be followed. In particular placing decoupling capacitors very close to the power pins, a RC-filter on the $\overline{\text{RESET}}$ pin, and a pull-up resistor on the SWCLK pin is critical for reliable operations. It is also relevant to eliminate or attenuate noise in order to avoid that it reaches supply pins, I/O pins and crystals.

32.2 Power Supply

The device supports a single power supply from 2.4 to 3.63V.

32.2.1 Power Supply Connections

Power Supply Schematic

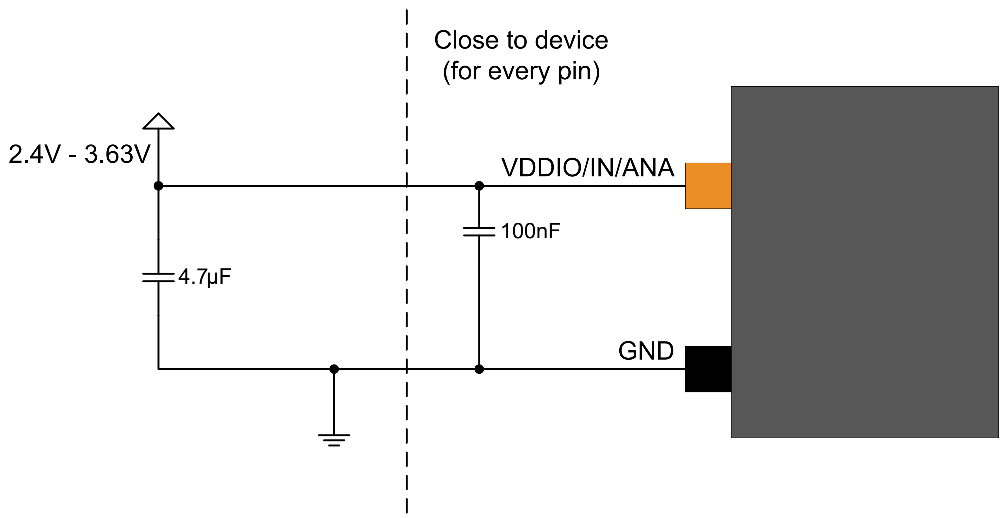


Table 32-1. Power Supply Connections, V_{DDCORE} From Internal Regulator

Signal Name	Recommended Pin Connection	Description
$V_{DDIO/IN/ANA}$	2.4V to 3.6V Decoupling/filtering capacitors 100nF[(1)(2)] and 4.7µF[(1)]	Supply voltage
GND		Ground

Notes: 1. These values are only given as typical examples.

2. Decoupling capacitor should be placed close to the device for each supply pin pair in the signal group, low ESR caps should be used for better decoupling.

32.3 External Analog Reference Connections

The following schematic checklist is only necessary if the application is using one or more of the external analog references. If the internal references are used instead, the following circuits in Figure 32-1 and Figure 32-2 are not necessary.

Figure 32-1. External Analog Reference Schematic With Two References

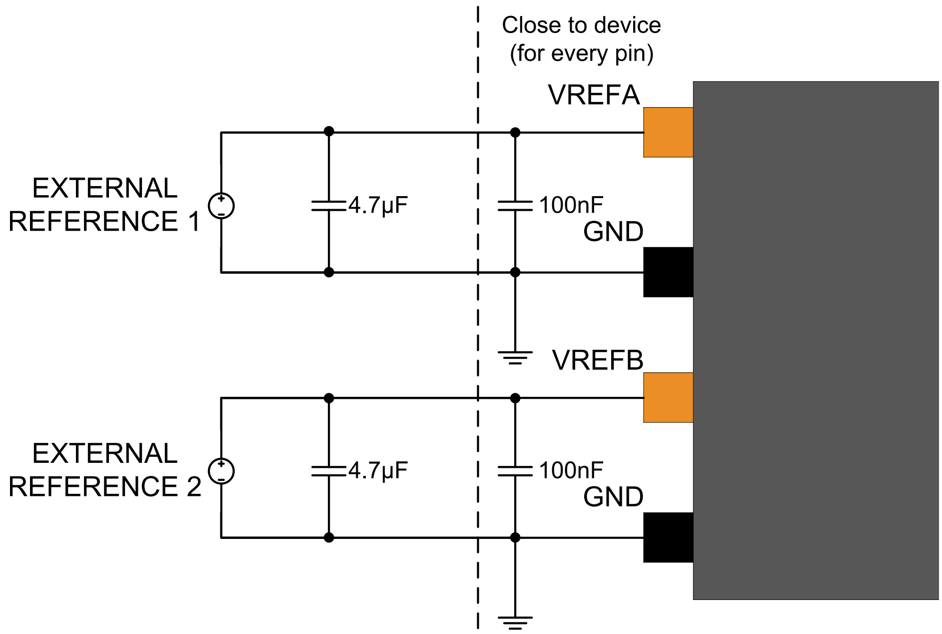


Figure 32-2. External Analog Reference Schematic With One Reference

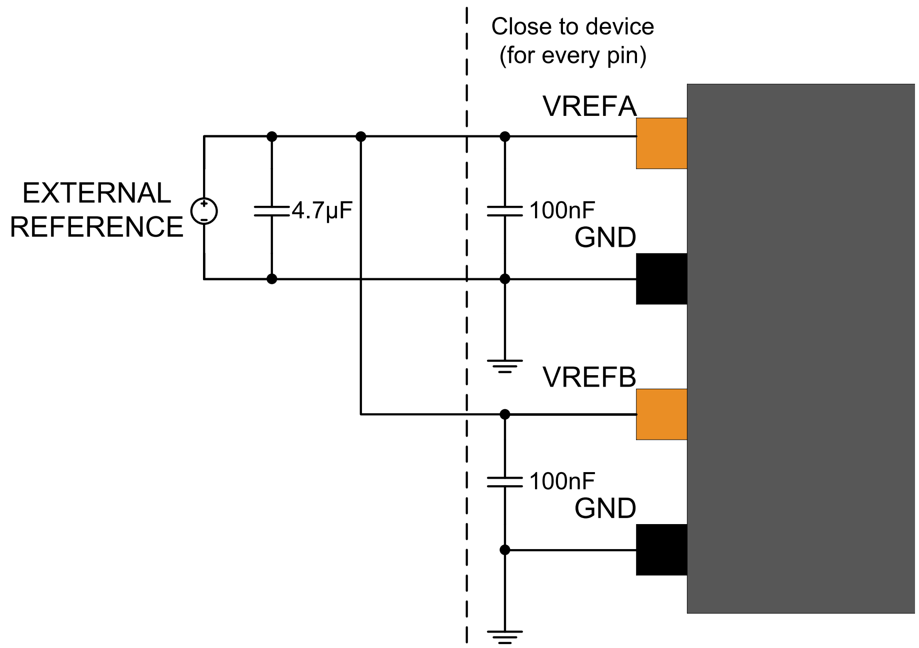


Table 32-2. External Analog Reference Connections

Signal Name	Recommended Pin Connection	Description
VREFx	1.0V to V_{DDANA} - 0.6V for ADC Decoupling/filtering capacitors 100nF[(1)(2)] and 4.7μF[(1)]	External reference from VREFx pin on the analog port
GND		Ground

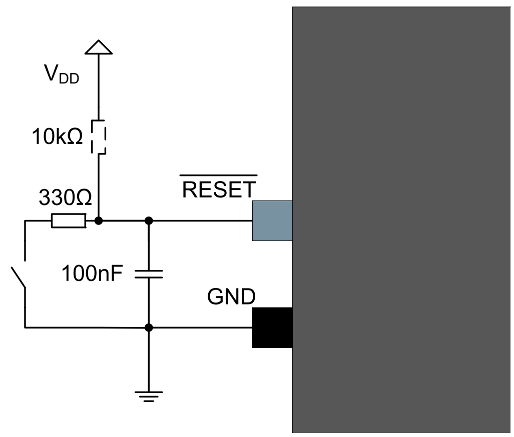
Notes: 1. These values are given as a typical example.

2. Decoupling capacitor should be placed close to the device for each supply pin pair in the signal group.

32.4 External Reset Circuit

The external reset circuit is connected to the $\overline{\text{RESET}}$ pin when the external reset function is used. If the external reset function has been disabled, the circuit is not necessary. The reset switch can also be removed, if the manual reset is not necessary. After power up, the $\overline{\text{RESET}}$ pin has $\overline{\text{RESET}}$ functionality enabled by default. To use this pin as GPIO, user has to disable the $\overline{\text{RESET}}$ functionality using External Reset Controller (EXTCTRL) register. The $\overline{\text{RESET}}$ pin itself has an internal pull-up resistor, hence it is optional to also add an external pull-up resistor.

Figure 32-3. External Reset Circuit Example Schematic



A pull-up resistor makes sure that the reset does not go low unintended causing a device reset. An additional resistor has been added in series with the switch to safely discharge the filtering capacitor, i.e. preventing a current surge when shorting the filtering capacitor which again causes a noise spike that can have a negative effect on the system.

Table 32-3. Reset Circuit Connections

Signal Name	Recommended Pin Connection	Description
$\overline{\text{RESET}}$	Reset low level threshold voltage V_{DDIO} = 1.6V - 2.0V: Below 0.33 * V_{DDIO} V_{DDIO} = 2.7V - 3.6V: Below 0.36 * V_{DDIO} Decoupling/filter capacitor 100nF[1] Pull-up resistor 10kΩ[1][2] Resistor in series with the switch 330Ω[1]	Reset pin

Notes: 1. These values are given as a typical example.

2. The device features an internal pull-up resistor on the $\overline{\text{RESET}}$ pin, hence an external pull-up is optional.

32.5 Unused or Unconnected Pins

For unused pins the default state of the pins for the will give the lowest current leakage. There is thus no need to do any configuration of the unused pins in order to lower the power consumption.

32.6 Clocks and Crystal Oscillators

The device can be run from internal or external clock sources, or a mix of internal and external sources. An example of usage will be to use the internal 8MHz oscillator as source for the system clock, and an external 32.768kHz watch crystal as clock source for the Real-Time counter (RTC).

32.6.1 External Clock Source

Figure 32-4. External Clock Source Example Schematic

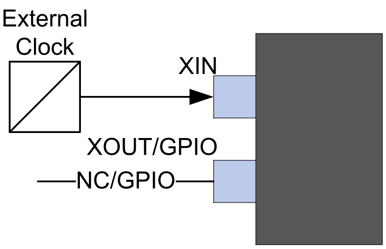
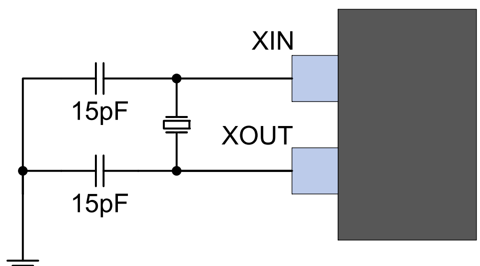


Table 32-4. External Clock Source Connections

Signal Name	Recommended Pin Connection	Description
XIN	XIN is used as input for an external clock signal	Input for inverting oscillator pin
XOUT/GPIO	Can be left unconnected or used as normal GPIO	

32.6.2 Crystal Oscillator

Figure 32-5. Crystal Oscillator Example Schematic



The crystal should be located as close to the device as possible. Long signal lines may cause too high load to operate the crystal, and cause crosstalk to other parts of the system.

Table 32-5. Crystal Oscillator Checklist

Signal Name	Recommended Pin Connection	Description
XIN	Load capacitor 15pF[(1)(2)]	External crystal between 0.4 to 30MHz
XOUT	Load capacitor 15pF[(1)(2)]	

Notes: 1. These values are given only as typical example.

2. Decoupling capacitor should be placed close to the device for each supply pin pair in the signal group.

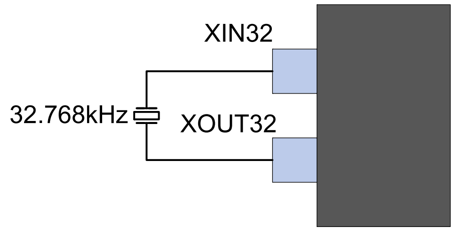
32.6.3 External Real Time Oscillator

The low frequency crystal oscillator is optimized for use with a 32.768kHz watch crystal. When selecting crystals, load capacitance and crystal's Equivalent Series Resistance (ESR) must be taken into consideration. Both values are specified by the crystal vendor.

The device's oscillator is optimized for very low power consumption, hence close attention should be made when selecting crystals, see "Crystal Oscillator Characteristics" on page 672 for maximum ESR recommendations on 12.5pF crystals.

The Low-frequency Crystal Oscillator provides an internal load capacitance of typical values available in Table 30-35. This internal load capacitance and PCB capacitance can allow to use a Crystal inferior to 12.5pF load capacitance without external capacitors as shown in Figure 32-6.

Figure 32-6. External Real Time Oscillator without Load Capacitor



However, to improve Crystal accuracy and Safety Factor, it can be recommended by crystal datasheet to add external capacitors as shown in the figure below.

To find suitable load capacitance for a 32.768kHz crystal, consult the crystal datasheet.

Figure 32-7. External Real Time Oscillator with Load Capacitor

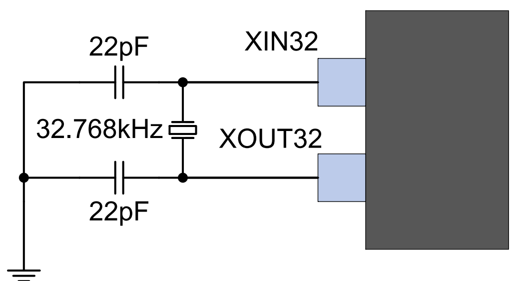


Table 32-6. External Real Time Oscillator Checklist

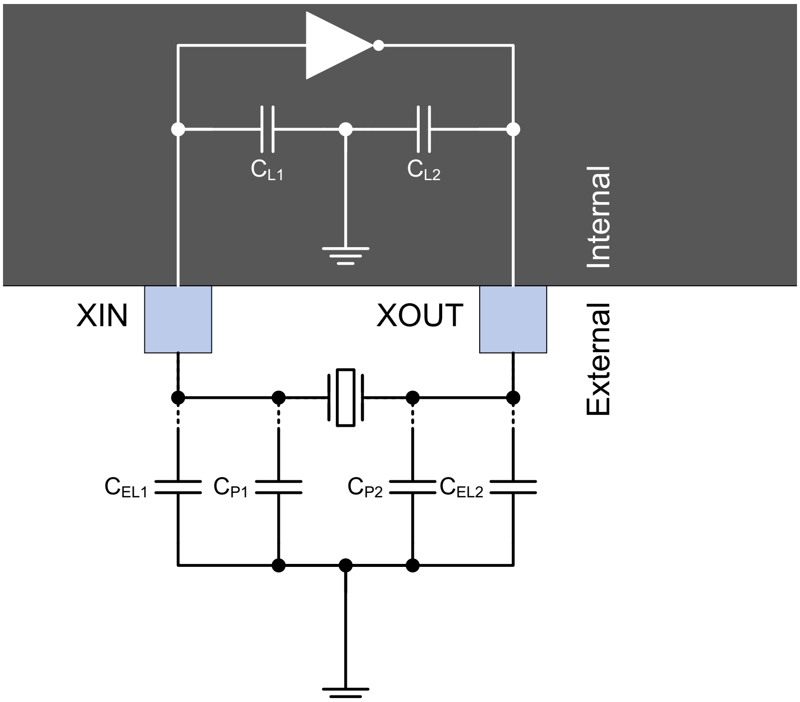
Signal Name	Recommended Pin Connection	Description
XIN32	Load capacitor 22pF[1][2]	Timer oscillator input
XOUT32	Load capacitor 22pF[1][2]	Timer oscillator output

Notes: 1. These values are given only as typical examples.
2. Decoupling capacitor should be placed close to the device for each supply pin pair in the signal group.

32.6.4 Calculating the Correct Crystal Decoupling Capacitor

In order to calculate correct load capacitor for a given crystal one can use the model shown in the following figure which includes internal capacitors C_{Ln}, external parasitic capacitance C_{ELn} and external load capacitance C_{Pn}.

Figure 32-8. Crystal Circuit With Internal, External and Parasitic Capacitance



Using this model the total capacitive load for the crystal can be calculated as shown in the equation below:

$$\sum C_{tot} = \frac{(C_{L1} + C_{P1} + C_{EL1})(C_{L2} + C_{P2} + C_{EL2})}{C_{L1} + C_{P1} + C_{EL1} + C_{L2} + C_{P2} + C_{EL2}}$$

where C_{tot} is the total load capacitance seen by the crystal, this value should be equal to the load capacitance value found in the crystal manufacturer datasheet.

The parasitic capacitance C_{ELn} can in most applications be disregarded as these are usually very small. If accounted for the value is dependent on the PCB material and PCB layout.

For some crystal the internal capacitive load provided by the device itself can be enough. To calculate the total load capacitance in this case. C_{ELn} and C_{Pn} are both zero, $C_{L1} = C_{L2} = C_L$, and the equation reduces to the following:

$$\sum C_{tot} = \frac{C_L}{2}$$

See “Electrical Characteristics” on page 648 for the device equivalent internal pin capacitance.

32.7 Programming and Debug Ports

For programming and/or debugging, the device should be connected using the Serial Wire Debug, SWD, interface. Currently the SWD interface is supported by several Atmel and third party programmers and debuggers, like the SAM-ICE, JTAGICE3 or the device’s Xplained Pro (device’s evaluation kit) Embedded Debugger.

Refer to the SAM-ICE, JTAGICE3 or the device's Xplained Pro user guides for details on debugging and programming connections and options. For connecting to any other programming or debugging tool please refer to that specific programmer or debugger's user guide.

The device's Xplained Pro evaluation board for the device supports programming and debugging through the onboard embedded debugger so no external programmer or debugger is needed.

Note that a pull-up resistor on the SWCLK pin is critical for reliable operations. Refer to "Operation in Noisy Environment" on page 687.

Figure 32-9. SWCLK Circuit Connections

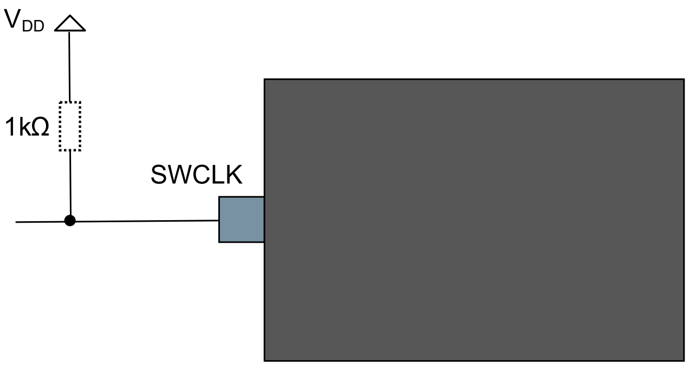


Table 32-7. SWCLK Circuit Connections

Pin Name	Description	Recommended Pin Connection
SWCLK	Serial wire clock pin	Pull-up resistor 1kΩ

32.7.1 Cortex Debug Connector (10-pin)

For debuggers and/or programmers that support the Cortex Debug Connector (10-pin) interface the signals should be connected as shown in the following figure and detailed table.

Figure 32-10.Cortex Debug Connector (10-pin)

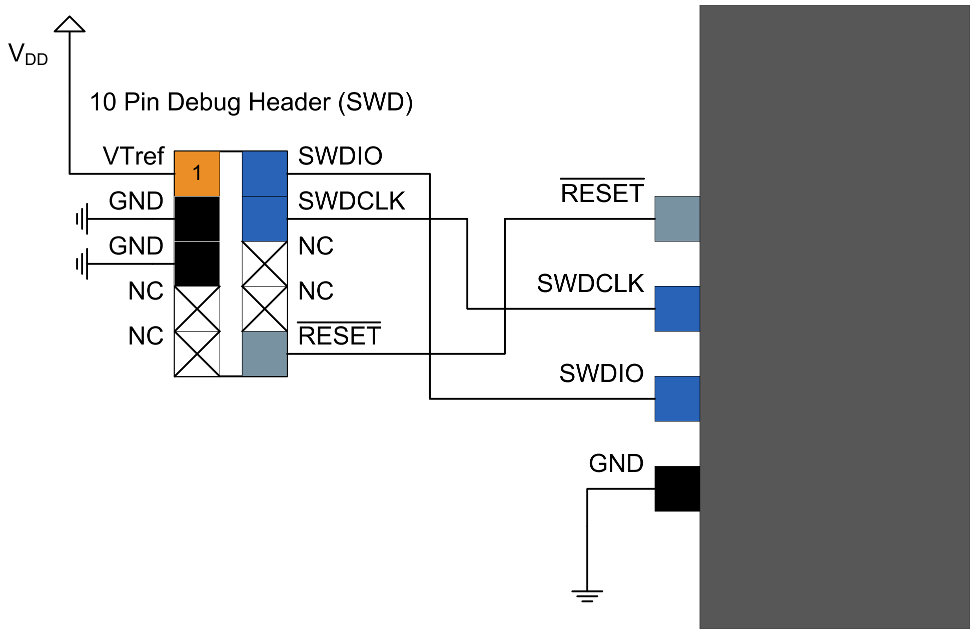


Table 32-8. Cortex Debug Connector (10-pin)

Signal Name	Description
SWDCLK	Serial wire clock pin
SWDIO	Serial wire bidirectional data pin
$\overline{\text{RESET}}$	Target device reset pin, active low
VTref	Target voltage sense, should be connected to the device V_{DD}
GND	Ground

32.7.2 10-pin JTAGICE3 Compatible Serial Wire Debug Interface

The JTAGICE3 debugger and programmer does not support the Cortex Debug Connector (10-pin) directly, hence a special pinout is needed to directly connect the device to the JTAGICE3, alternatively one can use the JTAGICE3 squid cable and manually match the signals between the JTAGICE3 and the device. The following figure describes how to connect a 10-pin header that support connecting the JTAGICE3 directly to the device without the need for a squid cable.

To connect the JTAGICE3 programmer and debugger to the device, one can either use the JTAGICE3 squid cable, or use a 10-pin connector as shown in the figure with details given in the table to connect to the target using the JTAGICE3 cable directly.

Figure 32-11.10-pin JTAGICE3 Compatible Serial Wire Debug Interface

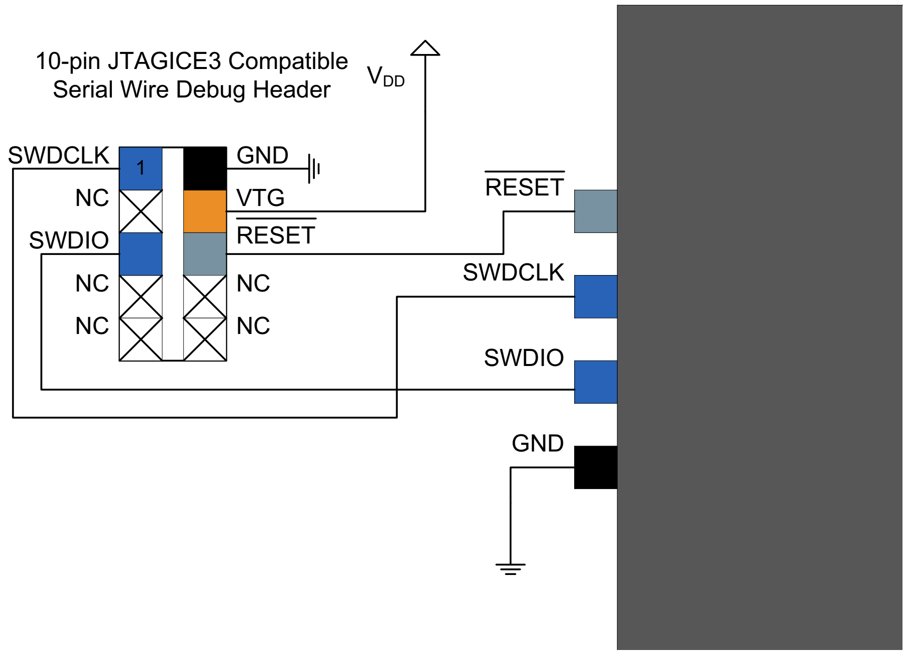


Table 32-9. 10-pin JTAGICE3 Compatible Serial Wire Debug Interface

Signal Name	Description
SWDCLK	Serial wire clock pin
SWDIO	Serial wire bidirectional data pin
$\overline{\text{RESET}}$	Target device reset pin, active low
VTG	Target voltage sense, should be connected to the device V_{DD}
GND	Ground

32.7.3 20-pin IDC JTAG Connector

For debuggers and/or programmers that support the 20-pin IDC JTAG Connector, e.g. the SAM-ICE, the signals should be connected as shown in the figure with details described in the table.

Figure 32-12.20-pin IDC JTAG Connector

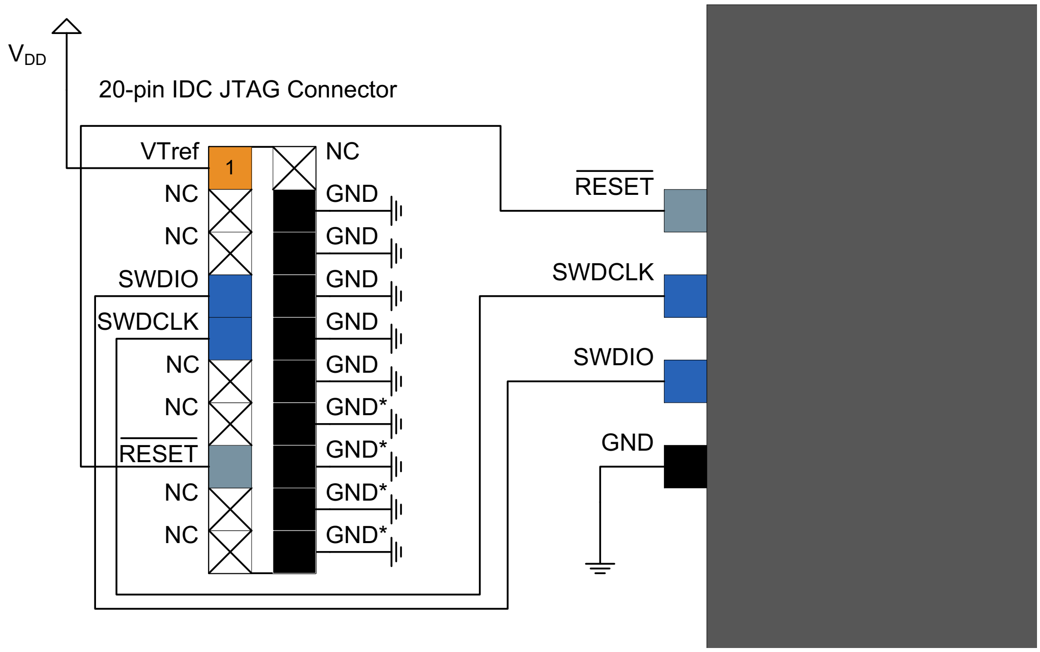


Table 32-10. 20-pin IDC JTAG Connector

Signal Name	Description
SWCLK	Serial wire clock pin
SWDIO	Serial wire bidirectional data pin
$\overline{RESET}$	Target device reset pin, active low
VTref	Target voltage sense, should be connected to the device V_{DD}
GND	Ground
GND*	These pins are reserved for firmware extension purposes. They can be left open or connected to GND in normal debug environment. They are not essential for SWD in general.

33. Errata

33.1 Revision A

Not Sampled

33.2 Revision B

33.2.1 DSU

1 - The MBIST ""Pause-on-Error"" feature is not functional on this device. Errata reference: 14324

Fix/Workaround: Do not use the ""Pause-on-Error"" feature.

33.2.2 DMAC

1 - If data is written to CRCDATAIN in two consecutive instructions, the CRC computation may be incorrect. Errata reference: 13507

Fix/Workaround:

Add a NOP instruction between each write to CRCDATAIN register.

33.2.3 NVMCTRL

1 - Default value of MANW in NVM.CTRLB is 0. Errata reference: 13134

This can lead to spurious writes to the NVM if a data write is done through a pointer with a wrong address corresponding to NVM area.

Fix/Workaround:

Set MANW in the NVM.CTRLB to 1 at startup

2 - When external reset is active it causes a high leakage current on VDDIO. Errata reference: 13446

Fix/Workaround:

Minimize the time external reset is active.

33.2.4 Device

1 - Internal BOD12 could be re-enabled too early on some parts when leaving standby sleep mode, this could lead to a device reset with BOD12 as reset cause when leaving standby mode. Errata reference: 15513

Fix/Workaround:

Disable BOD12 by software just before entering standby sleep mode writing 0x00000004 value at 0x40000838 location and re-enable it writing 0x00000006 at 0x40000838 when exiting sleep mode.

2 - The SYSTICK calibration value is incorrect. Errata reference: 14157

Fix/Workaround:

The correct SYSTICK calibration value is 0x40000000. This value should not be used to initialize the Systick RELOAD value register, which should be initialized instead with a value depending on the main clock frequency and on the tick period required by the application. For a detailed description of the SYSTICK module, refer to the official ARM Cortex-M0+ documentation.

3 - If APB clock is stopped and GCLK clock is running, APB read access to read-synchronized registers will freeze the system. The CPU and the DAP AHB-AP are stalled, as a consequence debug operation is impossible. Errata reference: 10416

Fix/Workaround:

Do not make read access to read-synchronized registers when APB clock is stopped and GCLK is running. To recover from this situation, power cycle the device or reset the device using the RESETN pin.

4 - The voltage regulator in low power mode is not functional at temperatures above 85C. Errata reference: 12291

Fix/Workaround:

Enable normal mode on the voltage regulator in standby sleep mode.

Example code:

```
// Set the voltage regulator in normal mode configuration in standby sleep mode
SYSCTRL->VREG.bit.RUNSTDBY = 1;
```

5 - In I2C Slave mode, writing the CTRLB register when in the AMATCH or DRDY interrupt service routines can cause the state machine to reset. Errata reference: 13574

Fix/Workaround:

Write CTRLB.ACKACT to 0 using the following sequence:

```
// If higher priority interrupts exist, then disable so that the
// following two writes are atomic.
SERCOM - STATUS.reg = 0;
SERCOM - CTRLB.reg  = 0;
// Re-enable interrupts if applicable.
```

Write CTRLB.ACKACT to 1 using the following sequence:

```
// If higher priority interrupts exist, then disable so that the
// following two writes are atomic.
SERCOM - STATUS.reg = 0;
SERCOM - CTRLB.reg  = SERCOM_I2CS_CTRLB_ACKACT;
// Re-enable interrupts if applicable.
```

Otherwise, only write to CTRLB in the AMATCH or DRDY interrupts if it is to close out a transaction.

When not closing a transaction, clear the AMATCH interrupt by writing a 1 to its bit position instead of using CTRLB.CMD. The DRDY interrupt is automatically cleared by reading/writing to the DATA register in smart mode. If not in smart mode, DRDY should be cleared by writing a 1 to its bit position.

Code replacements examples:

Current:

```
SERCOM - CTRLB.reg |= SERCOM_I2CS_CTRLB_ACKACT;
```

Change to:

```
// If higher priority interrupts exist, then disable so that the
// following two writes are atomic.
SERCOM - STATUS.reg = 0;
SERCOM - CTRLB.reg  = SERCOM_I2CS_CTRLB_ACKACT;
// Re-enable interrupts if applicable.
```

Current:

```
SERCOM - CTRLB.reg &= ~SERCOM_I2CS_CTRLB_ACKACT;
```

Change to:

```
// If higher priority interrupts exist, then disable so that the
// following two writes are atomic.
SERCOM - STATUS.reg = 0;
SERCOM - CTRLB.reg  = 0;
// Re-enable interrupts if applicable.
```

Current:

```
/* ACK or NACK address */
SERCOM - CTRLB.reg |= SERCOM_I2CS_CTRLB_CMD(0x3);
```

Change to:

```
// CMD=0x3 clears all interrupts, so to keep the result similar,
// PREC is cleared if it was set.
if (SERCOM - INTFLAG.bit.PREC) SERCOM - INTFLAG.reg =
SERCOM_I2CS_INTFLAG_PREC;
SERCOM - INTFLAG.reg = SERCOM_I2CS_INTFLAG_AMATCH;
```

6 - If the external XOSC32K is broken, neither the external pin RST nor the GCLK software reset can reset the GCLK generators using XOSC32K as source clock. Errata reference: 12164

Fix/Workaround:

Do a power cycle to reset the GCLK generators after an external XOSC32K failure.

33.2.5 DFLL48M

1 - The DFLL clock must be requested before being configured otherwise a write access to a DFLL register can freeze the device. Errata reference: 9905

Fix/Workaround:

Write a zero to the DFLL ONDEMAND bit in the DFLLCTRL register before configuring the DFLL module.

2 - If the DFLL48M reaches the maximum or minimum COARSE or FINE calibration values during the locking sequence, an out of bounds interrupt will be generated. These interrupts will be generated even if the final calibration values at DFLL48M lock are not at maximum or minimum, and might therefore be false out of bounds interrupts. Errata reference: 10669

Fix/Workaround:

Check that the lockbits: DFLLLCKC and DFLLLCKF in the SYSCTRL Interrupt Flag Status and Clear register (INTFLAG) are both set before enabling the DFLLOOB interrupt.

33.2.6 EIC

1 - When the EIC is configured to generate an interrupt on a low level or rising edge or both edges (CONFIGn.SENSEx) with the filter enabled (CONFIGn.FILTENx), a spurious flag might appear for the dedicated pin on the INTFLAG.EXTINT[x] register as soon as the EIC is enabled using CTRLA ENABLE bit. Errata reference: 15341

Fix/Workaround:

Clear the INTFLAG bit once the EIC enabled and before enabling the interrupts.

33.2.7 SERCOM

1 - The I2C Slave SCL Low Extend Time-out (CTRLA.SEXTTOEN) and Master SCL Low Extend Time-out (CTRLA.MEXTTOEN) cannot be used if SCL Low Time-out (CTRLA.LOWTOUT) is disabled. When SCTRLA.LOWTOUT=0, the GCLK_SERCOM_SLOW is not requested. Errata reference: 12003

Fix/Workaround:

To use the Master or Slave SCL low extend time-outs, enable the SCL Low Time-out (CTRLA.LOWTOUT=1).

2 - In USART autobaud mode, missing stop bits are not recognized as inconsistent sync (ISF) or framing (FERR) errors. Errata reference: 13852

Fix/Workaround:

None

3 - If the SERCOM is enabled in SPI mode with SSL detection enabled (CTRLB.SSDE) and CTRLB.RXEN=1, an erroneous slave select low interrupt (INTFLAG.SSL) can be generated. Errata reference: 13369

Fix/Workaround:

Enable the SERCOM first with CTRLB.RXEN=0. In a subsequent write, set CTRLB.RXEN=1.

4 - In TWI master mode, an ongoing transaction should be stalled immediately when DBGCTRL.DBGSTOP is set and the CPU enters debug mode. Instead, it is stopped when the current byte transaction is completed and the corresponding interrupt is triggered if enabled. Errata reference: 12499

Fix/Workaround:

In TWI master mode, keep DBGCTRL.DBGSTOP=0 when in debug mode.

34. About This Document

34.1 Conventions

34.1.1 Numerical Notation

Table 34-1. Numerical notation

165	Decimal number
0101b	Binary number (example 0b0101 = 5 decimal)
0101	Binary numbers are given without suffix if unambiguous
0x3B24	Hexadecimal number
X	Represents an unknown or don't care value
Z	Represents a high-impedance (floating) state for either a signal or a bus

34.1.2 Memory Size and Type

Table 34-2. Memory Size and Bit Rate

Symbol	Description
kB/kbyte	kilobyte (2^{10} = 1024)
MB/Mbyte	megabyte (2^{20} = 1024*1024)
GB/Gbyte	gigabyte (2^{30} = 1024*1024*1024)
b	bit (binary 0 or 1)
B	byte (8 bits)
1kbit/s	1,000 bit/s rate (not 1,024 bit/s)
1Mbit/s	1,000,000 bit/s rate
1Gbit/s	1,000,000,000 bit/s rate

34.1.3 Frequency and Time

Table 34-3. Frequency and Time

Symbol	Description
kHz	1kHz = 10^3Hz = 1,000Hz
MHz	10^6 = 1,000,000Hz
GHz	10^9 = 1,000,000,000Hz
s	second
ms	millisecond
µs	microsecond
ns	nanosecond

34.1.4 Registers and Bits

Table 34-4. Register and bit mnemonics

R/W	Read/Write accessible register bit. The user can read from and write to this bit.
R	Read-only accessible register bit. The user can only read this bit. Writes will be ignored.
W	Write-only accessible register bit. The user can only write this bit. Reading this bit will return an undefined value.
BIT	Bit names are shown in uppercase. (Example PINA1)
BITS[n:m]	A set of bits from bit n down to m. (Example: PINA3..0 = {PINA3, PINA2, PINA1, PINA0}
Reserved	Reserved bits are unused and reserved for future use. For compatibility with future devices, always write reserved bits to zero when the register is written. Reserved bits will always return zero when read.
PERIPHERAL*i*	If several instances of a peripheral exist, the peripheral name is followed by a number to indicate the number of the instance in the range 0-n. PERIPHERAL*i* denotes one specific instance.
Reset	
SET/CLR	Registers with SET/CLR suffix allows the user to clear and set bits in a register without doing a read-modify-write operation. These registers always come in pairs. Writing a one to a bit in the CLR register will clear the corresponding bit in both registers, while writing a one to a bit in the SET register will set the corresponding bit in both registers. Both registers will return the same value when read. If both registers are written simultaneously, the write to the CLR register will take precedence.

34.2 Acronyms and Abbreviations

Table 34-5 contains acronyms and abbreviations used in this document.

Table 34-5. Acronyms and Abbreviations

Abbreviation	Description
AC	Analog Comparator
ADC	Analog-to-Digital Converter
ADDR	Address
AHB	AMBA Advanced High-performance Bus
APB	AMBA Advanced Peripheral Bus
AREF	Analog reference voltage
AV_{DD}	Analog supply voltage
BLB	Boot Lock Bit
BOD	Brown-out detector
CAL	Calibration
CC	Compare/capture
CLK	Clock
CRC	Cyclic Redundancy Check

CTRL	Control
DAC	Digital to Analog converter
DFLL	Digital Frequency Locked Loop
DSU	Device service unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
EIC	External interrupt controller
EVSYS	Event System
GCLK	Generic clock
GND	Ground
GPIO	General Purpose Input/Output
I^2C	Inter-integrated circuit
IF	Interrupt Flag
INT	Interrupt
IOBUS	I/O Bus
NMI	Non-Maskable Interrupt
NVIC	Nested vector interrupt controller
NVMCTRL	Non-Volatile Memory controller
OSC	Oscillator
PAC	Peripheral access controller
PC	Program counter
PER	Period
PM	Power manager
POR	Power-on reset
PTC	Peripheral touch controller
PWM	Pulse Width Modulation
RAM	Random-access memory
REF	Reference
RMW	Read-modify-write
RTC	Real-time counter
RX	Receiver
SERCOM	Serial communication interface
SMBus	System Management Bus
SP	Stack Pointer
SPI	Serial peripheral interface

SRAM	Static random-access memory
SYSCTRL	System controller
SWD	Single-wire debug
TC	Timer/Counter
TX	Transmitter
ULP	Ultra Low Power
USART	Universal synchronous and asynchronous serial receiver and transmitter
V_{DD}	Digital supply voltage
VREF	Voltage reference
WDT	Watchdog timer
XOSC	Crystal oscillator

35. 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.

35.1 Rev. G – 08/2016

"Ordering Information" on page 4	
	Added "Device Identification" on page 4
"I/O Multiplexing and Considerations" on page 10	
	Added SWDIO in IO multiplexing table
"I/O Multiplexing and Considerations" on page 10	
	Added a table note related to PA24 and PA25 pins
"Memories" on page 18	
	Updated "Embedded Memories" on page 18: Added reference to the EEPROM Updated the Table : Set as "Reserved" 73:64 bit position Added "NVM Calibration and Auxiliary Space" on page 18
"PAC – Peripheral Access Controller" on page 27	
	Updated the PAC overiew: CLK_PAC0_APB and CLK_PAC1_APB are enabled at reset while CLK_PAC2_APB is disabled at reset. Added "Register Description" on page 28
"DSU – Device Service Unit" on page 36	
	Updated the table in "System Services Availability When Accessed Externally" on page 45. MBIST is not available when the device is protected from the external address space. Yes has been changed to No in the table. Updated the content in "Testing of Onboard Memories (MBIST)" on page 44 about the MBIST run time
"GCLK – Generic Clock Controller" on page 85	
	"Signal Description" on page 86: Updated the signal name to GCLK_IO[5:0] Updated all the tables according to generic clock generator [5:0] Updated GENDIV GENDIV Reset value tables
"SYSCTRL – System Controller" on page 140	
	Updated the content in "Open-Loop Operation" on page 149 and in "Closed-Loop Operation" on page 149 Updated "Debug Operation" on page 145 about debugger cold-plugging and hot plugging
"WDT – Watchdog Timer" on page 205	
	Updated the content in "Debug Operation" on page 206
"RTC – Real-Time Counter" on page 224	
	Updated the content in "Clock/Calendar (Mode 2)" on page 228
"ADC – Analog-to-Digital Converter" on page 610	

	"Prescaler" on page 613: Added the formula of the *PropagationDelay* for Free Running Mode Updated the Table 29-1 Updated the description of *Interrupts*
"Electrical Characteristics" on page 648	
	Updated Table 30-1 "Absolute Maximum Ratings: V_{PIN}: min and max changed respectively from GND-0.3V to GND-0.6V and GND+0.3V to GND+0.6V Updated Table 30-10: Normal I/O Pins Characteristics, the Pull-up - Pull-down resistance excludes PA24 and PA25 pins. Those USB pins have different pull-up and pull-down Added "Injection Current" on page 658 Updated "Analog-to-Digital (ADC) Characteristics" on page 662: Removed *"Conversion Speed"* from the Table 30-19 *"Operating Conditions"* Added Table 30-31 *"Flash Erase and Programming Current"* Updated Table 30-4: Renamed the table to "Supply Slew Rates" and added "Max Fall Rate = 0.05 V/µs Updated the Table 30-2 *"General Operating Conditions"*: Voltage drop caution added

35.2 Rev. F – 05/2016

"SYSCTRL – System Controller" on page 140	
	"Principle of Operation" on page 145: Updated the last paragraph to "To force the oscillator to run in standby mode except for DFLL and DPLL, the RUNSTDBY bit must be written to one" DFLLCTRL: Removed RUNSTDBY bit DPLLCTRLA: Removed RUNSTDBY bit
"ADC – Analog-to-Digital Converter" on page 610	
	Table 29-12 and Table 29-13: Updated the content in the two respective tables
"Electrical Characteristics" on page 648	
	Table 30-20 and Table 30-21: The device has single power domain. The table notes updated accordingly.

35.3 Rev. E – 04/2016

"Memories" on page 18	
	"NVM User Row Mapping" on page 20: Added "Production setting" in the table

35.4 Rev. D – 03/2016

"Electrical Characteristics" on page 648	
	Table 30-2: V_{DD} min updated from 1.62V to 2.4V.
"Schematic Checklist" on page 687	

	"Operation in Noisy Environment" on page 687: This section has been added "Programming and Debug Ports" on page 693: The pull-up resistor to SWCLK pin is 1kΩ
Errata:	
	Revision B: Added errata related to EIC. Errata reference 15341

35.5 Rev. C – 02/2016

"Memories" on page 18	
	"NVM Software Calibration Row Mapping" on page 21: Removed the text "CRr_SYSCTRL_DFLLVAL"
"TC – Timer/Counter" on page 570	
	"Capture Operations" on page 579: Removed timestamp from "Event Capture Action"

35.6 Rev. B – 01/2016

"DSU – Device Service Unit" on page 36	
	Updated Bit 21-16 - SERIES [5:0] in DID. The value of the field is 0x02.
"SYSCTRL – System Controller" on page 140	
	Updated description in "Drift Compensation" on page 151.
"NVMCTRL – Non-Volatile Memory Controller" on page 352	
	Removed the text related to "AUTOWS bit" from "Clocks" on page 353
"PORT" on page 375	
	Updated Bit 17 - INEN in WRCONFIGn. This bit determines the new value written to PINCFGy.INEN for all pins selected by.....
"TC – Timer/Counter" on page 570	
	TC instances are paired ..... starting from TC1 (not from TC0) in "Clocks" on page 572
"Electrical Characteristics" on page 648	
	"Digital Frequency Locked Loop (DFLL48M) Characteristics" on page 675: Removed note from Table 30-37. "Digital Frequency Locked Loop (DFLL48M) Characteristics" on page 675: Added note to Table 30-37. "NVM Characteristics" on page 670: Updated the Table 30-27 "Power Consumption" on page 651: Added Max values in the Table 30-7

35.7 Rev. A – 08/2015

	Initial revision

Table of Contents