



# ATSAMHAXEXXA

---

## Low-Power Automotive SiP Product with 32-bit ARM® Cortex® M0+ Processor and LIN SBC

---

### Introduction

---

The SAMHA0/1 is a low-power automotive system-in-package (SiP) product series using the 32-bit ARM® Cortex® M0+ processor featuring 32 pins with up to 64KB of Flash, up to 2KB of read-while-write data flash, and up to 8KB of SRAM. The SAM HA1 devices operate at a maximum frequency of 48MHz and reach 2.46 Coremark/MHz. They also include a LIN system basis chip (SBC) with a fully integrated LIN transceiver designed in compliance with the LIN specifications 2.0, 2.1, 2.2, 2.2A and SAEJ2602-2, and a 3.3V/85mA voltage regulator.

### Features

---

#### Microcontroller

- Processor
  - ARM Cortex-M0+ CPU running at up to 48MHz
    - Single-cycle hardware multiplier
    - Micro Trace Buffer (MTB)
- Memories
  - 16/32/64KB in-system self-programmable Flash
  - 0.5/1/2KB Read-While-Write (RWW) Flash section
  - 4/4/8KB 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)
  - Up to 14 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
  - 8-Channel Direct Memory Access Controller (DMAC)
  - 12-Channel Event System
  - Up to five 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
- Among other uses, the additional TC can be used for internal scheduling and has no compare outputs connected.
- Three 16-bit Timer/Counters for Control (TCC), with extended functions:
  - Up to four compare channels with optional complementary output
  - Generation of synchronized pulse width modulation (PWM) pattern across port pins
  - Deterministic fault protection, fast decay and configurable dead-time between complementary output
- Dithering for enhancing resolution with up to 5 bit and reduce quantization error
- 32-bit Real Time Counter (RTC) with clock/calendar function
- Watchdog Timer (WDT)
- CRC-32 generator
- Up to four Serial Communication Interfaces (SERCOM), each configurable to operate as either:
  - USART with full-duplex and single-wire half-duplex configuration
  - I<sup>2</sup>C up to 3.4MHz (only available on specific SERCOMs, see I/O Multiplexing and Considerations chapter for details)
  - SPI
- One 12-bit, 350ksps Analog-to-Digital Converter (ADC) with 6 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
- 10-bit, 350ksps Digital-to-Analog Converter (DAC)
- One Analog Comparator (AC) with window compare function
- Peripheral Touch Controller (PTC)
  - Note:** The PTC is not available on SAMHA0 devices
  - Up to 28-Channel capacitive touch and proximity sensing
- I/O
  - Up to 19 programmable I/O pins
- Packages
  - 32-pin VQFN

## LIN SBC<sup>(1)</sup>

- Up to 40V Supply Voltage
- Operating Voltage  $V_S=5V$  to 28V
- Supply Current
  - Typically 9 $\mu$ A supply current during Sleep mode
  - Typically 47 $\mu$ A supply current in Silent mode
  - Very low current consumption at low supply voltages ( $2V < V_S < 5.5V$ ): typically 130 $\mu$ A
- Linear Low-Drop Voltage Regulator, 85mA Current Capability
  - MLC<sup>(2)</sup> capacitor with 0 $\Omega$  ESR
  - Normal, Fail-safe and Silent modes
  - $V_{CC}=3.3V\pm 2\%$

- Sleep mode:  $V_{CC}$  is switched off
- $V_{CC}$  undervoltage detection with open drain reset output (NRES, 4ms reset time)
- Voltage Regulator is Short-Circuit and Overtemperature Protected
- LIN Physical Layer in Compliance with LIN 2.0, 2.1, 2.2, 2.2A and SAEJ2602-2
- Wake-Up Capability Via LIN Bus (100 $\mu$ s dominant)
- Wake-Up Source Recognition
- TXD Time-Out Timer
- Bus pin is Overtemperature and Short-Circuit Protected vs. GND and Battery
- Advanced EMC and ESD Performance
- Fulfills the OEM “Hardware Requirements for LIN in Automotive Applications Rev.1.3”
- Interference and Damage Protection Cccording to ISO7637

**Note:**

1. LIN SBC: LIN system basis chip including LIN transceiver and voltage regulator
2. Multilayer ceramic

## Table of Contents

Introduction.....	1
Features.....	1
1. Description.....	12
2. Configuration Summary.....	14
3. LIN System Basis Chip (LIN-SBC) Block.....	16
3.1. Features.....	16
3.2. Description.....	16
3.3. Pin Description.....	17
3.4. Functional Description.....	19
4. Block Diagram.....	26
5. Pinout.....	28
5.1. SAM HA1E - VQFN32.....	28
6. Signal Descriptions List.....	29
7. I/O Multiplexing and Considerations.....	31
7.1. Multiplexed Signals.....	31
7.2. Other Functions.....	32
8. Power Supply and Start-Up Considerations.....	34
8.1. Power Domain Overview.....	34
8.2. Power Supply Considerations.....	34
8.3. Power-Up.....	36
8.4. Power-On Reset and Brown-Out Detector.....	36
9. Product Mapping.....	38
10. Automotive Quality Grade.....	39
11. Data Retention.....	40
12. Memories.....	41
12.1. Embedded Memories.....	41
12.2. Physical Memory Map.....	41
12.3. NVM Calibration and Auxiliary Space.....	42
13. Processor And Architecture.....	45
13.1. Cortex M0+ Processor.....	45
13.2. Nested Vector Interrupt Controller.....	46
13.3. Micro Trace Buffer.....	48

13.4. High-Speed Bus System.....	49
13.5. AHB-APB Bridge.....	51
13.6. PAC - Peripheral Access Controller.....	52
<b>14. Peripherals Configuration Summary.....</b>	<b>66</b>
<b>15. DSU - Device Service Unit.....</b>	<b>68</b>
15.1. Overview.....	68
15.2. Features.....	68
15.3. Block Diagram.....	69
15.4. Signal Description.....	69
15.5. Product Dependencies.....	69
15.6. Debug Operation.....	70
15.7. Chip Erase.....	72
15.8. Programming.....	73
15.9. Intellectual Property Protection.....	73
15.10. Device Identification.....	75
15.11. Functional Description.....	76
15.12. Register Summary.....	81
15.13. Register Description.....	83
<b>16. Clock System.....</b>	<b>107</b>
16.1. Clock Distribution.....	107
16.2. Synchronous and Asynchronous Clocks.....	108
16.3. Register Synchronization.....	108
16.4. Enabling a Peripheral.....	113
16.5. Disabling a Peripheral.....	113
16.6. On-demand, Clock Requests.....	113
16.7. Power Consumption vs. Speed.....	114
16.8. Clocks after Reset.....	114
<b>17. GCLK - Generic Clock Controller.....</b>	<b>115</b>
17.1. Overview.....	115
17.2. Features.....	115
17.3. Block Diagram.....	115
17.4. Signal Description.....	116
17.5. Product Dependencies.....	116
17.6. Functional Description.....	117
17.7. Register Summary.....	123
17.8. Register Description.....	123
<b>18. PM – Power Manager.....</b>	<b>137</b>
18.1. Overview.....	137
18.2. Features.....	137
18.3. Block Diagram.....	138
18.4. Signal Description.....	138
18.5. Product Dependencies.....	138
18.6. Functional Description.....	140

18.7. Register Summary.....	148
18.8. Register Description.....	148
<b>19. SYSCTRL – System Controller.....</b>	<b>169</b>
19.1. Overview.....	169
19.2. Features.....	169
19.3. Block Diagram.....	171
19.4. Signal Description.....	171
19.5. Product Dependencies.....	171
19.6. Functional Description.....	173
19.7. Register Summary.....	189
19.8. Register Description.....	191
<b>20. WDT – Watchdog Timer.....</b>	<b>234</b>
20.1. Overview.....	234
20.2. Features.....	234
20.3. Block Diagram.....	235
20.4. Signal Description.....	235
20.5. Product Dependencies.....	235
20.6. Functional Description.....	236
20.7. Register Summary.....	242
20.8. Register Description.....	242
<b>21. RTC – Real-Time Counter.....</b>	<b>253</b>
21.1. Overview.....	253
21.2. Features.....	253
21.3. Block Diagram.....	254
21.4. Signal Description.....	254
21.5. Product Dependencies.....	254
21.6. Functional Description.....	256
21.7. Register Summary.....	262
21.8. Register Description.....	264
<b>22. DMAC – Direct Memory Access Controller.....</b>	<b>297</b>
22.1. Overview.....	297
22.2. Features.....	297
22.3. Block Diagram.....	299
22.4. Signal Description.....	299
22.5. Product Dependencies.....	299
22.6. Functional Description.....	300
22.7. Register Summary.....	321
22.8. Register Description.....	322
22.9. Register Summary - SRAM.....	353
22.10. Register Description - SRAM.....	353
<b>23. EIC – External Interrupt Controller.....</b>	<b>361</b>
23.1. Overview.....	361
23.2. Features.....	361

23.3. Block Diagram.....	361
23.4. Signal Description.....	362
23.5. Product Dependencies.....	362
23.6. Functional Description.....	363
23.7. Register Summary.....	368
23.8. Register Description.....	368
<b>24. NVMCTRL – Nonvolatile Memory Controller.....</b>	<b>380</b>
24.1. Overview.....	380
24.2. Features.....	380
24.3. Block Diagram.....	380
24.4. Signal Description.....	381
24.5. Product Dependencies.....	381
24.6. Functional Description.....	382
24.7. Register Summary.....	389
24.8. Register Description.....	389
<b>25. PORT - I/O Pin Controller.....</b>	<b>403</b>
25.1. Overview.....	403
25.2. Features.....	403
25.3. Block Diagram.....	404
25.4. Signal Description.....	404
25.5. Product Dependencies.....	404
25.6. Functional Description.....	407
25.7. Register Summary.....	413
25.8. PORT Pin Groups and Register Repetition.....	415
25.9. Register Description.....	415
<b>26. EVSYS – Event System.....</b>	<b>436</b>
26.1. Overview.....	436
26.2. Features.....	436
26.3. Block Diagram.....	436
26.4. Signal Description.....	437
26.5. Product Dependencies.....	437
26.6. Functional Description.....	438
26.7. Register Summary.....	443
26.8. Register Description.....	444
<b>27. SERCOM – Serial Communication Interface.....</b>	<b>457</b>
27.1. Overview.....	457
27.2. Features.....	457
27.3. Block Diagram.....	458
27.4. Signal Description.....	458
27.5. Product Dependencies.....	458
27.6. Functional Description.....	460
<b>28. SERCOM USART.....</b>	<b>466</b>
28.1. Overview.....	466

28.2. USART Features.....	466
28.3. Block Diagram.....	467
28.4. Signal Description.....	467
28.5. Product Dependencies.....	467
28.6. Functional Description.....	469
28.7. Register Summary.....	481
28.8. Register Description.....	482
<b>29. SERCOM SPI – SERCOM Serial Peripheral Interface.....</b>	<b>505</b>
29.1. Overview.....	505
29.2. Features.....	505
29.3. Block Diagram.....	506
29.4. Signal Description.....	506
29.5. Product Dependencies.....	506
29.6. Functional Description.....	508
29.7. Register Summary.....	517
29.8. Register Description.....	518
<b>30. SERCOM I<sup>2</sup>C – Inter-Integrated Circuit.....</b>	<b>538</b>
30.1. Overview.....	538
30.2. Features.....	538
30.3. Block Diagram.....	539
30.4. Signal Description.....	539
30.5. Product Dependencies.....	539
30.6. Functional Description.....	541
30.7. Register Summary - I2C Slave.....	560
30.8. Register Description - I <sup>2</sup> C Slave.....	560
30.9. Register Summary - I2C Master.....	578
30.10. Register Description - I <sup>2</sup> C Master.....	579
<b>31. TC – Timer/Counter.....</b>	<b>600</b>
31.1. Overview.....	600
31.2. Features.....	600
31.3. Block Diagram.....	601
31.4. Signal Description.....	601
31.5. Product Dependencies.....	602
31.6. Functional Description.....	603
31.7. Register Summary.....	615
31.8. Register Description.....	618
<b>32. TCC – Timer/Counter for Control Applications.....</b>	<b>642</b>
32.1. Overview.....	642
32.2. Features.....	642
32.3. Block Diagram.....	643
32.4. Signal Description.....	644
32.5. Product Dependencies.....	644
32.6. Functional Description.....	646
32.7. Register Summary.....	678



32.8. Register Description.....	680
<b>33. ADC – Analog-to-Digital Converter.....</b>	<b>726</b>
33.1. Overview.....	726
33.2. Features.....	726
33.3. Block Diagram.....	727
33.4. Signal Description.....	727
33.5. Product Dependencies.....	728
33.6. Functional Description.....	729
33.7. Register Summary.....	739
33.8. Register Description.....	740
<b>34. AC – Analog Comparators.....</b>	<b>765</b>
34.1. Overview.....	765
34.2. Features.....	765
34.3. Block Diagram.....	766
34.4. Signal Description.....	766
34.5. Product Dependencies.....	766
34.6. Functional Description.....	768
34.7. Register Summary.....	778
34.8. Register Description.....	778
<b>35. DAC – Digital-to-Analog Converter.....</b>	<b>795</b>
35.1. Overview.....	795
35.2. Features.....	795
35.3. Block Diagram.....	795
35.4. Signal Description.....	795
35.5. Product Dependencies.....	795
35.6. Functional Description.....	797
35.7. Register Summary.....	801
35.8. Register Description.....	801
<b>36. PTC - Peripheral Touch Controller.....</b>	<b>812</b>
36.1. Overview.....	812
36.2. Features.....	812
36.3. Block Diagram.....	813
36.4. Signal Description.....	814
36.5. System Dependencies.....	814
36.6. Functional Description.....	815
<b>37. Electrical Characteristics.....</b>	<b>817</b>
37.1. Disclaimer.....	817
37.2. Absolute Maximum Ratings.....	817
37.3. LIN System Basis Chip (LIN-SBC) Block.....	819
37.4. General Operating Ratings.....	825
37.5. Thermal Considerations.....	826
37.6. Supply Characteristics.....	827
37.7. Maximum Clock Frequencies.....	827

37.8. Power Consumption.....	829
37.9. Peripheral Power Consumption.....	831
37.10. HA1_I/O Pin Characteristics.....	833
37.11. Analog Characteristics.....	837
37.12. NVM Characteristics.....	846
37.13. Oscillators Characteristics.....	846
37.14. PTC Typical Characteristics.....	855
37.15. Timing Characteristics.....	857
<b>38. Schematic Checklist.....</b>	<b>862</b>
38.1. Introduction.....	862
38.2. Power Supply.....	862
38.3. External Analog Reference Connections.....	863
38.4. External Reset Circuit.....	864
38.5. Clocks and Crystal Oscillators.....	865
38.6. Unused or Unconnected Pins.....	868
38.7. Programming and Debug Ports.....	868
<b>39. Typical Application Diagram.....</b>	<b>872</b>
<b>40. Errata.....</b>	<b>873</b>
40.1. Die Revision F.....	873
<b>41. Conventions.....</b>	<b>876</b>
41.1. Numerical Notation.....	876
41.2. Memory Size and Type.....	876
41.3. Frequency and Time.....	876
41.4. Registers and Bits.....	877
<b>42. Acronyms and Abbreviations.....</b>	<b>878</b>
<b>43. Data Sheet Revision History.....</b>	<b>881</b>
43.1. Revision A - 12/2017.....	881
<b>44. Packaging Information.....</b>	<b>882</b>
44.1. Package Marking Information.....	882
The Microchip Web Site.....	886
Customer Change Notification Service.....	886
Customer Support.....	886
Product Identification System.....	887
Microchip Devices Code Protection Feature.....	888
Legal Notice.....	888

Trademarks.....	889
Quality Management System Certified by DNV.....	889
Worldwide Sales and Service.....	890

## 1. Description

The SAMHA0/1 is a low-power automotive system-in-package (SiP) product series using the 32-bit ARM® Cortex® M0+ processor featuring 32 pins with up to 64KB of Flash, up to 2KB of read-while-write data flash, and up to 8KB of SRAM. The SAMHA0/1 device operates at a maximum frequency of 48MHz and reach 2.46 Coremark/MHz. They also include a LIN system basis chip (SBC) with a fully integrated LIN transceiver designed in compliance with the LIN specifications 2.0, 2.1, 2.2, 2.2A and SAEJ2602-2, and a 3.3V/85mA voltage regulator.

Combining a microcontroller, a LIN transceiver, and a voltage regulator makes it possible to develop simple but powerful slave nodes in LIN bus systems. Sleep mode and silent mode guarantee minimized current consumption even in the event of a floating or a short-circuited LIN bus.

The series is qualified in compliance with AEC Q-100 (−40 to +105°C  $T_{case}$ ).

All devices in the product series include intelligent and flexible peripherals, the Event System for interperipheral signaling, and support for capacitive touch button, slider and wheel user interfaces.

The SAMHA0/1 provides the following features: In-system programmable Flash, 8-channel direct memory access (DMA) controller, 12-channel Event System, programmable interrupt controller, 19 programmable I/O pins, 32-bit real-time clock and calendar, up to five 16-bit Timer/Counters (TC) and three 24-bit Timer/Counters for Control (TCC). Two of the TCs 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 three timer/counters have extended functions optimized for motor, lighting and other control applications.

The series provides up to five Serial Communication Modules (SERCOM) that each can be configured to act as an USART, UART, SPI, I<sup>2</sup>C up to 3.4MHz, and LIN slave; up to 6-channel 12-bit ADC with programmable gain and optional oversampling and decimation supporting up to 16-bit resolution, one 10-bit 350ksps DAC, two analog comparators with window mode, Peripheral Touch Controller supporting buttons, sliders, wheels and proximity sensing; 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 SAMHA0/1 devices offer different Sleep modes according to the needs of the application. The LIN SBC offers Sleep and Silent mode and can shut down the voltage regulator to achieve very low power consumption. The microcontroller has 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 except those selected to continue running. The device supports SleepWalking. SleepWalking allows peripherals to wake up from sleep based on predefined conditions, allowing the CPU to wake up only when needed, e.g., when a threshold is crossed or a result is ready. The Peripheral Event System supports synchronous and asynchronous events, allowing peripherals to receive, respond 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 nonintrusive on-chip debug 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 SAMHA0/1 microcontrollers are supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, programmers and evaluation kits.

## 2. Configuration Summary

	SAMHA0/1
Pins	32
General Purpose I/O-pins (GPIOs)	19
Flash	64/32/16KB
Data Flash (read-while-write)	2KB/1KB/512B
SRAM	8/4/4KB
Timer Counter (TC) instances	2+3
Waveform output channels per TC instance	2/0
Timer Counter for Control (TCC) instances	3
Waveform output channels per TCC	7/4/2
DMA channels	8
Serial Communication Interface (SERCOM) instances	5
Analog-to-Digital Converter (ADC) channels	6
Analog Comparators (AC)	1
Digital-to-Analog Converter (DAC) channels	1
Real-Time Counter (RTC)	Yes
RTC alarms	1
RTC compare values	One 32-bit value or two 16-bit values
External Interrupt lines	14
Peripheral Touch Controller (PTC) X and Y lines <b>Note:</b> Is not available on SAMHA0 devices	7x4
Maximum CPU frequency	48MHz
Packages	QFN
Oscillators	32.768kHz crystal oscillator (XOSC32K) 0.4-32MHz crystal oscillator (XOSC) 32.768kHz internal oscillator (OSC32K) 8MHz high-accuracy internal oscillator (OSC8M) 48MHz Digital Frequency Locked Loop (DFLL48M) 96MHz Fractional Digital Phased Locked Loop (FDPLL96M)

# ATSAMHAXEXXA

## Configuration Summary

---

---

	SAMHA0/1
Event System channels	12
SW Debug Interface	Yes
Watchdog Timer (WDT)	Yes

## 3. LIN System Basis Chip (LIN-SBC) Block

### 3.1 Features

- Supply voltage up to 40V
- Operating voltage  $V_S = 5V$  to 28V
- Supply current
  - Sleep mode: typically 9 $\mu$ A
  - Silent mode: typically 47 $\mu$ A
  - Very low current consumption at low supply voltages ( $2V < V_S < 5.5V$ ): typically 130 $\mu$ A
- Linear low-drop voltage regulator, 85mA current capability:
  - MLC (multi-layer ceramic) capacitor with 0 $\Omega$  ESR
  - Normal, fail-safe, and silent mode
    - ATA663231:  $V_{CC} = 3.3V \pm 2\%$
  - Sleep mode: VCC is switched off
- VCC undervoltage detection with open drain reset output (NRES, 4ms reset time)
- Voltage regulator is short-circuit and over-temperature protected
- LIN physical layer according to LIN 2.0, 2.1, 2.2, 2.2A and SAEJ2602-2
- Wake-up capability via LIN bus (100 $\mu$ s dominant)
- TXD time-out timer
- Bus pin is over-temperature and short-circuit protected versus GND and battery
- Advanced EMC and ESD performance
- Fulfills the OEM “Hardware Requirements for LIN in Automotive Applications Rev.1.3”
- Interference and damage protection according to ISO7637
- Qualified according to AEC-Q100

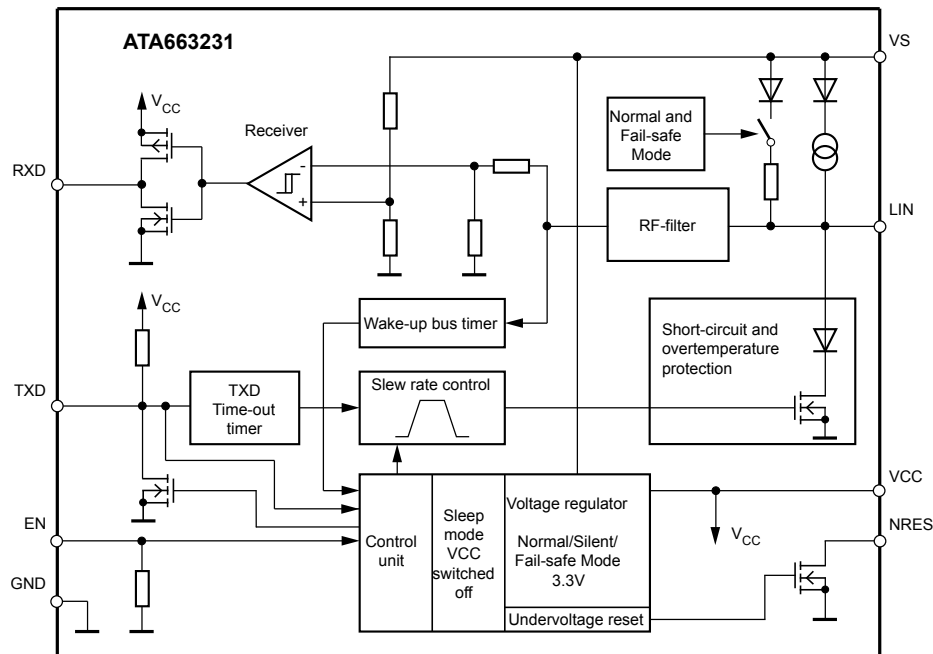
**Note:** LIN SBC: LIN system basis chip including LIN transceiver and voltage regulator.

### 3.2 Description

The system basis chip is a fully integrated LIN transceiver, designed according to the LIN specification 2.0, 2.1, 2.2, 2.2A and SAEJ2602-2, with a low-drop voltage regulator (3.3V/85mA). The combination of voltage regulator and bus transceiver makes it possible to develop simple but powerful slave nodes in LIN bus systems. It is designed to handle the low-speed data communication in vehicles (for example, in convenience electronics). Improved slope control at the LIN driver ensures secure data communication up to 20Kbaud. The bus output is designed to withstand high voltage. Sleep mode and silent mode guarantee minimized current consumption even in the case of a floating or a short-circuited LIN bus.



**Figure 3-1. Block Diagram LIN Transceiver with Integrated Voltage Regulator (SBC)**



### 3.3 Pin Description

#### 3.3.1 Supply Pin (VS)

LIN operating voltage is  $V_S = 5V$  to  $28V$ . Undervoltage detection is implemented to disable transmission if  $V_S$  falls below typ.  $4.5V$ , thereby avoiding false bus messages. After switching on  $V_S$ , the IC starts in fail-safe mode and the voltage regulator is switched on.

The supply current in sleep mode is typically  $9\mu A$  and  $47\mu A$  in silent mode.

#### 3.3.2 Ground Pin (GND)

The IC does not affect the LIN bus in the event of GND disconnection. It is able to handle a ground shift of up to 11.5% of  $V_S$ .

#### 3.3.3 Voltage Regulator Output Pin (VCC)

The internal 3.3V voltage regulator is capable of driving loads up to 85mA, supplying the microcontroller and other ICs on the PCB and is protected against overload by means of current limitation and overtemperature shutdown. Furthermore, the output voltage is monitored and causes a reset signal at the NRES output pin if it drops below a defined threshold  $V_{VCC\_th\_uv\_down}$ .

#### 3.3.4 Undervoltage Reset Output (NRES)

If the  $V_{CC}$  voltage falls below the undervoltage detection threshold  $V_{VCC\_th\_uv\_down}$ , NRES switches to low after  $t_{res\_f}$ . The NRES stays low even if  $V_{CC} = 0V$  because NRES is internally driven from the  $V_S$  voltage. If  $V_S$  voltage ramps down, NRES stays low until  $V_S < 1.5V$  and then becomes highly impedant.

The implemented undervoltage delay keeps NRES low for  $t_{Reset} = 4ms$  after  $V_{CC}$  reaches its nominal value.

### 3.3.5 Bus Pin (LIN)

A low-side driver with internal current limitation and thermal shutdown as well as an internal pull-up resistor according to LIN specification 2.x is implemented. The voltage range is from  $-27V$  to  $+40V$ . This pin exhibits no reverse current from the LIN bus to  $V_S$ , even in the event of a GND shift or  $V_{Bat}$  disconnection. The LIN receiver thresholds comply with the LIN protocol specification.

The fall time (from recessive to dominant) and the rise time (from dominant to recessive) are slope-controlled.

During a short circuit at LIN to  $V_{Bat}$ , the output limits the output current to  $I_{BUS\_LIM}$ . Due to the power dissipation, the chip temperature exceeds  $T_{LINoff}$  and the LIN output is switched off. The chip cools down and after a hysteresis of  $T_{hys}$ , switches the output on again. RXD stays on high because LIN is high. The  $V_{CC}$  regulator works independently during LIN overtemperature switch-off.

During a short circuit from LIN to GND the IC can be switched into sleep or silent mode and even in this case the current consumption is lower than  $100\mu A$  in sleep mode and lower than  $120\mu A$  in silent mode. If the short-circuit disappears, the IC starts with a remote wake-up.

The reverse current is  $< 2\mu A$  at pin LIN during loss of  $V_{Bat}$ . This is optimal behavior for bus systems where some slave nodes are supplied from battery or ignition.

### 3.3.6 Input/Output (TXD)

In normal mode the TXD pin is the microcontroller interface for controlling the state of the LIN output. TXD must be pulled to ground in order to drive the LIN bus low. If TXD is high or unconnected (internal pull-up resistor), the LIN output transistor is turned off and the bus is in the recessive state. If the TXD pin stays at GND level while switching into normal mode, it must be pulled to high level longer than  $10\mu s$  before the LIN driver can be activated. This feature prevents the bus line from being accidentally driven to dominant state after normal mode has been activated (also in case of a short circuit at TXD to GND). During fail-safe mode, this pin is used as output and signals the fail-safe source.

The TXD input has an internal pull-up resistor.

An internal timer prevents the bus line from being driven permanently in the dominant state. If TXD is forced to low longer than  $t_{dom} > 20ms$ , the LIN bus driver is switched to the recessive state. Nevertheless, when switching to sleep mode, the actual level at the TXD pin is relevant.

To reactivate the LIN bus driver, switch TXD to high ( $> 10\mu s$ ).

### 3.3.7 Output Pin (RXD)

In normal mode this pin reports the state of the LIN bus to the microcontroller. LIN high (recessive state) is indicated by a high level at RXD; LIN low (dominant state) is indicated by a low level at RXD.

The output is a push-pull stage switching between VCC and GND. The AC characteristics are measured by an external load capacitor of  $20pF$ .

In silent mode the RXD output switches to high.

### 3.3.8 Enable Input Pin (EN)

The enable input pin controls the operating mode of the device. If EN is high, the circuit is in normal mode, with transmission paths from TXD to LIN and from LIN to RXD both active. The VCC voltage regulator operates with  $3.3V/85mA$  output capability.

If EN is switched to low while TXD is still high, the device is forced to silent mode. No data transmission is then possible, and current consumption is reduced to  $I_{VSSilent}$  typ.  $47\mu A$ . The VCC regulator retains its full functionality.

If EN is switched to low while TXD is low, the device is forced to sleep mode. No data transmission is possible, and the voltage regulator is switched off.

The EN pin provides a pull-down resistor to force the transceiver into recessive mode if EN is disconnected.

### 3.4 Functional Description

#### 3.4.1 Physical Layer Compatibility

Because the LIN physical layer is independent of higher LIN layers (e.g., LIN protocol layer), all nodes with a LIN physical layer according to revision 2.x can be mixed with LIN physical layer nodes based on earlier versions (i.e., LIN 1.0, LIN 1.1, LIN 1.2, LIN 1.3) without any restrictions.

#### 3.4.2 Operating Modes

Figure 3-2. SBC Operating Modes

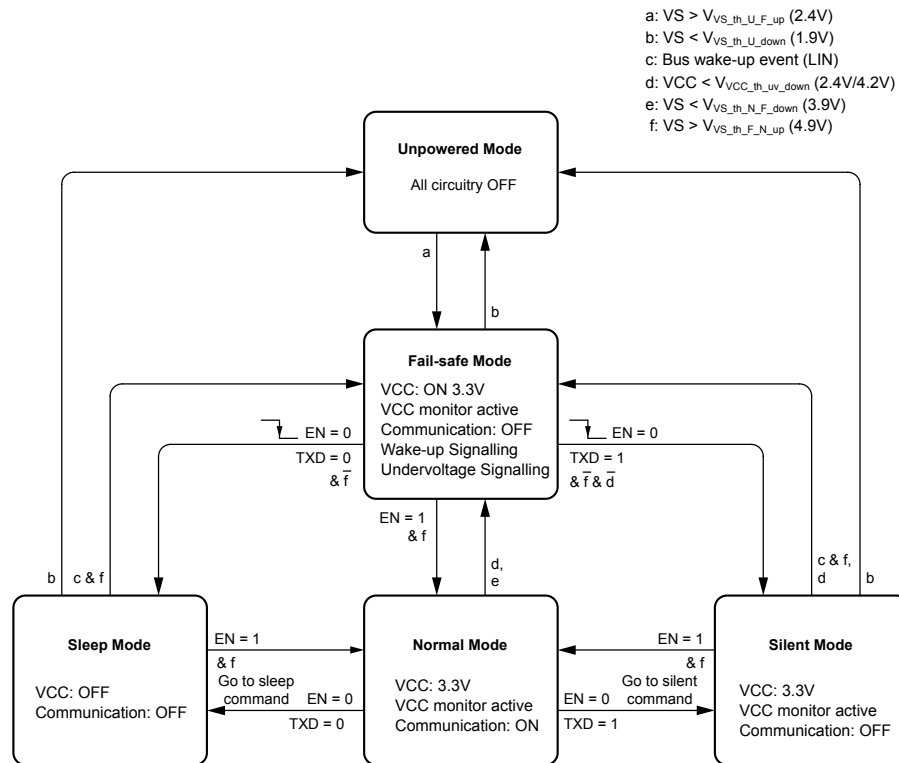


Table 3-1. SBC Operating Modes

Operating Mode	Transceiver	V <sub>CC</sub>	LIN	TXD	RXD
Fail-safe	OFF	3.3V	Recessive	Signaling fail-safe sources (see Table <a href="#">Signaling in Fail-safe Mode</a> )	
Normal	ON	3.3V	TXD-dependent	Follows data transmission	
Silent	OFF	3.3V	Recessive	High	High
Sleep/Unpowered	OFF	0V	Recessive	Low	Low

**Related Links**

[Fail-Safe Mode](#)

**3.4.2.1 Normal Mode**

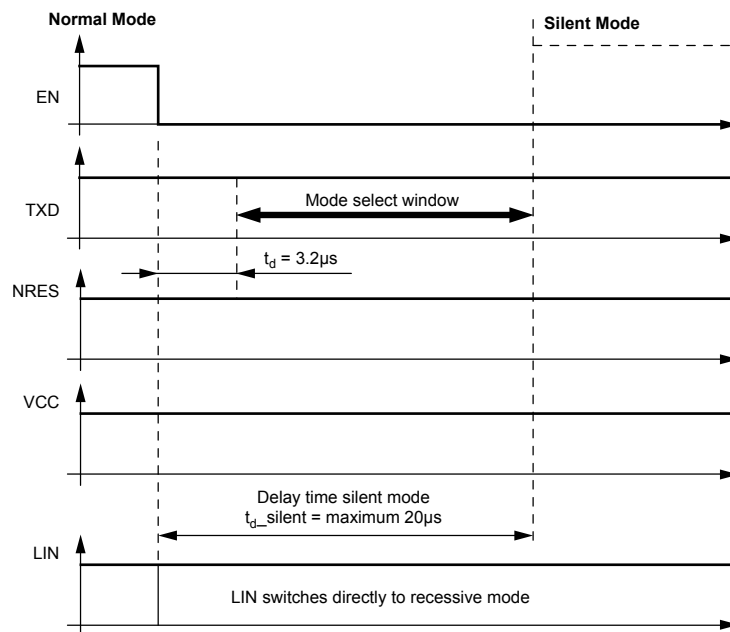
This is the normal transmitting and receiving mode of the LIN Interface, in accordance with LIN specification 2.x.

The VCC voltage regulator operates with 3.3V output voltage, with a low tolerance of  $\pm 2\%$  and a maximum output current of 85mA. If an undervoltage condition occurs, NRES is switched to low and the IC changes its state to fail-safe mode.

**3.4.2.2 Silent Mode**

A falling edge at EN while TXD is high switches the IC into silent mode. The TXD signal has to be logic high during the mode select window. The transmission path is disabled in silent mode. The voltage regulator is active. The overall supply current from  $V_{Bat}$  is a combination of the  $I_{VSSilent} = 47\mu A$  plus the  $V_{CC}$  regulator output current  $I_{VCC}$ .

**Figure 3-3. Switching to Silent Mode**



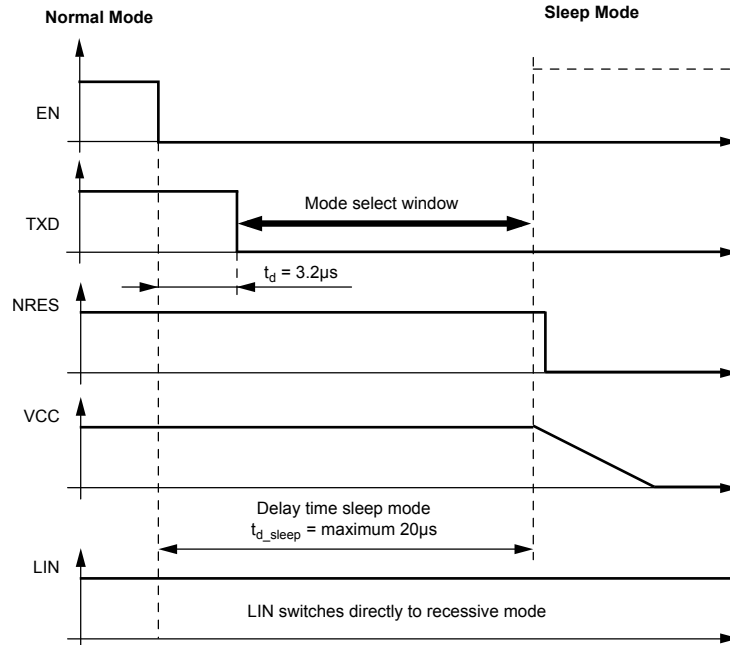
In silent mode the internal slave termination between the LIN pin and VS pin is disabled to minimize the current consumption in case the pin LIN is short-circuited to GND. Only a weak pull-up current (typically 10µA) between the LIN pin and VS pin is present. Silent mode can be activated independently from the current level on pin LIN.

If an undervoltage condition occurs, NRES is switched to low and the SBC changes its state to fail-safe mode.

**3.4.2.3 Sleep Mode**

A falling edge at EN while TXD is low switches the IC into sleep mode. The TXD signal has to be logic low during the mode select window (see the following figure).

**Figure 3-4. Switching to Sleep Mode**



In order to avoid any influence to the LIN pin when switching into sleep mode it is possible to switch the EN up to  $3.2\mu\text{s}$  earlier to low than the TXD. The easiest and best way to do this is by having two falling edges at TXD and EN at the same time.

In sleep mode the transmission path is disabled. Supply current from  $V_{\text{Bat}}$  is typically  $I_{V_{\text{S}}\text{sleep}} = 9\mu\text{A}$ . The  $V_{\text{CC}}$  regulator is switched off; NRES and RXD are low. The internal slave termination between the LIN pin and VS pin is disabled to minimize the current consumption in case the LIN pin is short-circuited to GND. Only a weak pull-up current (typically  $10\mu\text{A}$ ) between the LIN pin and the VS pin is present. The sleep mode can be activated independently from the current level on the LIN pin. Voltage below the LIN pre-wake detection  $V_{\text{LINL}}$  at the LIN pin activates the internal LIN receiver and starts the wake-up detection timer.

If the TXD pin is short-circuited to GND, it is possible to switch to sleep mode via EN after  $t > t_{\text{dom}}$ .

#### 3.4.2.4 Fail-Safe Mode

The device automatically switches to fail-safe mode at system power-up. The voltage regulator is switched on. The NRES output remains low for  $t_{\text{res}} = 4\text{ms}$  and causes the microcontroller to be reset. LIN communication is switched off. The IC stays in this mode until EN is switched to high. The IC then changes to normal mode. A low at NRES switches the IC into fail-safe mode directly. During fail-safe mode the TXD pin is an output and, together with the RXD output pin, signals the fail-safe source.

If the device enters fail-safe mode coming from the normal mode ( $\text{EN}=1$ ) due to an  $V_{\text{S}}$  undervoltage condition

( $V_{\text{S}} < V_{V_{\text{S}}\text{th}_N_F\_down}$ ), it is possible to switch into sleep or silent mode by a falling edge at the EN input. With this feature the current consumption can be further reduced.

A wake-up event from either silent or sleep mode is signalled to the microcontroller using the RXD pin and the TXD pin. A  $V_{\text{S}}$  undervoltage condition is also signalled at these two pins. The coding is shown in the table below.

A wake-up event switches the IC to fail-safe mode.

**Table 3-2. Signaling in Fail-Safe Mode**

Fail-Safe Sources	TXD	RXD
LIN wake-up (LIN pin)	Low	Low
$V_{S_{th}}$ (battery) undervoltage detection ( $V_S < 3.9V$ )	High	Low

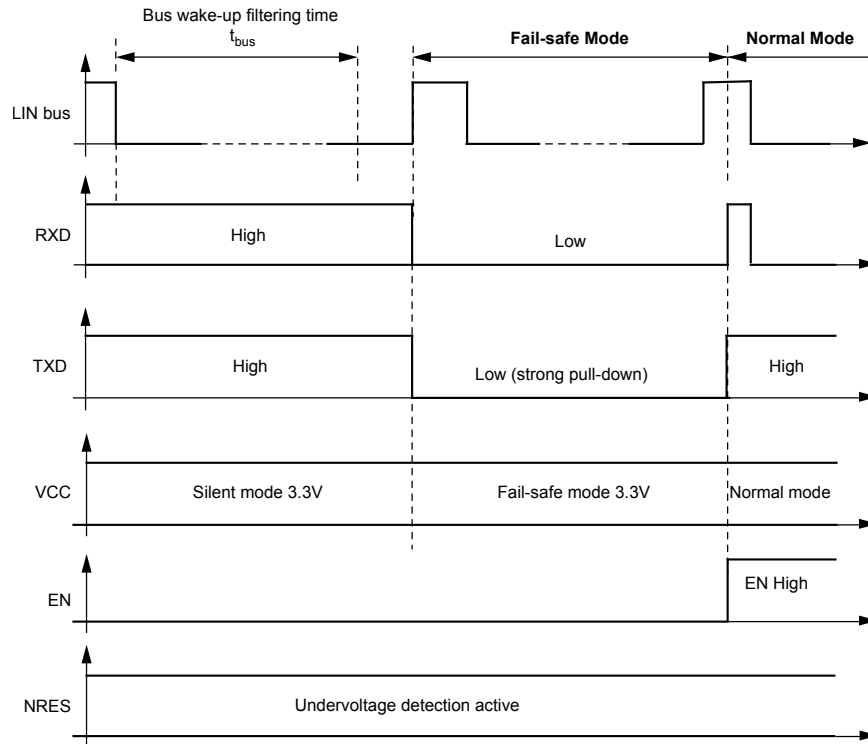
**3.4.3 Wake-Up Scenarios from Silent Mode or Sleep Mode**

**3.4.3.1 Remote Wake-up via LIN Bus**

**Remote Wake-up from Silent Mode**

A remote wake-up from silent mode is only possible if TXD is high. A voltage less than the LIN pre-wake detection  $V_{LINL}$  at the LIN pin activates the internal LIN receiver and starts the wake-up detection timer. A falling edge at the LIN pin followed by a dominant bus level maintained for a certain period of time ( $> t_{bus}$ ) and the following rising edge at pin LIN result in a remote wake-up request. The device switches from silent mode to fail-safe mode, the VCC voltage regulator remains activated and the internal LIN slave termination resistor is switched on. The remote wake-up request is indicated by a low level at the RXD pin and TXD pin (strong pull-down at TXD). EN high can be used to switch directly to normal mode.

**Figure 3-5. LIN Wake-Up from Silent Mode**



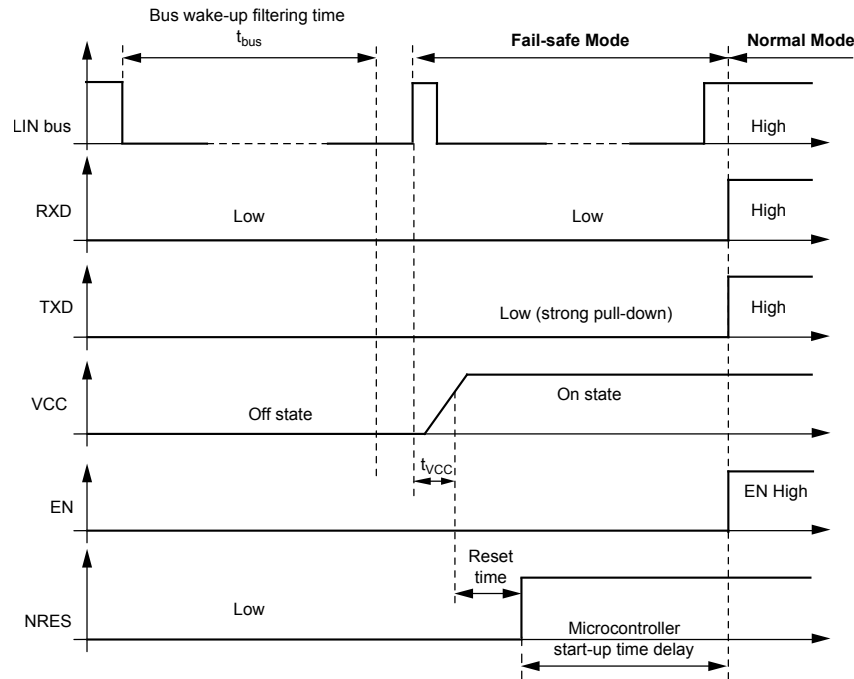
**Remote Wake-up from Sleep Mode**

A falling edge at the LIN pin followed by a dominant bus level maintained for a certain period of time ( $> t_{bus}$ ) and a following rising edge at the LIN pin result in a remote wake-up request, causing the device to switch from sleep mode to fail-safe mode.

The  $V_{CC}$  regulator is activated, and the internal LIN slave termination resistor is switched on. The remote wake-up request is indicated by a low level at RXD and TXD (strong pull-down at TXD) (see [Figure 1-6](#)).

EN high can be used to switch directly from sleep/silent mode to fail-safe mode. If EN is still high after VCC ramp-up and undervoltage reset time, the IC switches to normal mode.

**Figure 3-6. LIN Wake-Up from Sleep Mode**



### 3.4.3.2 Wake-Up Source Recognition

The device can distinguish between different wake-up sources. The wake-up source can be read on the TXD and RXD pin in fail-safe mode. These flags are immediately reset if the microcontroller sets the EN pin to high and the IC is in normal mode.

**Table 3-3. Signaling in Fail-safe Mode**

Fail-Safe Sources	TXD	RXD
LIN wake-up (LIN pin)	Low	Low
$V_{S_{th}}$ (battery) undervoltage detection ( $V_S < 3.9V$ )	High	Low

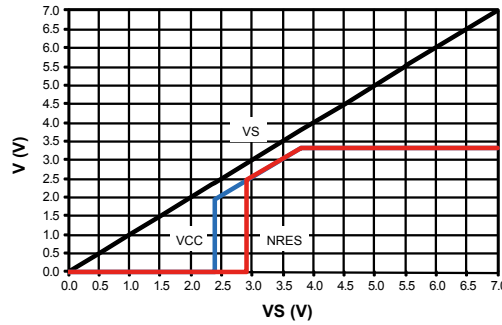
### 3.4.4 Behavior under Low Supply Voltage Condition

After the battery voltage has been connected to the application circuit, the voltage at the VS pin increases according to the block capacitor used in the application. If  $V_{VS}$  is higher than the minimum VS operation threshold  $V_{VS_{th\_U\_F\_up}}$ , the IC mode changes from unpowered mode to fail-safe mode. As soon as  $V_{VS}$  exceeds the undervoltage threshold  $V_{VS_{th\_F\_N\_up}}$ , the LIN transceiver can be activated.

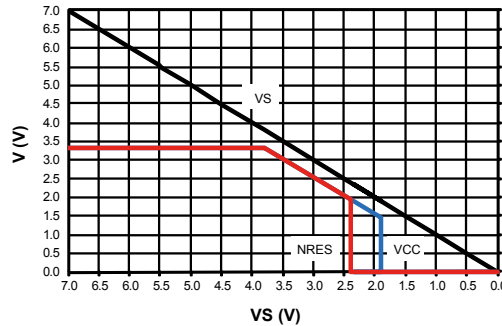
The VCC output voltage reaches its nominal value after  $t_{VCC}$ . This parameter depends on the externally applied VCC capacitor and the load. The NRES output is low for the reset time delay  $t_{reset}$ . No mode change is possible during this time  $t_{reset}$ .

The behavior of VCC, NRES and VS is shown in the following diagrams (ramp-up and ramp-down):

**Figure 3-7. VCC and NRES versus VS (Ramp-up) for 3.3V**



**Figure 3-8. VCC and NRES versus VS (Ramp-down) for 3.3V**



Please note that the upper graphs are only valid if the VS ramp-up and ramp-down times are much slower than the VCC ramp-up time  $t_{VCC}$  and the NRES delay time  $t_{reset}$ .

If during sleep mode the voltage level of  $V_{VS}$  drops below the undervoltage detection threshold  $V_{VS\_th\_N\_F\_down}$  (typ. 4.3V), the operation mode is not changed and no wake-up is possible. Only if the supply voltage on pin VS drops below the VS operation threshold  $V_{VS\_th\_U\_down}$  (typ. 2.05V), does the IC switch to unpowered mode.

If during silent mode the VCC voltage drops below the VCC undervoltage threshold  $V_{VCC\_th\_uv\_down}$  the IC switches into fail-safe mode. If the supply voltage on pin VS drops below the VS operation threshold  $V_{VS\_th\_U\_down}$  (typ. 2.05V), does the IC switch to unpowered mode.

If during normal mode the voltage level on the VS pin drops below the VS undervoltage detection threshold  $V_{VS\_th\_N\_F\_down}$  (typ. 4.3V), the IC switches to fail-safe mode. This means the LIN transceiver is disabled in order to avoid malfunctions or false bus messages. The voltage regulator remains active.

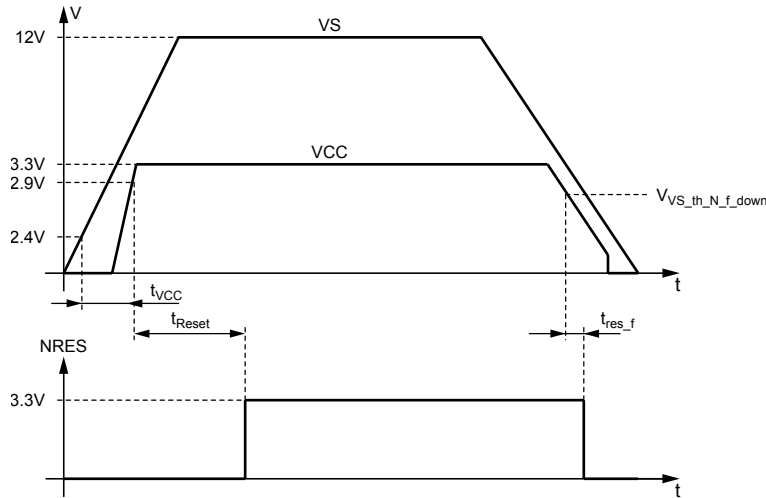
In this undervoltage situation it is possible to switch the device into sleep mode or silent mode by a falling edge at the EN input. For this feature, switching into these two current saving modes is always guaranteed, allowing current consumption to be reduced even further. When the VCC voltage drops below the VCC undervoltage threshold  $V_{VCC\_th\_uv\_down}$  (typ. 2.6V) the IC switches into fail-safe mode.

The current consumption of the SBC in silent mode or in fail-safe mode and the voltage regulator is always below 170 $\mu$ A, even when the supply voltage VS is lower than the regulator's nominal output voltage VCC.



3.4.5 Voltage Regulator

Figure 3-9. Voltage Regulator: Supply Voltage Ramp-Up and Ramp-Down



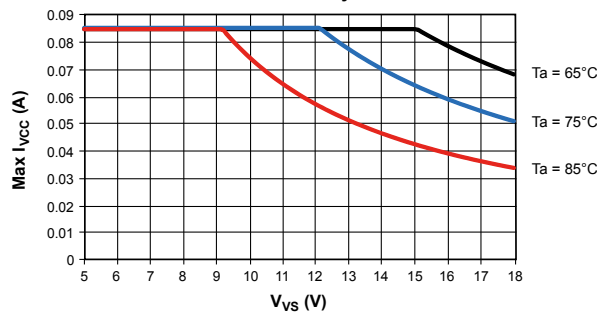
The voltage regulator needs an external capacitor for compensation and to smooth the disturbances from the microcontroller. It is recommended to use an MLC capacitor with a minimum capacitance of 1.8µF together with a 100nF ceramic capacitor. Depending on the application, the values of these capacitors can be modified by the customer.

During a short circuit at VCC, the output limits the output current to  $I_{VCClim}$ . Because of undervoltage, NRES switches to low and sends a reset to the microcontroller. If the chip temperature exceeds the value  $T_{VCCoff}$ , the VCC output switches off. The chip cools down and, after a hysteresis of  $T_{hys}$ , switches the output on again.

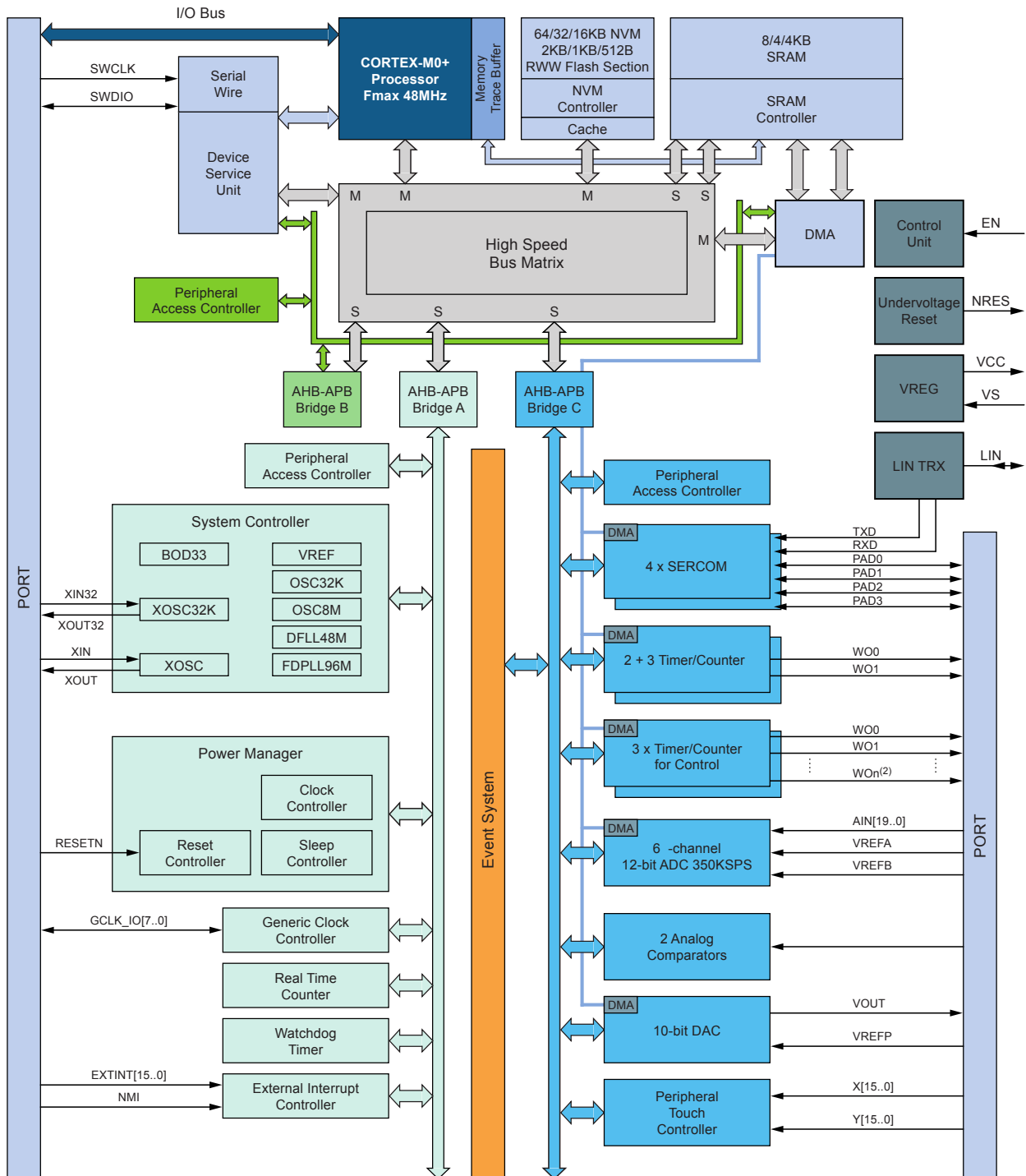
When the IC is being soldered onto the PCB it is mandatory to connect the heat slug with a wide GND plate on the printed board to get a good heat sink.

The main power dissipation of the IC is created from the VCC output current  $I_{VCC}$ , which is needed for the application. “Power Dissipation: Safe Operating Area: Regulator’s Output Current  $I_{VCC}$  versus Supply Voltage  $V_S$ ” is shown in the figure below.

Figure 3-10. Power Dissipation: Safe Operating Area: Regulator’s Output Current  $I_{VCC}$  versus Supply Voltage  $V_S$  at Different Ambient Temperatures (32-pin version,  $R_{thja} = 64K/W$  assumed)



### 4. Block Diagram



1. Some products have different number of SERCOM instances, Timer/Counter instances, PTC signals and ADC signals. Refer to the Configuration Summary for details.
2. The three TCC instances have different configurations, including the number of Waveform Output (WO) lines. Refer to the TCC Configuration for details.

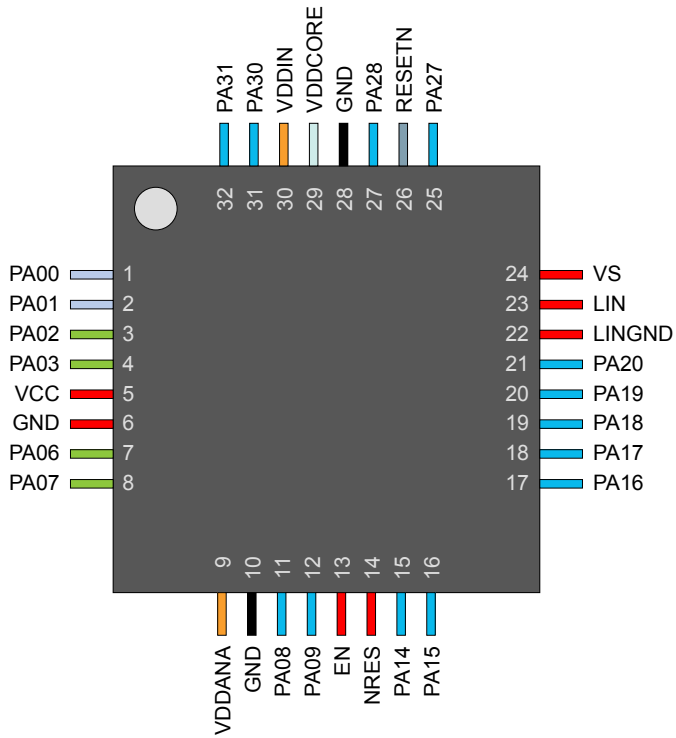
### Related Links

[Configuration Summary](#)

[TCC Configurations](#)

### 5. Pinout

#### 5.1 SAM HA1E - VQFN32



## 6. Signal Descriptions List

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

Signal Name	Function	Type	Active Level
<b>Analog Comparators - AC</b>			
AIN[3:0]	AC Analog Inputs	Analog	
CMP[:0]	AC Comparator Outputs	Digital	
<b>Analog Digital Converter - ADC</b>			
AIN[19:0]	ADC Analog Inputs	Analog	
VREFA	ADC Voltage External Reference A	Analog	
VREFB	ADC Voltage External Reference B	Analog	
<b>Digital Analog Converter - DAC</b>			
VOUT	DAC Voltage output	Analog	
VREFA	DAC Voltage External Reference	Analog	
<b>External Interrupt Controller</b>			
EXTINT[15:0]	External Interrupts	Input	
NMI	External Non-Maskable Interrupt	Input	
<b>Generic Clock Generator - GCLK</b>			
GCLK_IO[7:0]	Generic Clock (source clock or generic clock generator output)	I/O	
<b>Power Manager - PM</b>			
RESETN	Reset	Input	Low
<b>Serial Communication Interface - SERCOMx</b>			
PAD[3:0]	SERCOM I/O Pads	I/O	
<b>System Control - SYSCTRL</b>			
XIN	Crystal Input	Analog/ Digital	
XIN32	32kHz Crystal Input	Analog/ Digital	
XOUT	Crystal Output	Analog	
XOUT32	32kHz Crystal Output	Analog	
<b>Timer Counter - TCx</b>			
WO[1:0]	Waveform Outputs	Output	
<b>Timer Counter - TCCx</b>			
WO[1:0]	Waveform Outputs	Output	

# ATSAMHAXEXXA

## Signal Descriptions List

Signal Name	Function	Type	Active Level
Peripheral Touch Controller - PTC			
<b>Note:</b> Is not available on SAMHA0 devices			
X[15:0]	PTC Input	Analog	
Y[15:0]	PTC Input	Analog	
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	

## 7. I/O Multiplexing and Considerations

### 7.1 Multiplexed Signals

By default each pin is controlled by the PORT as a general purpose I/O and alternatively can be assigned to one of the peripheral functions A, B, C, D, E, F, 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 “1”. 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.

This table describes the peripheral signals multiplexed to the PORT I/O pins.

**Table 7-1. PORT Function Multiplexing**

Pin	I/O Pin	Supply	Type	A		B				C	D	E	F	G	H
				EIC	REF	ADC	AC	PTC	DAC	SERCOM	SERCOM-ALT	TC /TCC	TCC	COM	AC/ GCLK
1	PA00	VDDANA		E00							SERCOM 1/ PAD[0]	TCC2/WO[0]			
2	PA01	VDDANA		E01							SERCOM 1/ PAD[1]	TCC2/WO[1]			
3	PA02	VDDANA		E02		AIN[0]		Y[0]	VOUT						
4	PA03	VDDANA		E03	ADC/ VREFA DAC/ VREFP	AIN[1]		Y[1]							
7	PA06	VDDANA		E06		AIN[2]	AIN[2]	Y[4]			SERCOM 0/PAD[2]	TCC0/WO[0]			
8	PA07	VDDANA		E07		AIN[7]	AIN[3]	Y[5]			SERCOM 0/PAD[3]	TCC0/WO[1]			
11	PA08	VDDIO	i <sup>2</sup> C	NMI		AIN[6]		X[0]		SERCOM 0/PAD[0]	SERCOM 2/ PAD[0]	TCC0/WO[0]	TCC1/ WO[2]		
12	PA09	VDDIO	i <sup>2</sup> C	E09		AIN[17]		X[1]		SERCOM 0/PAD[1]	SERCOM 2/ PAD[1]	TCC0/WO[1]	TCC1/ WO[3]		
15	PA14	VDDIO		E14						SERCOM 2/ PAD[2]	SERCOM 4/PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[ 0]
16	PA15	VDDIO		E15						SERCOM 2/ PAD[3]	SERCOM 4/PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[ 1]
17	PA16	VDDIO	i <sup>2</sup> C	E00				X[4]		SERCOM 1/ PAD[0]	SERCOM 3/ PAD[0]	TC2/WO[0]	TCC0/ WO[6]		GCLK_IO[ 2]
18	PA17	VDDIO	i <sup>2</sup> C	E01				X[5]		SERCOM 1/ PAD[1]	SERCOM 3/ PAD[1]	TC5/WO[1]	TCC0/ WO[7]		GCLK_IO[ 3]
19	PA18	VDDIO		E02				X[6]		SERCOM 1/ PAD[2]	SERCOM 3/ PAD[0]	TCC3/WO[0]	TCC0/ WO[2]		AC/ CMP[0]
20	PA19	VDDIO		E03				X[7]		SERCOM 1/ PAD[3]	SERCOM 3/ PAD[3]	TCC3/WO[1]	TCC0/ WO[3]		AC/ CMP[1]
21	PA20	VDDIO		E04				X[8]		SERCOM 5/ PAD[2]	SERCOM 3/ PAD[0]	TC7/WO[0]	TCC0/ WO[6]		GCLK_IO[ 4]
25	PA27	VDDIO		E15						SERCOM 1/PAD[0]		TC5/WO[0]			GCLK_IO[ 0]
27	PA28	VDDIO		E08						SERCOM 1/PAD[1]		TC5/WO[1]			GCLK_IO[ 0]

# ATSAMHAXEXXA

## I/O Multiplexing and Considerations

Pin	I/O Pin	Supply	Type	A		B				C	D	E	F	G	H
				EIC	REF	ADC	AC	PTC	DAC	SERCOM	SERCOM-ALT	TC /TCC	TCC	COM	AC/GCLK
31	PA30	VDDIO		E10							SERCOM 1/ PAD[2]	TCC1/WO[0]		SWD CLK	GCLK_IO[0]
32	PA31	VDDIO		E11							SERCOM 1/ PAD[3]	TCC1/WO[1]		SWDIO	

**Note:** 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 (< 1 mA) due to a not stable level on pad.

**Table 7-2. Internal Connections 32-pin**

SBC Pin	Microcontroller Pin	MUX Function
TXD	PA13	SERCOM[4]-PAD[2], Mux: C
RXD	PB11	SERCOM[4]-PAD[3], Mux: D

## 7.2 Other Functions

### 7.2.1 Oscillator Pinout

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

**Table 7-3. Oscillator Pinout**

Oscillator	Supply	Signal	I/O pin
XOSC	VDDIO	XIN	PA14
		XOUT	PA15
XOSC32K	VDDANA	XIN32	PA00
		XOUT32	PA01

### 7.2.2 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.

**Table 7-4. Serial Wire Debug Interface Pinout**

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

### 7.2.3 SERCOM I<sup>2</sup>C Pins

**Table 7-5. SERCOM Pins Supporting I<sup>2</sup>C**

Device	Pins Supporting I <sup>2</sup> C Hs mode
SAMHA0/1E	PA08, PA09, PA16, PA17



### 7.2.4 GPIO Clusters

**Table 7-6. GPIO Clusters**

PACKAGE	CLUSTER	GPIO															SUPPLIES PINS CONNECTED TO THE CLUSTER	
		PA31	PA30															
32pins	1	PA31	PA30															VDDIN pin30/GND pin 28
	2	PA28	PA27	PA25	PA24	PA23	PA22	PA19	PA18	PA17	PA16	PA15	PA14	PA11	PA10	PA09	PA08	VDDIN pin30/GND pin 28 and VDDANA pin9/GND pin10
	3	PA07	PA06	PA05	PA04	PA03	PA02	PA01	PA00									VDDANA pin9/GND pin10

### 7.2.5 TCC Configurations

The SAMHA0/1 has three instances of the Timer/Counter for Control applications (TCC) peripheral, , TCC[2:0]. The following table lists the features for each TCC instance.

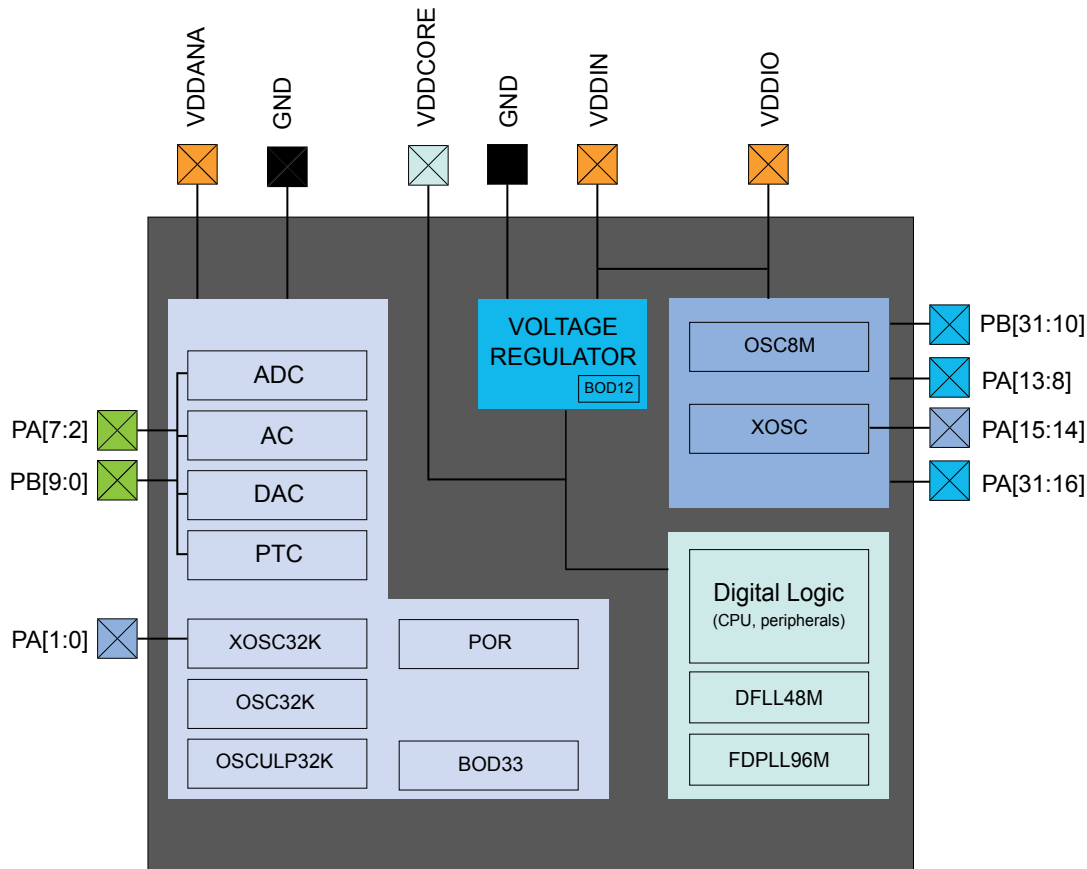
**Table 7-7. TCC Configuration Summary**

TCC#	Channels (CC_NUM)	Waveform Output (WO_NUM)	Counter size	Fault	Dithering	Output matrix	Dead Time Insertion (DTI)	SWAP	Pattern generation
0	4	8	24-bit	Yes	Yes	Yes	Yes	Yes	Yes
1	2	4	24-bit	Yes	Yes				Yes
2	2	2	16-bit	Yes					

**Note:** The number of CC registers (CC\_NUM) for each TCC corresponds to the number of compare/capture channels, so that a TCC can have more Waveform Outputs (WO\_NUM) than CC registers.

## 8. Power Supply and Start-Up Considerations

### 8.1 Power Domain Overview



## 8.2 Power Supply Considerations

### 8.2.1 Power Supplies

The device has several different power supply pins:

- VDDIO: Powers I/O lines, OSC8M and XOSC. Voltage is 2.7V to 3.63V.
- VDDIN: Powers I/O lines and the internal regulator. Voltage is 2.7V to 3.63V.
- VDDANA: Powers I/O lines and the ADC, AC, DAC, PTC, OSCULP32K, OSC32K, XOSC32K. Voltage is 2.7V to 3.63V.
- VDDCORE: Internal regulated voltage output. Powers the core, memories, peripherals, FDPLL96M, and DFLL48M. Voltage is 1.2V.

The same voltage must be applied to both VDDIN, VDDIO and VDDANA. This common voltage is referred to as  $V_{DD}$  in the datasheet.

The ground pins, GND, are common to VDDANA VDDCORE, VDDIO and VDDIN.

For decoupling recommendations for the different power supplies. Refer to *Schematic Checklist* for details.

### 8.2.2 Voltage Regulator

The voltage regulator has two different modes:

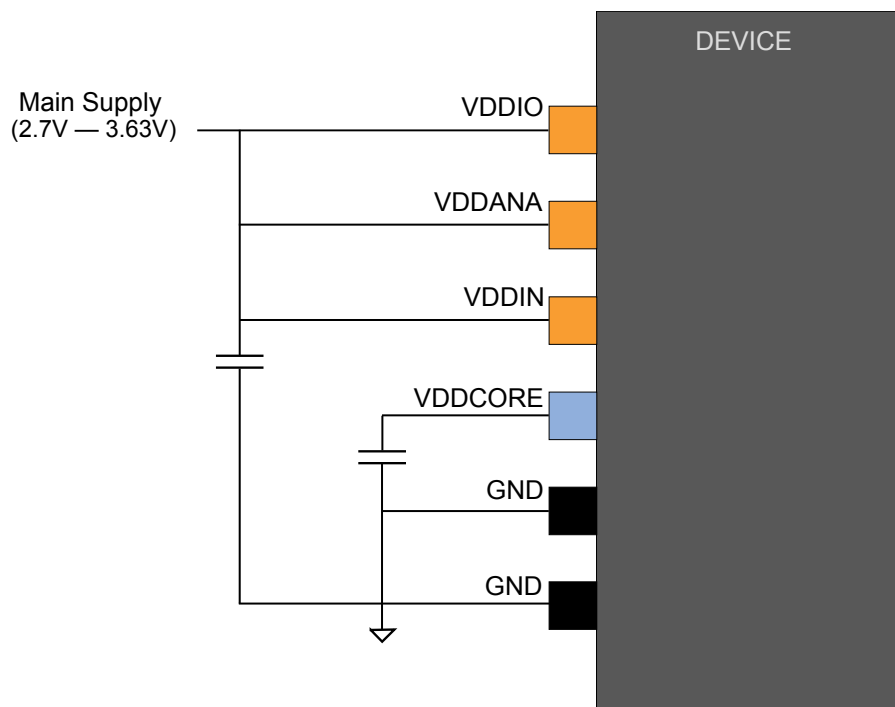
- Normal mode: To be used when the CPU and peripherals are running
- Low Power (LP) mode: To be used when the regulator draws small static current. It can be used in standby mode

### 8.2.3 Typical Powering Schematics

The device uses a single main supply with a range of 2.7V - 3.63V.

The following figure shows the recommended power supply connection.

**Figure 8-1. Power Supply Connection**



### 8.2.4 Power-Up Sequence

#### 8.2.4.1 Minimum Rise Rate

The integrated Power-on Reset (POR) circuitry monitoring the VDDANA power supply requires a minimum rise rate. Refer to the *Electrical Characteristics* for details.

#### Related Links

[Electrical Characteristics](#)

#### 8.2.4.2 Maximum Rise Rate

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

#### Related Links

[Electrical Characteristics](#)

### 8.3 Power-Up

This section summarizes the power-up sequence of the device. The behavior after power-up is controlled by the Power Manager. Refer to *PM – Power Manager* for details.

**Related Links**

[PM – Power Manager](#)

#### 8.3.1 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* 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).

**Related Links**

[PM – Power Manager](#)

#### 8.3.2 I/O Pins

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

#### 8.3.3 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*, *GCLK – Generic Clock Controller* and *SYSCTRL – System Controller* for details. Refer to the ARM Architecture Reference Manual for more information on CPU startup (<http://www.arm.com>).

**Related Links**

[PM – Power Manager](#)

[SYSCTRL – System Controller](#)

[Clock System](#)

### 8.4 Power-On Reset and Brown-Out Detector

The SAMHA0/1 embeds three features to monitor, warn and/or reset the device:

- POR: Power-On Reset on VDDANA
- BOD33: Brown-Out Detector on VDDANA
- 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.

### 8.4.1 Power-On Reset on VDDANA

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

### 8.4.2 Brown-Out Detector on VDDANA

BOD33 monitors VDDANA. Refer to *SYSCTRL – System Controller* for details.

#### Related Links

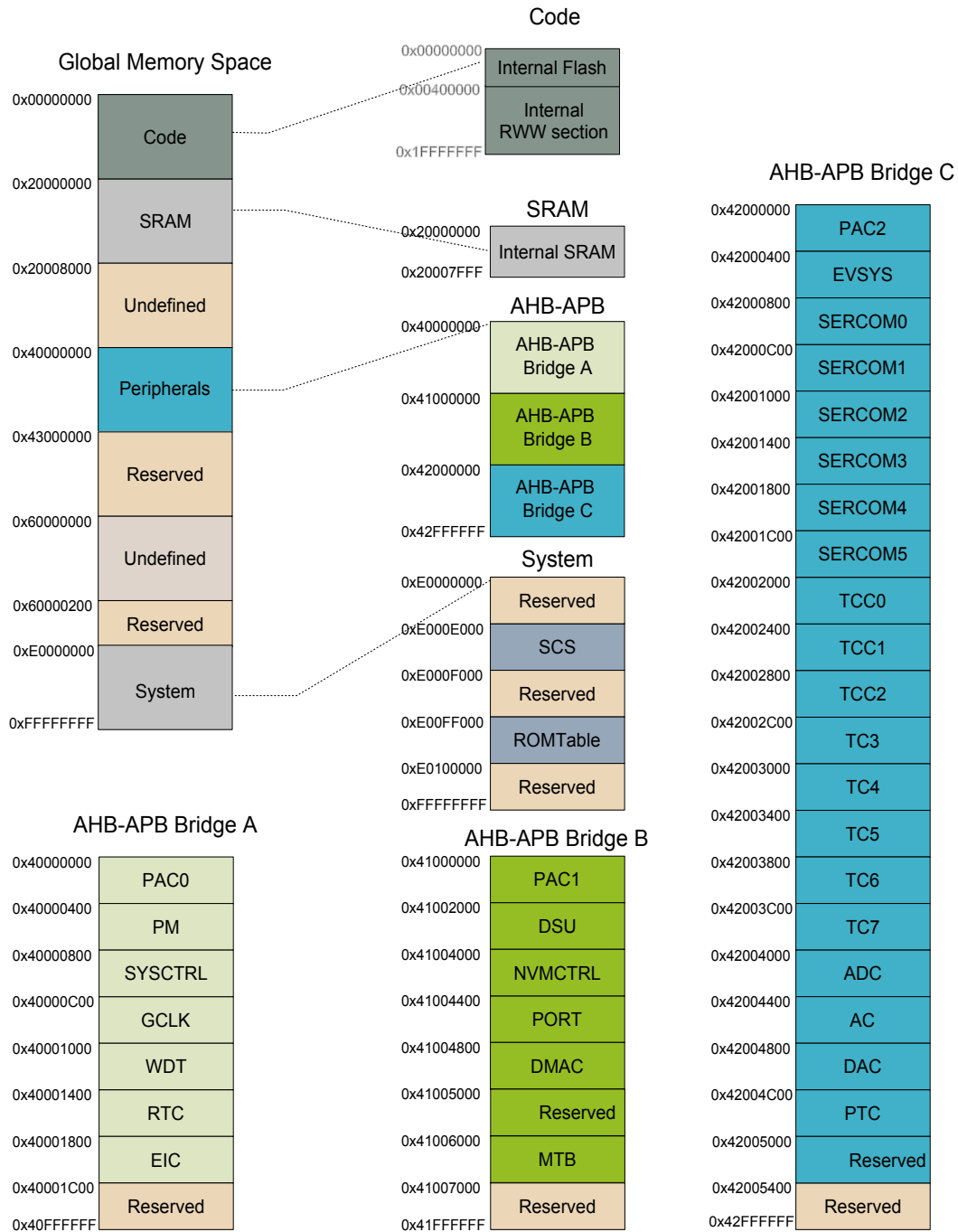
[SYSCTRL – System Controller](#)

### 8.4.3 Brown-Out Detector on VDDCORE

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

## 9. Product Mapping

Figure 9-1. SAMHA0/1 Product Mapping



This figure represents the full configuration of the SAMHA0/1 with maximum flash and SRAM capabilities and a full set of peripherals.

## 10. Automotive Quality Grade

The SAMHA0/1 has been developed and manufactured according to the most stringent requirements of the international standard ISO-TS 16949. This data sheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the SAMHA0/1 have been verified during regular product qualification as per AEC-Q100.

As indicated in the ordering information paragraph, the product is available in only one temperature grade. Refer to the table below.

**Table 10-1. Temperature Grade Identification for Automotive Products**

Temperature	Temperature Identifier	Comments
-40°C to +105°C T <sub>C</sub>	B	Full automotive temperature range.
-40°C to +115°C T <sub>C</sub>	Z	Full automotive temperature range.

### Related Links

[Product Identification System](#)

**11. Data Retention**

Reliability qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 105°C or 100 years at 25°C.



## 12. Memories

### 12.1 Embedded Memories

- Internal high-speed flash with Read-While-Write (RWW) capability on section of the array
- Internal high-speed RAM, single-cycle access at full speed

### 12.2 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 follows:

**Table 12-1. Physical memory map**

Memory	Start address	Size		
		SAMHA1E16	SAMHA1E15	SAMHA1E14
Internal Flash	0x00000000	64Kbytes	32Kbytes	16Kbytes
Internal RWW section	0x00400000	2Kbytes	1Kbytes	512bytes
Internal SRAM	0x20000000	8Kbytes	4Kbytes	4Kbytes
Peripheral Bridge A	0x40000000	64Kbytes	64Kbytes	64Kbytes
Peripheral Bridge B	0x41000000	64Kbytes	64Kbytes	64Kbytes
Peripheral Bridge C	0x42000000	64Kbytes	64Kbytes	64Kbytes

**Table 12-2. Flash memory parameters**

Device	Flash size	Number of pages	Page size
SAMHA1x16	64Kbytes	1024	64 bytes
SAMHA1x15	32Kbytes	512	64 bytes
SAMHA1x14	16Kbytes	256	64 bytes

2. The number of pages (NVMP) and page size (PSZ) can be read from the NVM Pages and Page Size bits in the NVM Parameter register in the NVMCTRL (PARAM.NVMP and PARAM.PSZ, respectively). Refer to *NVM Parameter* (PARAM) register for details.

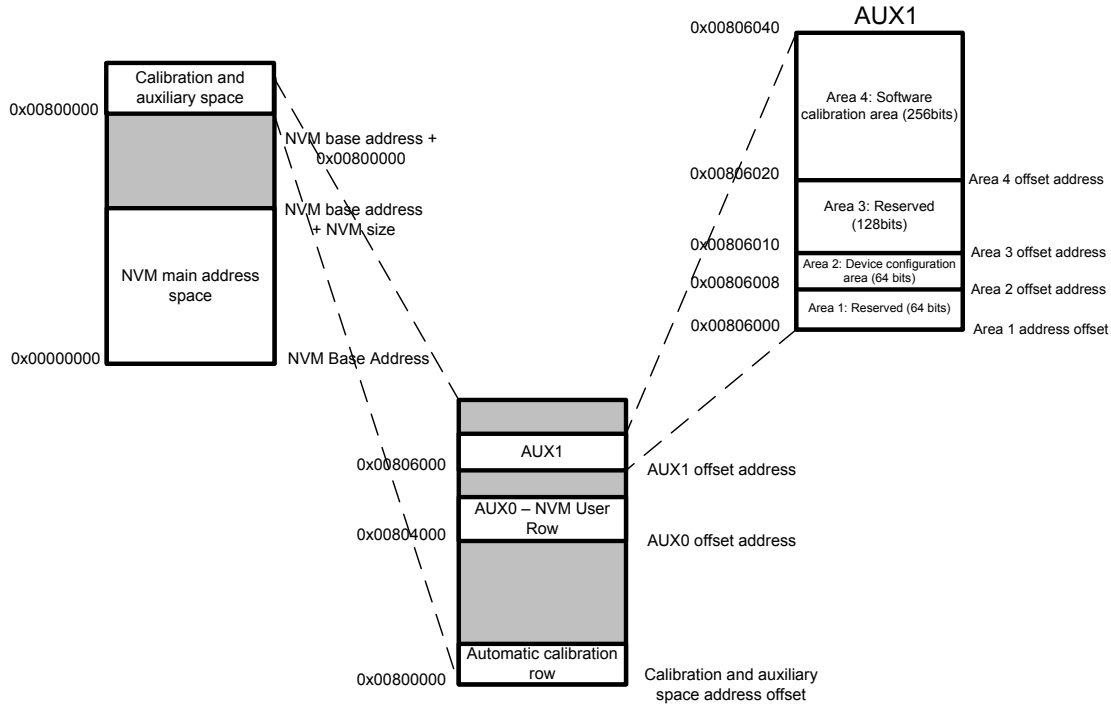
**Table 12-3. RWW section parameters**

Device	Flash size	Number of pages	Page size
SAMHA1x16	2Kbytes	32	64 bytes
SAMHA1x15	1Kbytes	16	64 bytes
SAMHA1x14	512 bytes	8	64 bytes

### 12.3 NVM Calibration and Auxiliary Space

The device calibration data are stored in different sections of the NVM calibration and auxiliary space presented in Figure. Calibration and Auxiliary Space.

**Figure 12-1. Calibration and Auxiliary Space**



The values from the automatic calibration row are loaded into their respective registers at startup.

#### 12.3.1 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*.

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 12-4. NVM User Row Mapping**

Bit Position	Name	Usage
2:0	BOOTPROT	Used to select one of eight different bootloader sizes. Refer to <i>NVMCTRL – Non-Volatile Memory Controller</i> . Default value = 7 except for WLCSP (Default value = 3).
3	Reserved	
6:4	EEPROM	Used to select one of eight different EEPROM sizes. Refer to <i>NVMCTRL – Non-Volatile Memory Controller</i> . Default value = 7.
7	Reserved	

Bit Position	Name	Usage
13:8	BOD33 Level	BOD33 Threshold Level at power on. Refer to <i>SYSCTRL BOD33 register</i> . Default value = 0x22.
14	BOD33 Enable	BOD33 Enable at power on . Refer to <i>SYSCTRL BOD33 register</i> . Default value = 1.
16:15	BOD33 Action	BOD33 Action at power on. Refer to <i>SYSCTRL BOD33 register</i> . Default value = 1.
24:17	Reserved	Voltage Regulator Internal BOD (BOD12) configuration. These bits are written in production and must not be changed. Default value = 0x70.
25	WDT Enable	WDT Enable at power on. Refer to <i>WDT CTRL register</i> . Default value = 0.
26	WDT Always-On	WDT Always-On at power on. Refer to <i>WDT CTRL register</i> . Default value = 0.
30:27	WDT Period	WDT Period at power on. Refer to <i>WDT CONFIG register</i> . Default value = 0x0B.
34:31	WDT Window	WDT Window mode time-out at power on. Refer to <i>WDT CONFIG register</i> . Default value = 0x05.
38:35	WDT EWOFFSET	WDT Early Warning Interrupt Time Offset at power on. Refer to <i>WDT EWCTRL register</i> . Default value = 0x0B.
39	WDT WEN	WDT Timer Window Mode Enable at power on. Refer to <i>WDT CTRL register</i> . Default value = 0.
40	BOD33 Hysteresis	BOD33 Hysteresis configuration at power on. Refer to <i>SYSCTRL BOD33 register</i> . Default value = 0.
41	Reserved	Voltage Regulator Internal BOD(BOD12) configuration. This bit is written in production and must not be changed. Default value = 0.
47:42	Reserved	
63:48	LOCK	NVM Region Lock Bits. Refer to <i>NVMCTRL – Non-Volatile Memory Controller</i> . Default value = 0xFFFF.

### Related Links

[NVMCTRL – Nonvolatile Memory Controller](#)

[BOD33](#)

[CTRL](#)

### 12.3.2 NVM Software Calibration Area Mapping

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

The NVM Software Calibration Area can not be written.

**Table 12-5. NVM Software Calibration Area 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 ADC CALIB register.
37:35	ADC BIASCAL	ADC Bias Calibration. Should be written to ADC CALIB register.
44:38	OSC32K CAL	OSC32K Calibration. Should be written to SYCTRL OSC32K register.
49:45	Reserved	
54:50	Reserved	
57:55	Reserved	
63:58	DFLL48M COARSE CAL	DFLL48M Coarse calibration value. Should be written to SYCTRL DFLLVAL register.
73:64	Reserved	
127:74	Reserved	

### 12.3.3 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.

## 13. Processor And Architecture

### 13.1 Cortex M0+ Processor

The SAMHA0/1 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 <http://www.arm.com>.

#### 13.1.1 Cortex M0+ Configuration

**Table 13-1. Cortex M0+ Configuration**

Features	Configurable option	Device configuration
Interrupts	External interrupts 0-32	28
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 <sup>(1)</sup>
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 the following 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.

#### 13.1.2 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](http://www.arm.com)).
- System Timer (SysTick)

- The System Timer is a 24-bit timer clocked by CLK\_CPU that extends the functionality of both the processor and the NVIC. Refer to the Cortex-M0+ Technical Reference Manual for details ([www.arm.com](http://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](#) and the Cortex-M0+ Technical Reference Manual for details ([www.arm.com](http://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](http://www.arm.com)).
- Micro Trace Buffer (MTB)
  - The CoreSight MTB-M0+ (MTB) provides a simple execution trace capability to the Cortex-M0+ processor. Refer to section [Micro Trace Buffer](#) and the CoreSight MTB-M0+ Technical Reference Manual for details ([www.arm.com](http://www.arm.com)).

### 13.1.3 Cortex-M0+ Address Map

**Table 13-2. 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)
0x41006000 (see also Product Mapping)	Micro Trace Buffer (MTB)

### 13.1.4 I/O Interface

#### 13.1.4.1 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. Refer to *CPU Local Bus* for more information.

#### Related Links

[CPU Local Bus](#)

#### 13.1.4.2 Description

Direct access to PORT registers.

## 13.2 Nested Vector Interrupt Controller

### 13.2.1 Overview

The Nested Vectored Interrupt Controller (NVIC) in the SAMHA0/1 supports 32 interrupt lines with four different priority levels. For more details, refer to the Cortex-M0+ Technical Reference Manual ([www.arm.com](http://www.arm.com)).

### 13.2.2 Interrupt Line Mapping

Each of the 28 interrupt lines is connected to one peripheral instance, as shown in the table below. Each peripheral can have one or more interrupt flags, located in the peripheral's Interrupt Flag Status and Clear (INTFLAG) register. The interrupt flag is set when the interrupt condition occurs. Each interrupt in the peripheral can be individually enabled by writing a one to the corresponding bit in the peripheral's Interrupt Enable Set (INTENSET) register, and disabled by writing a one to the corresponding bit in the peripheral's Interrupt Enable Clear (INTENCLR) register. An interrupt request is generated from the peripheral when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt requests for one peripheral are ORed together on system level, generating one interrupt request for each peripheral. An interrupt request will set the corresponding interrupt pending bit in the NVIC interrupt pending registers (SETPEND/CLRPEND bits in ISPR/ICPR). For the NVIC to activate the interrupt, it must be enabled in the NVIC interrupt enable register (SETENA/CLRENA bits in ISER/ICER). The NVIC interrupt priority registers IPR0-IPR7 provide a priority field for each interrupt.

**Table 13-3. 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 Counter	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 Interface 0	9
SERCOM1 – Serial Communication Interface 1	10
SERCOM2 – Serial Communication Interface 2	11
SERCOM3 – Serial Communication Interface 3	12
SERCOM4 – Serial Communication Interface 4	13
SERCOM5 – Serial Communication Interface 5	14
TCC0 – Timer Counter for Control 0	15
TCC1 – Timer Counter for Control 1	16
TCC2 – Timer Counter for Control 2	17
TC3 – Timer Counter 3	18
TC4 – Timer Counter 4	19
TC5 – Timer Counter 5	20

Peripheral Source	NVIC Line
TC6 – Timer Counter 6	21
TC7 – Timer Counter 7	22
ADC – Analog-to-Digital Converter	23
AC – Analog Comparator	24
DAC – Digital-to-Analog Converter	25
PTC – Peripheral Touch Controller	26
Reserved	27

## 13.3 Micro Trace Buffer

### 13.3.1 Features

- Program flow tracing for the Cortex-M0+ processor
- MTB SRAM can be used for both trace and general purpose storage by the processor
- The position and size of the trace buffer in SRAM is configurable by software
- CoreSight compliant

### 13.3.2 Overview

When enabled, the MTB records changes in program flow, reported by the Cortex-M0+ processor over the execution trace interface shared between the Cortex-M0+ processor and the CoreSight MTB-M0+. This information is stored as trace packets in the SRAM by the MTB. An off-chip debugger can extract the trace information using the Debug Access Port to read the trace information from the SRAM. The debugger can then reconstruct the program flow from this information.

The MTB simultaneously stores trace information into the SRAM, and gives the processor access to the SRAM. The MTB ensures that trace write accesses have priority over processor accesses.

The execution trace packet consists of a pair of 32-bit words that the MTB generates when it detects the processor PC value changes non-sequentially. A non-sequential PC change can occur during branch instructions or during exception entry. See the CoreSight MTB-M0+ Technical Reference Manual for more details on the MTB execution trace packet format.

Tracing is enabled when the MASTER.EN bit in the Master Trace Control Register is 1. There are various ways to set the bit to 1 to start tracing, or to 0 to stop tracing. See the CoreSight Cortex-M0+ Technical Reference Manual for more details on the Trace start and stop and for a detailed description of the MTB's MASTER register. The MTB can be programmed to stop tracing automatically when the memory fills to a specified watermark level or to start or stop tracing by writing directly to the MASTER.EN bit. If the watermark mechanism is not being used and the trace buffer overflows, then the buffer wraps around overwriting previous trace packets.

The base address of the MTB registers is 0x41006000; this address is also written in the CoreSight ROM Table. The offset of each register from the base address is fixed and as defined by the CoreSight MTB-M0+ Technical Reference Manual. The MTB has 4 programmable registers to control the behavior of the trace features:

- POSITION: Contains the trace write pointer and the wrap bit,



- MASTER: Contains the main trace enable bit and other trace control fields,
- FLOW: Contains the WATERMARK address and the AUTOSTOP and AUTOHALT control bits,
- BASE: Indicates where the SRAM is located in the processor memory map. This register is provided to enable auto discovery of the MTB SRAM location, by a debug agent.

See the CoreSight MTB-M0+ Technical Reference Manual for a detailed description of these registers.

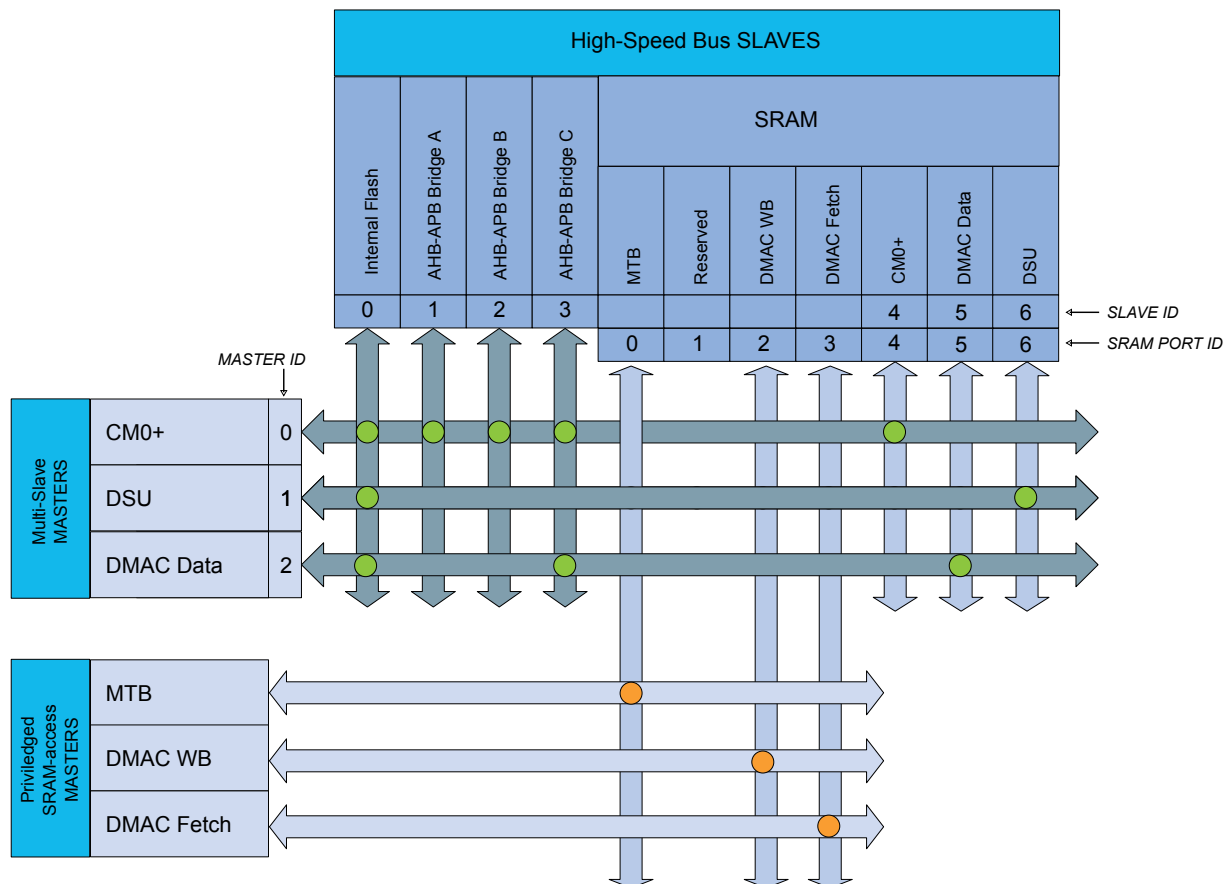
## 13.4 High-Speed Bus System

### 13.4.1 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 one-to-one clock frequency with the bus masters

### 13.4.2 Configuration



**Table 13-4. 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 13-5. 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 13-6. SRAM Port Connection**

SRAM Port Connection	Port ID	Connection Type
MTB - Micro Trace Buffer	0	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

### 13.4.3 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. Quality of Service.

**Table 13-7. Quality of Service**

Value	Name	Description
00	DISABLE	Background (no sensitive operation)
01	LOW	Sensitive Bandwidth

Value	Name	Description
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).

### 13.5 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.

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

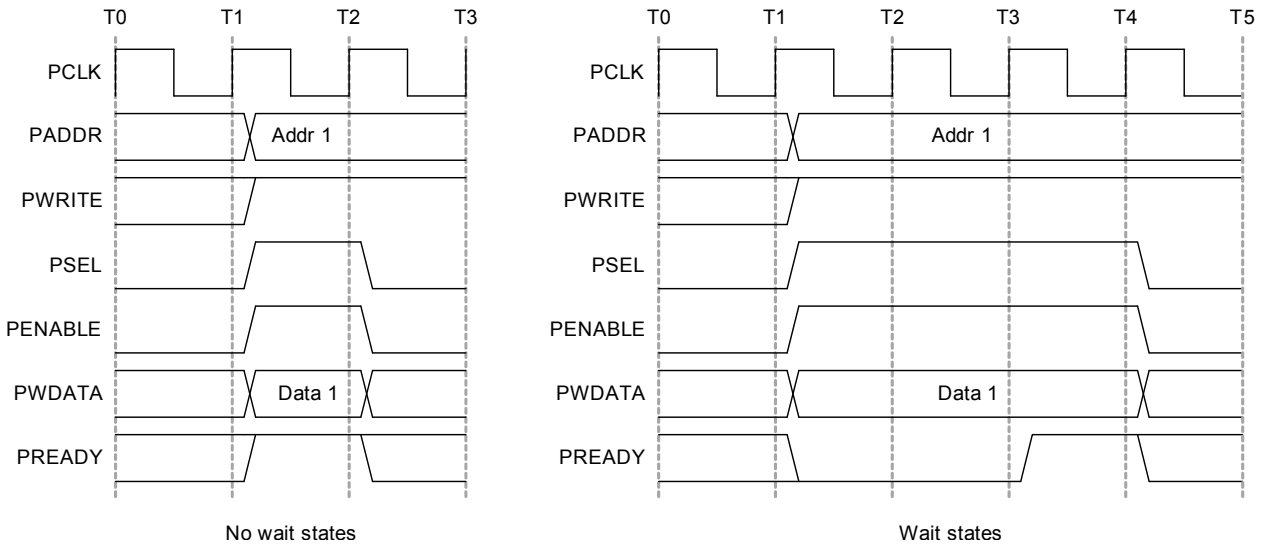
- Wait state support
- Error reporting
- Transaction protection
- Sparse data transfer (byte, half-word and word)

Additional enhancements:

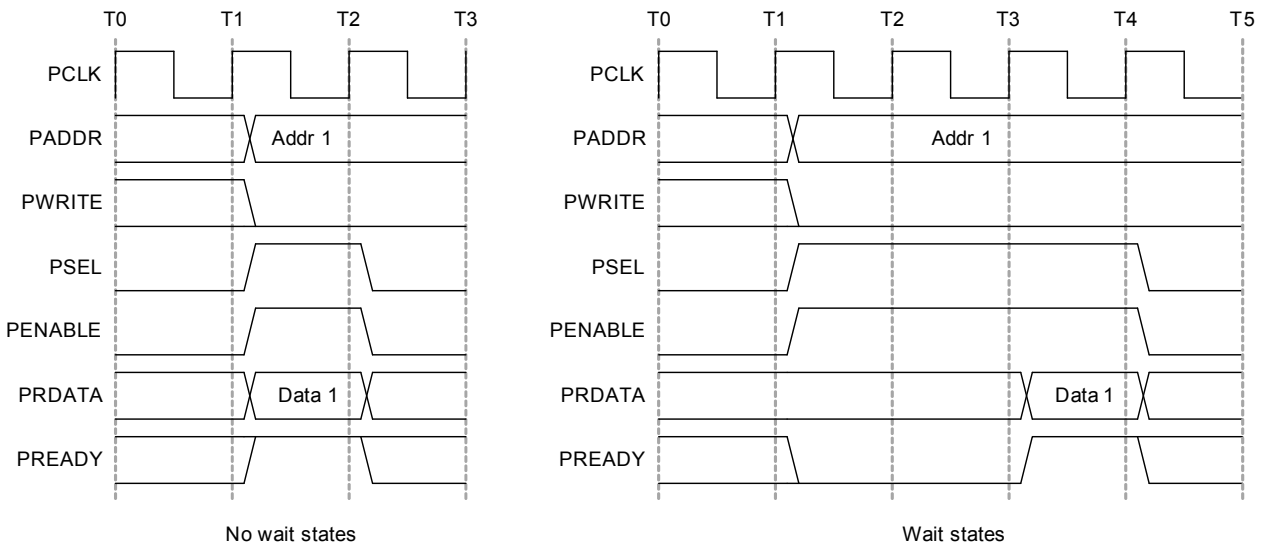
- Address and data cycles merged into a single cycle
- Sparse data transfer also apply to read access

to operate the AHB-APB bridge, the clock (CLK\_HPbX\_AHB) must be enabled. See *PM – Power Manager* for details.

**Figure 13-1. APB Write Access.**



**Figure 13-2. APB Read Access.**



**Related Links**

- [PM – Power Manager](#)
- [Product Mapping](#)

## 13.6 PAC - Peripheral Access Controller

### 13.6.1 Overview

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

The PAC peripheral bus clock (CLK\_PACx\_APB) can be enabled and disabled in the Power Manager. CLK\_PAC0\_APB and CLK\_PAC1\_APB are enabled at reset. CLK\_PAC2\_APB is disabled at reset. Refer to *PM – Power Manager* for details. 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, for example, 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.

### Related Links

[PM – Power Manager](#)

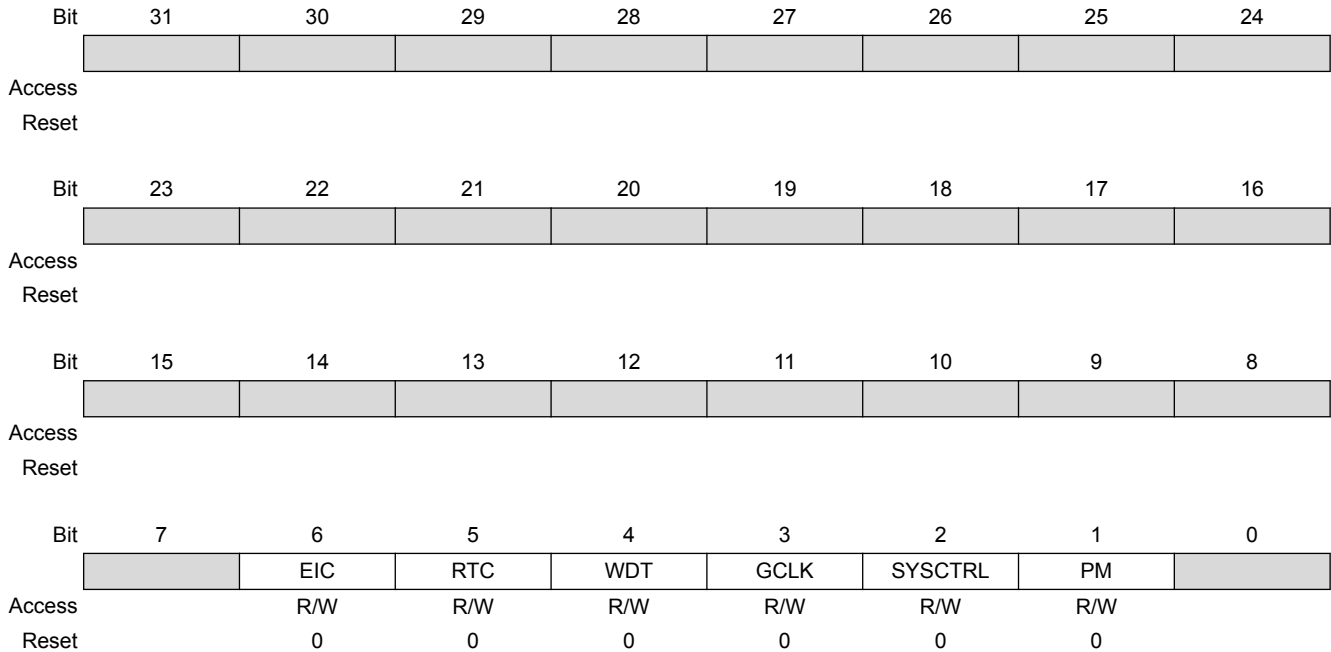
## 13.6.2 Register Description

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 the Product Mapping for PAC locations.

### 13.6.2.1 PAC0 Register Description

### Write Protect Clear

**Name:** WPCLR  
**Offset:** 0x00  
**Reset:** 0x000000  
**Property:** –



#### Bit 6 – EIC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 5 – RTC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 4 – WDT

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.

---

---

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 3 – GCLK**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 2 – SYSCTRL**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 1 – PM**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### Write Protect Set

**Name:** WPSET  
**Offset:** 0x04  
**Reset:** 0x000000  
**Property:** –

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
		EIC	RTC	WDT	GCLK	SYSCTRL	PM	
Access		R/W	R/W	R/W	R/W	R/W	R/W	
Reset		0	0	0	0	0	0	

#### Bit 6 – EIC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 5 – RTC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 4 – WDT

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.



---

---

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 3 – GCLK**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 2 – SYSCTRL**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 1 – PM**

Writing a zero to these bits has no effect.

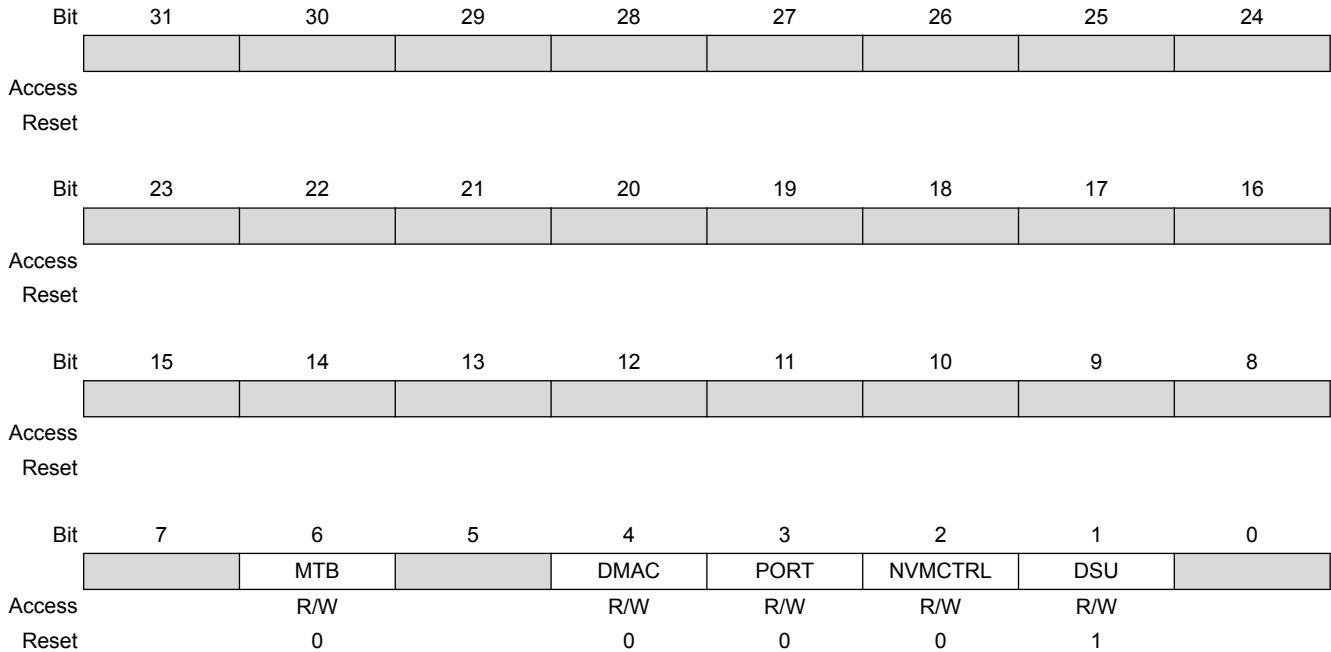
Writing a one to these bits will clear the Write Protect bit for the corresponding peripherals.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**13.6.2.2 PAC1 Register Description**

### Write Protect Clear

**Name:** WPCLR  
**Offset:** 0x00  
**Reset:** 0x000002  
**Property:** –



#### Bit 6 – MTB

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 4 – DMAC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 3 – PORT

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.

---

---

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 2 – NVMCTRL**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 1 – DSU**

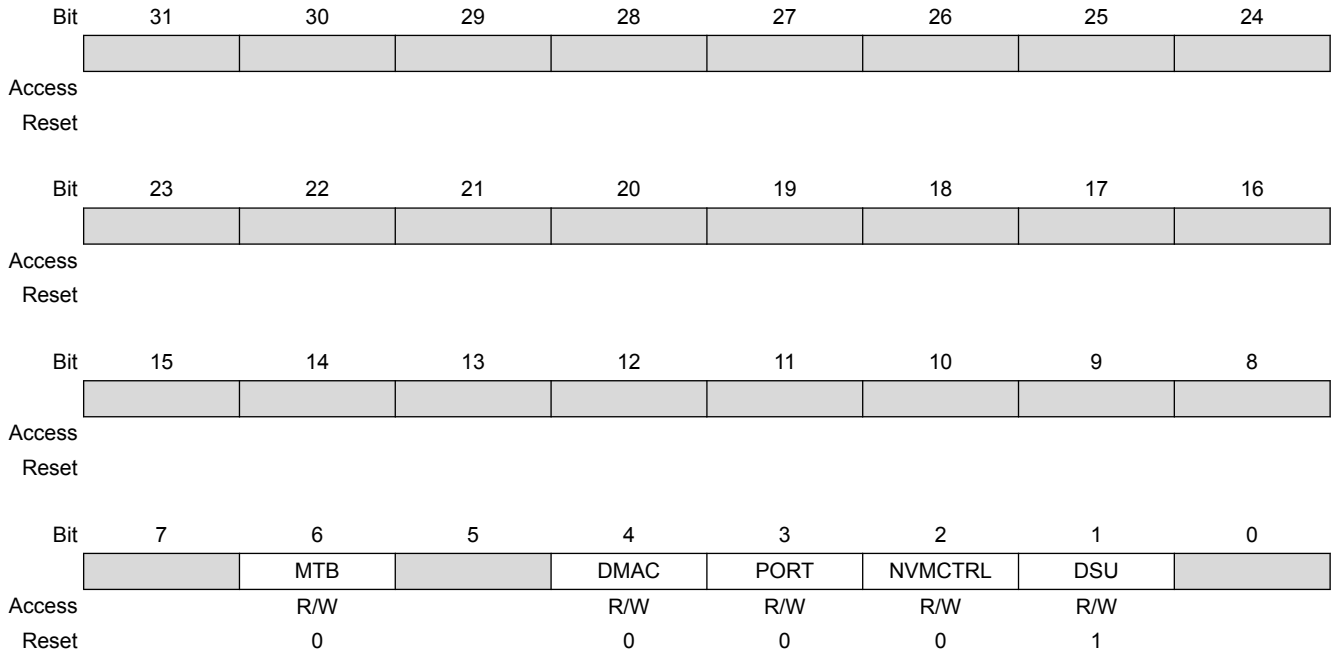
Writing a zero to these bits has no effect.

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

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### Write Protect Set

**Name:** WPSET  
**Offset:** 0x04  
**Reset:** 0x000002  
**Property:** –



#### Bit 6 – MTB

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 4 – DMAC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 3 – PORT

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.

---

---

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 2 – NVMCTRL**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 1 – DSU**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**13.6.2.3 PAC2 Register Description**

### Write Protect Clear

**Name:** WPCLR  
**Offset:** 0x00  
**Reset:** 0x00800000  
**Property:** –

Bit	31	30	29	28	27	26	25	24
	[Greyed out bits 31-24]							
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	[Greyed out bits 23-20]				PTC	DAC	AC	ADC
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	TC7	TC4	TC5	TC4	TC3	TCC2	TCC1	TCC0
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
	SERCOM[5:0]						EVSYS	[Greyed out bit 0]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

#### Bit 19 – PTC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 18 – DAC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 17 – AC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### Bit 16 – ADC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### Bits 11, 12, 13, 14, 15 – TC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### Bits 8, 9, 10 – TCC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### Bits 7:2 – SERCOM[5:0]

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### Bit 1 – EVSYS

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### Write Protect Set

**Name:** WPSET  
**Offset:** 0x04  
**Reset:** 0x00800000  
**Property:** –

	Bit	31	30	29	28	27	26	25	24
Access									
Reset									
	Bit	23	22	21	20	19	18	17	16
						PTC	DAC	AC	ADC
Access						R/W	R/W	R/W	R/W
Reset						0	0	0	0
	Bit	15	14	13	12	11	10	9	8
		TC7	TC6	TC5	TC4	TC3	TCC2	TCC1	TCC0
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
		SERCOM5	SERCOM4	SERCOM3	SERCOM2	SERCOM1	SERCOM0	EVSYS	
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset		0	0	0	0	0	0	0	

#### Bit 19 – PTC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 18 – DAC

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

#### Bit 17 – AC

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.



Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 16 – ADC**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bits 11, 12, 13, 14, 15 – TC**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bits 8, 9, 10 – TCC**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bits 2, 3, 4, 5, 6, 7 – SERCOM**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

**Bit 1 – EVSYS**

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.

Value	Description
0	Write-protection is disabled.
1	Write-protection is enabled.

### 14. Peripherals Configuration Summary

Table 14-1. Peripherals Configuration Summary

Periph. Name	Base Address	IRQ Line	AHB Clock		APB Clock		Generic Clock Index	PAC		Events		DMA Index	Sleep Walking
			Index	Enabled at Reset	Index	Enabled at Reset		Index	Prot. at Reset	User	Generator		
AHB-APB Bridge A	0x40000000	0	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	3	Y	1	Y		1	Y				
NVMCTRL	0x41004000	5	4	Y	2	Y		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	7-18: one per CHANNEL	1	N				Y
SERCOM0	0x42000800	9			2	N	20: CORE 19: SLOW	2	N			1: RX 2: TX	Y
SERCOM1	0x42000C00	10			3	N	21: CORE 19: SLOW	3	N			3: RX 4: TX	Y
SERCOM2	0x42001000	11			4	N	22: CORE 19: SLOW	4	N			5: RX 6: TX	Y
SERCOM3	0x42001400	12			5	N	23: CORE 19: SLOW	5	N			7: RX 8: TX	Y
SERCOM4	0x42001800	13			6	N	24: CORE 19: SLOW	6	N			9: RX 10: TX	Y
SERCOM5	0x42001C00	14			7	N	25: CORE 19: SLOW	7	N			11: RX 12: TX	Y
TCC0	0x42002000	15			8	N	26	8	N	4-5: EV0-1 6-9: MC0-3	34: OVF 35: TRG 36: CNT 37-40: MC0-3	13: OVF 14-17: MC0-3	Y
TCC1	0x42002400	16			9	N	26	9	N	10-11: EV0-1 12-13: MC0-1	41: OVF 42: TRG 43: CNT 44-45: MC0-1	18: OVF 19-20: MC0-1	Y
TCC2	0x42002800	17			10	N	27	10	N	14-15: EV0-1 16-17: MC0-1	46: OVF 47: TRG 48: CNT 49-50: MC0-1	21: OVF 22-23: MC0-1	Y
TC3	0x42002C00	18			11	N	27	11	N	18: EV	51: OVF 52-53: MC0-1	24: OVF 25-26: MC0-1	Y
TC4	0x42003000	19			12	N	28	12	N	19: EV	54: OVF 55-56: MCX0-1	27: OVF 28-29: MC0-1	Y

# ATSAMHAXEXXA

## Peripherals Configuration Summary

Periph. Name	Base Address	IRQ Line	AHB Clock		APB Clock		Generic Clock	PAC		Events		DMA	Sleep Walking
			Index	Enabled at Reset	Index	Enabled at Reset	Index	Index	Prot. at Reset	User	Generator	Index	
TC5	0x42003400	20			13	N	28	13	N	20: EV	57: OVF 58-59: MC0-1	30: OVF 31-32: MC0-1	Y
TC6	0x42003800	21			14	N	29	14	N	21: EV	60: OVF 61-62: MC0-1	33: OVF 34-35: MC0-1	Y
TC7	0x42003C00	22			15	N	29	15	N	22: EV	63: OVF 64-65: MC0-1	36: OVF 37-38: MC0-1	Y
ADC	0x42004000	23			16	Y	30	16	N	23: START 24: SYNC	66: RESRDY 67: WINMON	39: RESRDY	Y
AC	0x42004400	24			17	N	31: DIG  32: ANA	17	N	25-26: SOC0-1	68-69: COMP0-1  70: WIN0		Y
DAC	0x42004800	25			18	N	33	18	N	27: START	71: EMPTY	40: EMPTY	Y
PTC	0x42004C00	26			19	N	34	19	N	28: STCONV	72: EOC 73: WCOMP		

## **15. DSU - Device Service Unit**

### **15.1 Overview**

The Device Service Unit (DSU) provides a means of detecting debugger probes. It 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 within 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.

#### **Related Links**

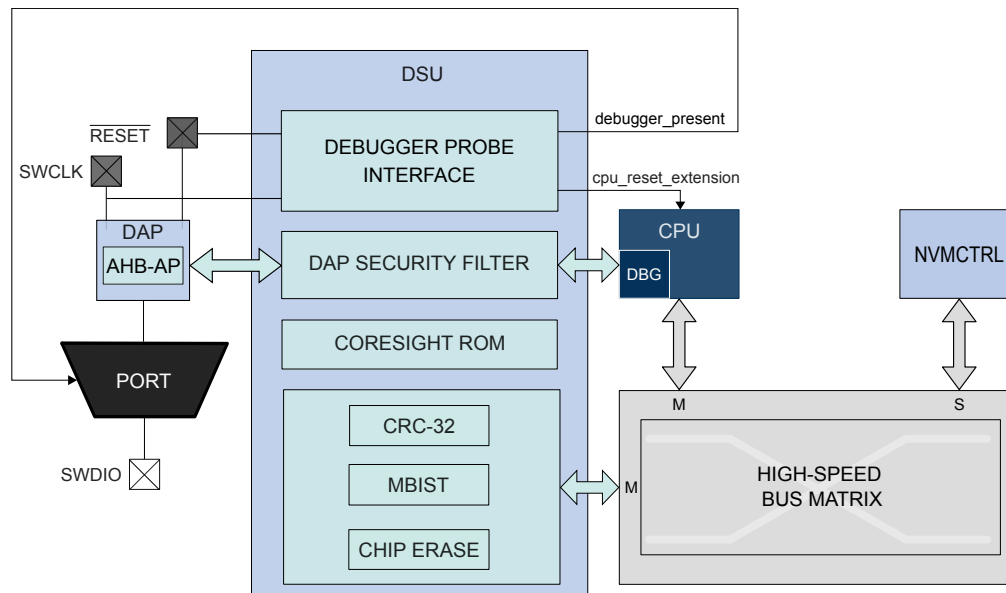
[System Services Availability when Accessed Externally and Device is Protected](#)  
[NVMCTRL – Nonvolatile Memory Controller](#)  
[Security Bit](#)

### **15.2 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)

## 15.3 Block Diagram

Figure 15-1. DSU Block Diagram



## 15.4 Signal Description

The DSU uses three signals to function.

Signal Name	Type	Description
RESET	Digital Input	External reset
SWCLK	Digital Input	SW clock
SWDIO	Digital I/O	SW bidirectional data pin

### Related Links

[I/O Multiplexing and Considerations](#)

## 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

The SWCLK pin is by default assigned to the DSU module to allow debugger probe detection and to stretch the CPU reset phase. For more information, refer to [Debugger Probe Detection](#). 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 is performed.

### 15.5.2 Power Management

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

**Related Links**

[PM – Power Manager](#)

**15.5.3 Clocks**

The DSU bus clocks (CLK\_DSU\_APB and CLK\_DSU\_AHB) can be enabled and disabled by the Power Manager. Refer to *PM – Power Manager*

**Related Links**

[PM – Power Manager](#)

**15.5.4 DMA**

Not applicable.

**15.5.5 Interrupts**

Not applicable.

**15.5.6 Events**

Not applicable.

**15.5.7 Register Access Protection**

Registers with write-access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

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

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

Write-protection does not apply for accesses through an external debugger.

**Related Links**

[PAC - Peripheral Access Controller](#)

**15.5.8 Analog Connections**

Not applicable.

**15.6 Debug Operation**

**15.6.1 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

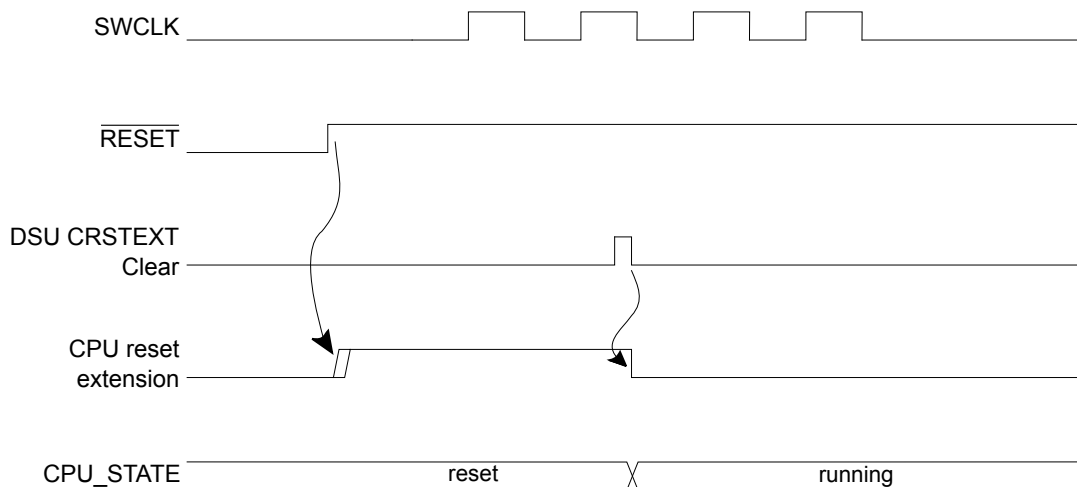
For more details on the ARM debug components, refer to the ARM Debug Interface v5 Architecture Specification.

**15.6.2 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 is connects to

the system. The debugger is detected on a  $\overline{\text{RESET}}$  release event when SWCLK is low. At startup, SWCLK is internally pulled up to avoid false detection of a debugger if the SWCLK pin is left unconnected. When the CPU is held in the reset extension phase, the CPU Reset Extension bit of the Status A register (STATUSA.CRSTEXT) is set. To release the CPU, write a '1' to STATUSA.CRSTEXT. STATUSA.CRSTEXT will then be set to '0'. Writing a '0' 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. Trying to do so sets the Protection Error bit (PERR) of the Status A register (STATUSA.PERR).

**Figure 15-2. Typical CPU Reset Extension Set and Clear Timing Diagram**



**Related Links**

[NVMCTRL – Nonvolatile Memory Controller Security Bit](#)

**15.6.3 Debugger Probe Detection**

**15.6.3.1 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.

**15.6.3.2 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  $\overline{\text{RESET}}$  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 15-3. 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.

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.

**Related Links**

[NVMCTRL – Nonvolatile Memory Controller Security Bit](#)

**15.7 Chip Erase**

Chip-Erase consists of removing all sensitive information stored in the chip and clearing the NVMCTRL security bit. Therefore, all volatile memories and the Flash memory (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 first 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 '1' 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](#)). The device then:
  - 1.1. Detects the debugger probe.
  - 1.2. Holds the CPU in reset.
2. Issue the Chip-Erase command by writing a '1' to CTRL.CE. The device then:
  - 2.1. Clears the system volatile memories.



- 2.2. Erases the whole Flash array (including the EEPROM emulation area, not including auxiliary rows).
- 2.3. Erases the lock row, removing the NVMCTRL security bit protection.
3. Check for completion by polling STATUSA.DONE (read as '1' when completed).
4. Reset the device to let the NVMCTRL update the fuses.

## 15.8 Programming

Programming the Flash or RAM memories is only possible when the device is not protected by the NVMCTRL security bit. The programming procedure is as follows:

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). 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.  $\overline{\text{RESET}}$  is released, resulting 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 the operation is completed, the chip can be restarted either by asserting  $\overline{\text{RESET}}$  or toggling power. Make sure that the SWCLK pin is high when releasing  $\overline{\text{RESET}}$  to prevent extending the CPU reset.

### Related Links

[Electrical Characteristics](#)

[NVMCTRL – Nonvolatile Memory Controller](#)

[Security Bit](#)

## 15.9 Intellectual Property Protection

Intellectual property protection consists of restricting access to internal memories from external tools when the device is protected, and this is accomplished by setting the NVMCTRL security bit. This protected state can be removed by issuing a Chip-Erase (refer to [Chip Erase](#)). When the device is protected, read/write accesses using the AHB-AP are limited to the DSU address range and DSU commands are restricted. When issuing a Chip-Erase, sensitive information is erased from volatile memory and Flash.

The DSU implements a security filter that monitors the AHB transactions 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

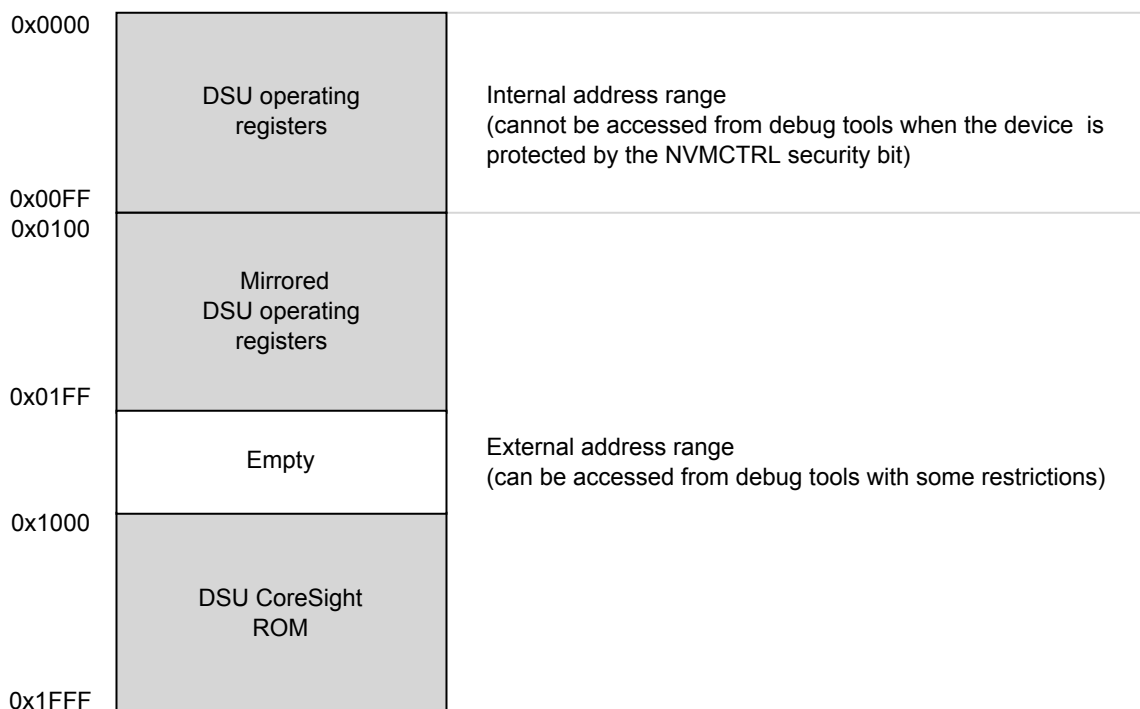
For security reasons, DSU features have limitations when used from a debug adapter. To differentiate external accesses from internal ones, the first 0x100 bytes of the DSU register map has been mirrored at offset 0x100:

- 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 range 0x0100-0x2000.

The DSU operating registers are located in the 0x0000-0x00FF area and remapped in 0x0100-0x01FF to differentiate accesses coming from a debugger and the CPU. If the device is protected and an access is issued in the region 0x0100-0x01FF, it is subject to security restrictions. For more information, refer to the [Table 15-1](#).

**Figure 15-4. APB Memory Mapping**



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

**Table 15-1. Feature Availability Under Protection**

Features	Availability when the device is protected
CPU Reset Extension	Yes
Clear CPU Reset Extension	No
Debugger Cold-Plugging	Yes
Debugger Hot-Plugging	No

### Related Links

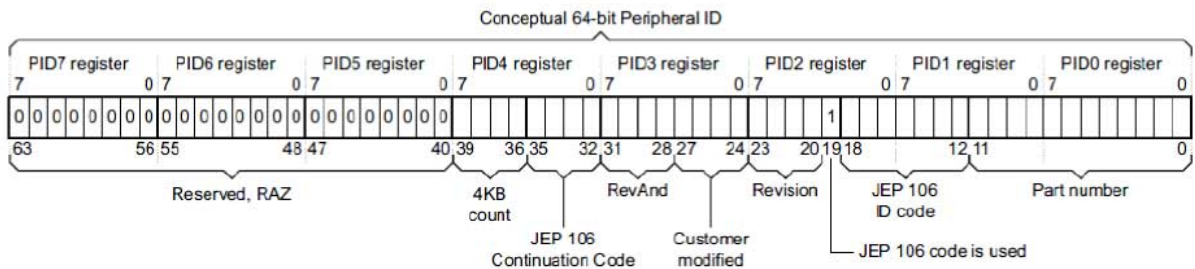
## 15.10 Device Identification

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

### 15.10.1 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 15-5. Conceptual 64-bit Peripheral ID**



**Table 15-2. Conceptual 64-Bit Peripheral ID Bit Descriptions**

Field	Size	Description	Location
JEP-106 CC code	4	Continuation code: 0x0	PID4
JEP-106 ID code	7	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)	PID2

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

### 15.10.2 Chip Identification Method

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

- Processor identification
- Product family identification

- Product series identification
- Device select

## **15.11 Functional Description**

### **15.11.1 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 (ADDR, LENGTH, DATA) are shared. These shared registers 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.

### **15.11.2 Basic Operation**

#### **15.11.2.1 Initialization**

The module is enabled by enabling its clocks. For more details, refer to [Clocks](#). The DSU registers can be PAC write-protected.

##### **Related Links**

[PAC - Peripheral Access Controller](#)

#### **15.11.2.2 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, accessing the first 0x100 bytes causes the system to return an error. Refer to [Intellectual Property Protection](#).

##### **Related Links**

[NVMCTRL – Nonvolatile Memory Controller Security Bit](#)

#### **15.11.2.3 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](#).

### **15.11.3 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 15-3. 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).

### 15.11.3.1 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 (DATA). 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, it is only possible to calculate the CRC32 of the entire Flash array when operated from the external debug interface, where 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. Such restriction is not applicable when the DSU is accessed by the CPU using the internal address range, as shown in [Figure 15-4](#).

The actual test is started by writing a '1' 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 '1' to CTRL.SWRST).

#### Related Links

[NVMCTRL – Nonvolatile Memory Controller Security Bit](#)

### 15.11.3.2 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.

### 15.11.4 Debug Communication Channels

The Debug Communication Channels (DCC0 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. 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).

Two Debug Communication Channel status bits in the Status B registers (STATUS.DCCDx) 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.

**Note:** The DCC0 and DCC1 registers are shared with the on-board memory testing logic (MBIST). Accordingly, DCC0 and DCC1 must not be used while performing MBIST operations.

#### **Related Links**

[NVMCTRL – Nonvolatile Memory Controller Security Bit](#)

### **15.11.5 Testing of On-Board Memories MBIST**

The DSU implements a feature for automatic testing of memory, also known as MBIST (memory built-in self test). This is primarily intended for production test of on-board memories. MBIST cannot be operated from the external address range when the device is protected by the NVMCTRL security bit. If an 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:

- 1.1. Write entire memory to '0', in any order.
- 1.2. Bit by bit read '0', write '1', in descending order.
- 1.3. Bit by bit read '1', write '0', read '0', write '1', in ascending order.
- 1.4. Bit by bit read '1', write '0', in ascending order.
- 1.5. Bit by bit read '0', write '1', read '1', write '0', in ascending order.
- 1.6. Read '0' from entire memory, in ascending order.

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

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

#### 2. Starting MBIST

To test a memory, you need to write the start address of the memory to the ADDR.ADDR bit field, and the size of the memory into the Length register.

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 '1' to CTRL.MBIST. A running MBIST operation can be canceled by writing a '1' 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:

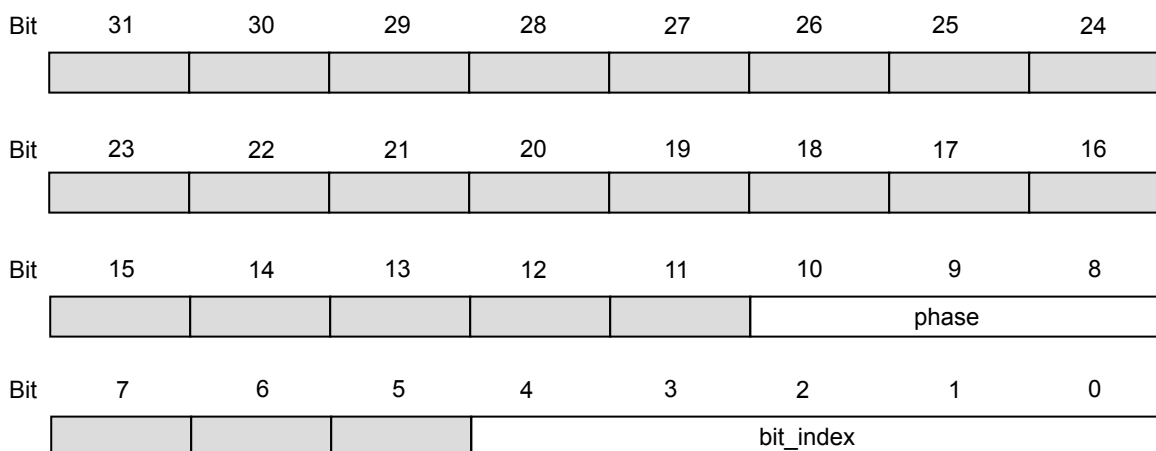
- ADDR.AMOD=0: exit-on-error (default)  
In this mode, the algorithm terminates either when a fault is detected or on successful completion. In both cases, STATUSA.DONE is set. If an error was detected, STATUSA.FAIL will be set. User then can read the DATA and ADDR registers to locate the fault.
- ADDR.AMOD=1: 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 '1' in STATUSA.FAIL to resume. Prior to resuming, user can read the DATA and ADDR registers to locate the fault.

#### 4. Locating Faults

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:

**Figure 15-6. DATA bits Description When MBIST Operation Returns an Error**



- bit\_index: contains the bit number of the failing bit
- phase: indicates which phase of the test failed and the cause of the error, as listed in the following table.

**Table 15-4. 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'

Phase	Test actions
6	Read '1', write '0', decrement address
7	Read all zeros. bit_index is not used

**Table 15-5. AMOD Bit Descriptions for MBIST**

AMOD[1:0]	Description
0x0	Exit on Error
0x1	Pause on Error
0x2, 0x3	Reserved

**Related Links**

- [NVMCTRL – Nonvolatile Memory Controller](#)
- [Security Bit](#)
- [Product Mapping](#)

**15.11.6 System Services Availability when Accessed Externally and Device is Protected**

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

Internal access: Access performed in the DSU address offset 0x000-0x100 range.

**Table 15-6. Available Features when Operated From The External Address Range and Device is Protected**

Features	Availability From The External Address Range and Device is Protected
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)



## 15.12 Register Summary

Offset	Name	Bit Pos.										
0x00	<b>CTRL</b>	7:0				CE	MBIST	CRC		SWRST		
0x01	<b>STATUSA</b>	7:0				PERR	FAIL	BERR	CRSTEXT	DONE		
0x02	<b>STATUSB</b>	7:0				HPE	DCCDx	DCCDx	DBGPRES	PROT		
0x03	Reserved											
0x04	<b>ADDR</b>	7:0	ADDR[5:0]						AMOD[1:0]			
		15:8	ADDR[13:6]									
		23:16	ADDR[21:14]									
		31:24	ADDR[29:22]									
0x08	<b>LENGTH</b>	7:0	LENGTH[5:0]									
		15:8	LENGTH[13:6]									
		23:16	LENGTH[21:14]									
		31:24	LENGTH[29:22]									
0x0C	<b>DATA</b>	7:0	DATA[7:0]									
		15:8	DATA[15:8]									
		23:16	DATA[23:16]									
		31:24	DATA[31:24]									
0x10	<b>DCC0</b>	7:0	DATA[7:0]									
		15:8	DATA[15:8]									
		23:16	DATA[23:16]									
		31:24	DATA[31:24]									
0x14	<b>DCC1</b>	7:0	DATA[7:0]									
		15:8	DATA[15:8]									
		23:16	DATA[23:16]									
		31:24	DATA[31:24]									
0x18	<b>DID</b>	7:0	DEVSEL[7:0]									
		15:8	DIE[3:0]			REVISION[3:0]						
		23:16	FAMILY[0:0]	SERIES[5:0]								
		31:24	PROCESSOR[3:0]			FAMILY[4:1]						
0x1C ... 0x0FFF	Reserved											
0x1000	<b>ENTRY0</b>	7:0							FMT	EPRES		
		15:8	ADDOFF[3:0]									
		23:16	ADDOFF[11:4]									
		31:24	ADDOFF[19:12]									
0x1004	<b>ENTRY1</b>	7:0							FMT	EPRES		
		15:8	ADDOFF[3:0]									
		23:16	ADDOFF[11:4]									
		31:24	ADDOFF[19:12]									
0x1008	<b>END</b>	7:0	END[7:0]									
		15:8	END[15:8]									
		23:16	END[23:16]									
		31:24	END[31:24]									

# ATSAMHAXEXXA

## DSU - Device Service Unit

Offset	Name	Bit Pos.							
0x100C ... 0x1FCB	Reserved								
0x1FCC	MEMTYPE	7:0							SMEMP
		15:8							
		23:16							
		31:24							
0x1FD0	PID4	7:0	FKBC[3:0]			JEPCC[3:0]			
		15:8							
		23:16							
		31:24							
0x1FD4 ... 0x1FDF	Reserved								
0x1FE0	PID0	7:0	PARTNBL[7:0]						
		15:8							
		23:16							
		31:24							
0x1FE4	PID1	7:0	JEPIDCL[3:0]			PARTNBH[3:0]			
		15:8							
		23:16							
		31:24							
0x1FE8	PID2	7:0	REVISION[3:0]			JEPU	JEPIDCH[2:0]		
		15:8							
		23:16							
		31:24							
0x1FEC	PID3	7:0	REVAND[3:0]			CUSMOD[3:0]			
		15:8							
		23:16							
		31:24							
0x1FF0	CID0	7:0	PREAMBLEB0[7:0]						
		15:8							
		23:16							
		31:24							
0x1FF4	CID1	7:0	CCLASS[3:0]			PREAMBLE[3:0]			
		15:8							
		23:16							
		31:24							
0x1FF8	CID2	7:0	PREAMBLEB2[7:0]						
		15:8							
		23:16							
		31:24							
0x1FFC	CID3	7:0	PREAMBLEB3[7:0]						
		15:8							
		23:16							
		31:24							

### **15.13 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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#).

**15.13.1 Control**

**Name:** CTRL  
**Offset:** 0x0000  
**Reset:** 0x00  
**Property:** PAC Write-Protection

	7	6	5	4	3	2	1	0
				CE	MBIST	CRC		SWRST
Access				W	W	W		W
Reset				0	0	0		0

**Bit 4 – CE** Chip-Erase

Writing a '0' to this bit has no effect.

Writing a '1' to this bit starts the Chip-Erase operation.

**Bit 3 – MBIST** Memory Built-In Self-Test

Writing a '0' to this bit has no effect.

Writing a '1' to this bit starts the memory BIST algorithm.

**Bit 2 – CRC** 32-bit Cyclic Redundancy Check

Writing a '0' to this bit has no effect.

Writing a '1' to this bit starts the cyclic redundancy check algorithm.

**Bit 0 – SWRST** Software Reset

Writing a '0' to this bit has no effect.

Writing a '1' to this bit resets the module.

### 15.13.2 Status A

**Name:** STATUSA  
**Offset:** 0x0001  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

**Bit 4 – PERR** Protection Error

Writing a '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' to this bit clears the Done bit.

This bit is set when a DSU operation is completed.

### 15.13.3 Status B

**Name:** STATUSB  
**Offset:** 0x0002  
**Reset:** 0x1X  
**Property:** PAC Write-Protection

	7	6	5	4	3	2	1	0
				HPE	DCCDx	DCCDx	DBGPRES	PROT
Access				R	R	R	R	R
Reset				1	0	0	x	x

**Bit 4 – HPE** Hot-Plugging Enable

Writing a '0' to this bit has no effect.

Writing a '1' 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** Debug Communication Channel x Dirty [x=1..0]

Writing a '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' to this bit has no effect.

This bit is set at power-up when the device is protected.

This bit is never cleared.

### 15.13.4 Address

**Name:** ADDR  
**Offset:** 0x0004  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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**

The functionality of these bits is dependent on the operation mode.

Bit description when operating CRC32: refer to [32-bit Cyclic Redundancy Check CRC32](#)

Bit description when testing onboard memories (MBIST): refer to [Testing of On-Board Memories MBIST](#)

### 15.13.5 Length

**Name:** LENGTH  
**Offset:** 0x0008  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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		
Reset	0	0	0	0	0	0		

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



**15.13.6 Data**

**Name:** DATA  
**Offset:** 0x000C  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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.

**15.13.7 Debug Communication Channel 0**

**Name:** DCC0  
**Offset:** 0x0010  
**Reset:** 0x00000000  
**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.

**15.13.8 Debug Communication Channel 1**

**Name:** DCC1  
**Offset:** 0x0014  
**Reset:** 0x00000000  
**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.

### 15.13.9 Device Identification

**Name:** DID  
**Offset:** 0x0018  
**Property:** PAC Write-Protection

The information in this register is related to the *Ordering Information*.

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	p	p	p	p	f	f	f	f
Bit	23	22	21	20	19	18	17	16
	FAMILY[0:0]		SERIES[5:0]					
Access	R		R	R	R	R	R	R
Reset	f		s	s	s	s	s	s
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	d	d	d	d	r	r	r	r
Bit	7	6	5	4	3	2	1	0
	DEVSEL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

**Bits 31:28 – PROCESSOR[3:0]** Processor

The value of this field defines the processor used on the device.

**Bits 27:23 – FAMILY[4:0]** Product Family

The value of this field corresponds to the product family part of the ordering code.

**Bits 21:16 – SERIES[5:0]** Product Series

The value of this field corresponds to the product series part of the ordering code.

**Bits 15:12 – DIE[3:0]** Die Number

Identifies the die family.

**Bits 11:8 – REVISION[3:0]** Revision Number

Identifies the die revision number. 0x0=rev.A, 0x1=rev.B etc.

**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.

**Bits 7:0 – DEVSEL[7:0]** Device Selection

This bit field identifies a device within a product family and product series. Refer to the ordering information for device configurations and corresponding values for Flash memory density, pin count, and device variant.

**15.13.10 CoreSight ROM Table Entry 0**

**Name:** ENTRY0  
**Offset:** 0x1000  
**Reset:** 0XXXXXX00X  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	ADDOFF[19:12]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x
Bit	23	22	21	20	19	18	17	16
	ADDOFF[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x
Bit	15	14	13	12	11	10	9	8
	ADDOFF[3:0]							
Access	R	R	R	R				
Reset	x	x	x	x				
Bit	7	6	5	4	3	2	1	0
							FMT	EPRES
Access							R	R
Reset							1	x

**Bits 31:12 – ADDOFF[19:0] Address Offset**

The base address of the component, relative to the base address of this ROM table.

**Bit 1 – FMT Format**

Always reads as '1', indicating 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 power-up if the device is not protected indicating that the entry is not present.

This bit is cleared at power-up if the device is not protected indicating that the entry is present.

**15.13.11 CoreSight ROM Table Entry 1**

**Name:** ENTRY1  
**Offset:** 0x1004  
**Reset:** 0XXXXXX00X  
**Property:** PAC Write-Protection

	Bit	31	30	29	28	27	26	25	24	
		ADDOFF[19:12]								
Access		R	R	R	R	R	R	R	R	
Reset		x	x	x	x	x	x	x	x	
	Bit	23	22	21	20	19	18	17	16	
		ADDOFF[11:4]								
Access		R	R	R	R	R	R	R	R	
Reset		x	x	x	x	x	x	x	x	
	Bit	15	14	13	12	11	10	9	8	
		ADDOFF[3:0]								
Access		R	R	R	R					
Reset		x	x	x	x					
	Bit	7	6	5	4	3	2	1	0	
								FMT	EPRES	
Access								R	R	
Reset								1	x	

**Bits 31:12 – ADDOFF[19:0] Address Offset**

The base address of the component, relative to the base address of this ROM table.

**Bit 1 – FMT Format**

Always read as '1', indicating 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 power-up if the device is not protected indicating that the entry is not present.

This bit is cleared at power-up if the device is not protected indicating that the entry is present.

**15.13.12 CoreSight ROM Table End**

**Name:** END  
**Offset:** 0x1008  
**Reset:** 0x00000000  
**Property:** -

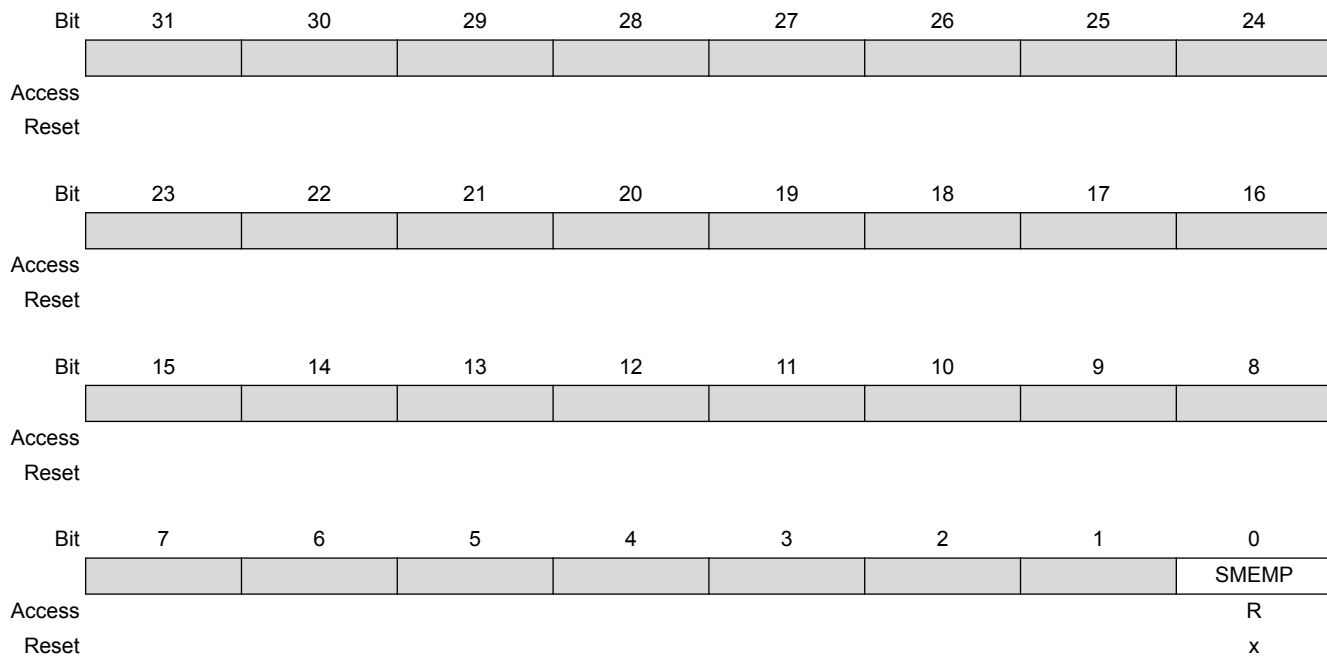
Bit	31	30	29	28	27	26	25	24
END[31:24]								
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
END[23:16]								
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
END[15:8]								
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
END[7:0]								
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.



**15.13.13 CoreSight ROM Table Memory Type**

**Name:** MEMTYPE  
**Offset:** 0x1FCC  
**Reset:** 0x0000000x  
**Property:** -



**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 power-up if the device is not protected, indicating that the system memory is accessible from a debug adapter.

This bit is cleared at power-up if the device is protected, indicating that the system memory is not accessible from a debug adapter.

**15.13.14 Peripheral Identification 4**

**Name:** PID4  
**Offset:** 0x1FD0  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
Access								
Reset								
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 7:4 – FKBC[3:0] 4KB Count**

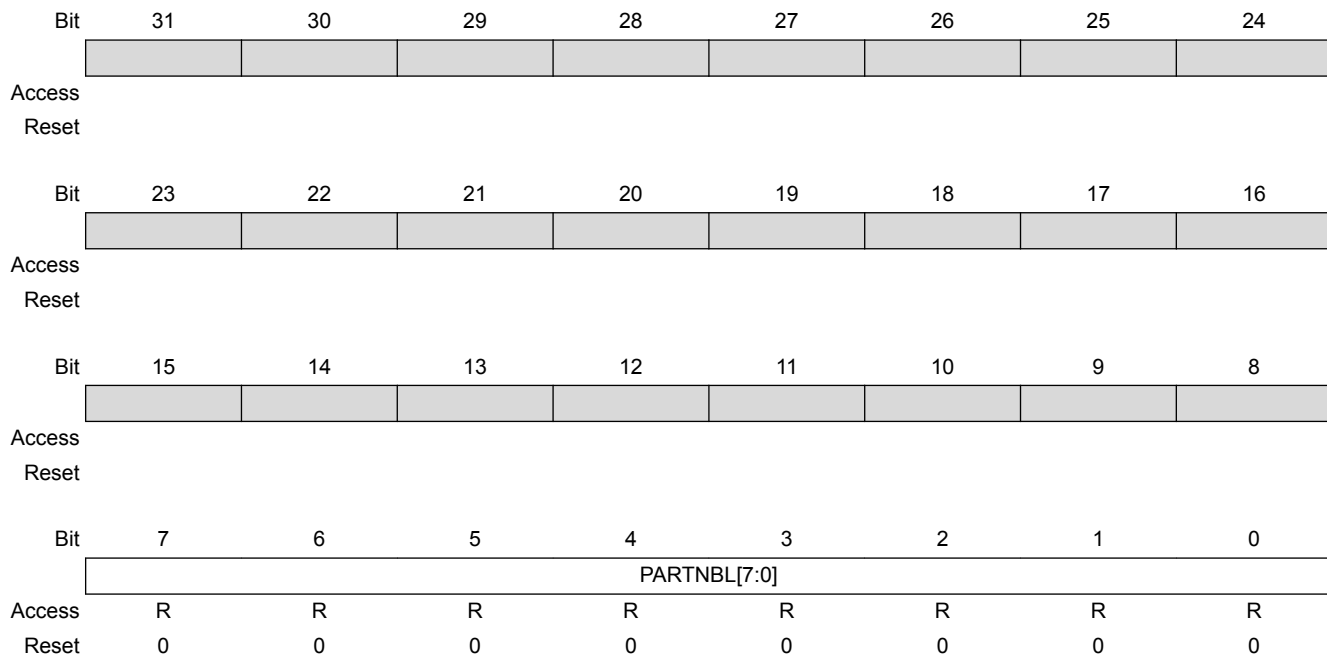
These bits will always return zero when read, indicating that this debug component occupies one 4KB block.

**Bits 3:0 – JEPCC[3:0] JEP-106 Continuation Code**

These bits will always return zero when read.

**15.13.15 Peripheral Identification 0**

**Name:** PID0  
**Offset:** 0x1FE0  
**Reset:** 0x00000000  
**Property:** -

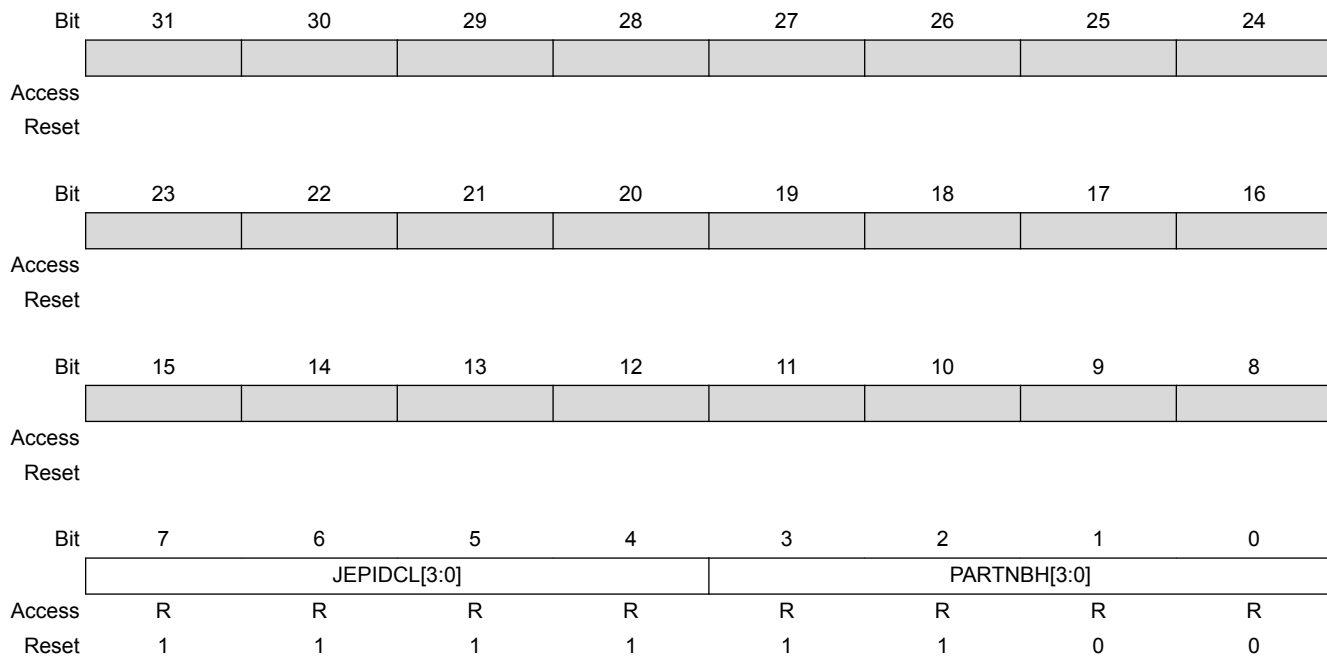


**Bits 7:0 – PARTNBL[7:0] Part Number Low**

These bits will always return 0xD0 when read, indicating that this device implements a DSU module instance.

**15.13.16 Peripheral Identification 1**

**Name:** PID1  
**Offset:** 0x1FE4  
**Reset:** 0x000000FC  
**Property:** -

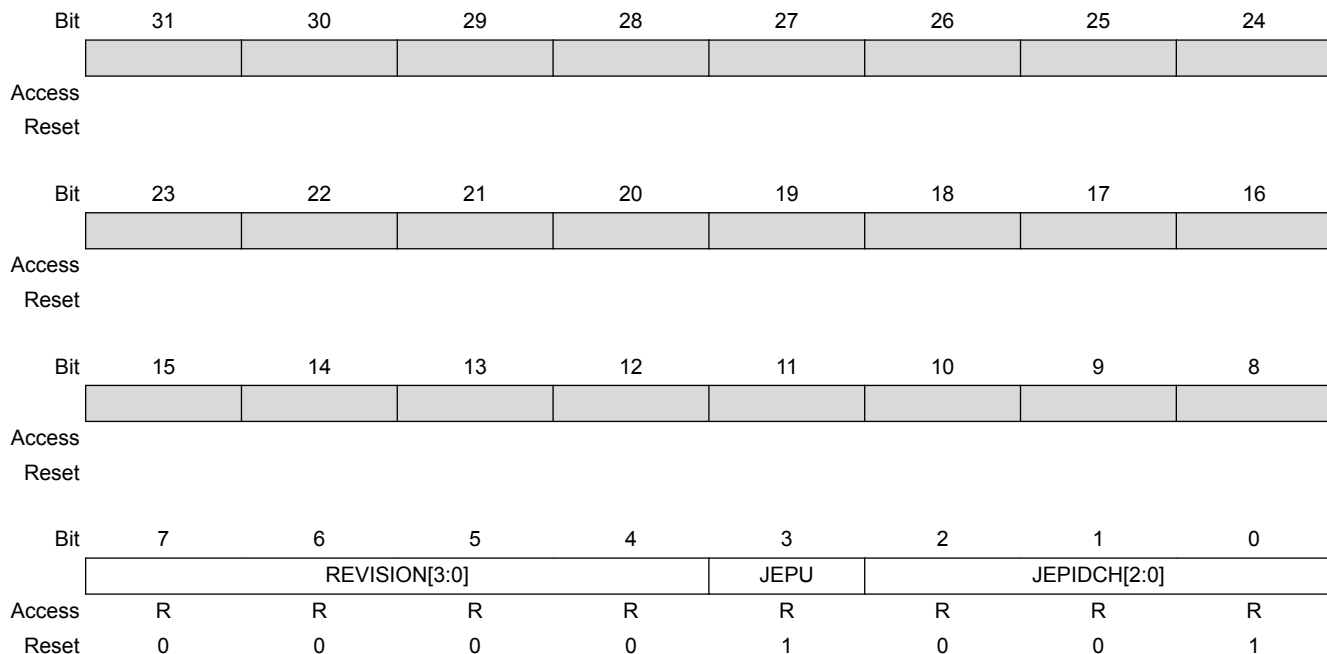


**Bits 7:4 – JEPIDCL[3:0]** Low part of the JEP-106 Identity Code  
 These bits will always return 0xF when read (JEP-106 identity code is 0x1F).

**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.

**15.13.17 Peripheral Identification 2**

**Name:** PID2  
**Offset:** 0x1FE8  
**Reset:** 0x00000009  
**Property:** -



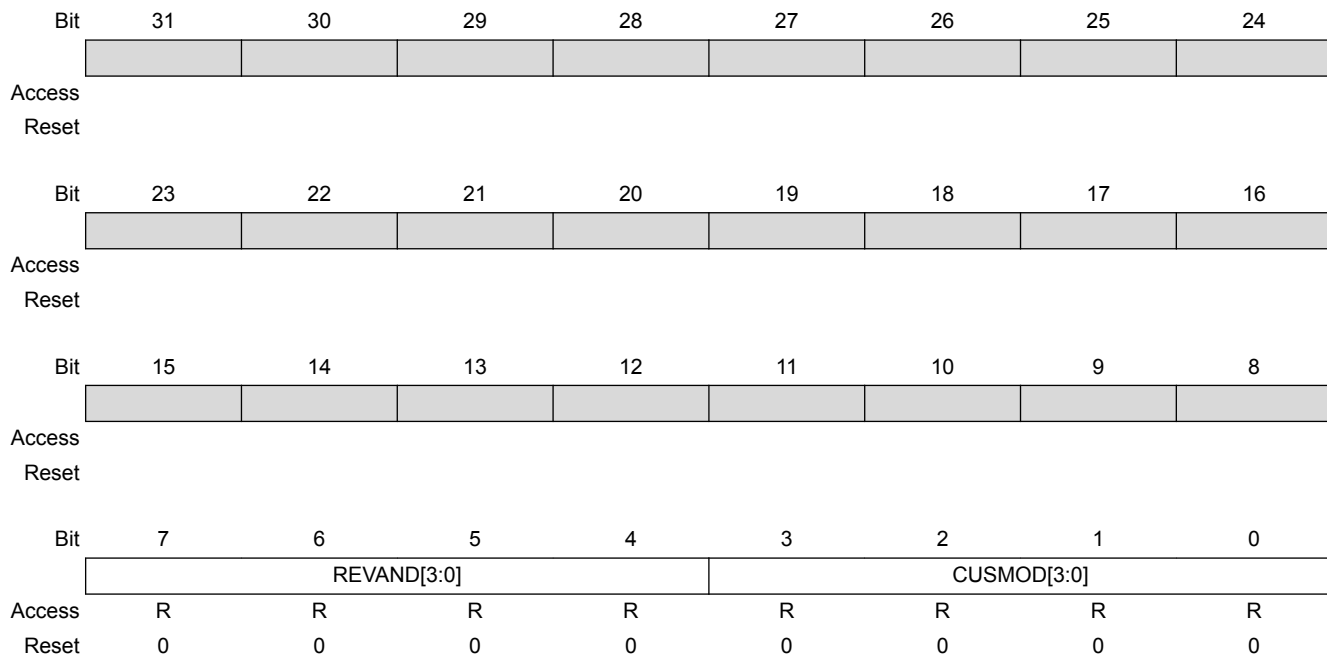
**Bits 7:4 – REVISION[3:0]** Revision Number  
Revision of the peripheral. Starts at 0x0 and increments by one at both major and minor revisions.

**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, (JEP-106 identity code is 0x1F).

**15.13.18 Peripheral Identification 3**

**Name:** PID3  
**Offset:** 0x1FEC  
**Reset:** 0x00000000  
**Property:** -

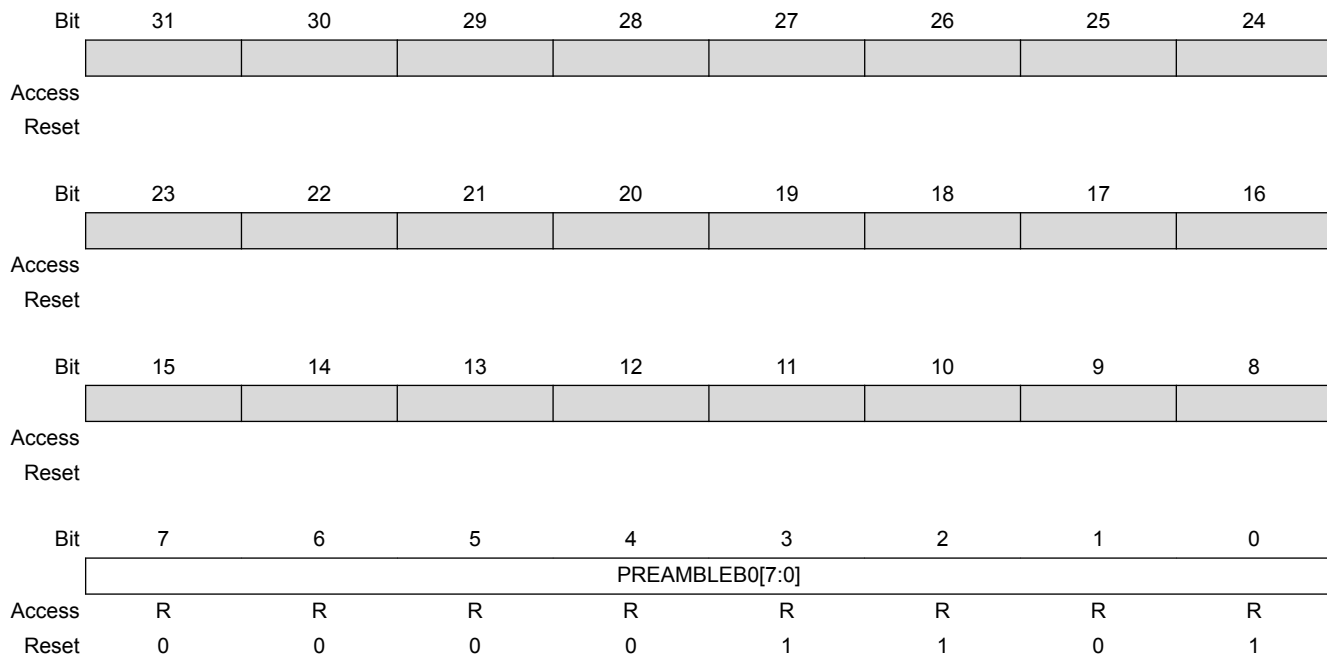


**Bits 7:4 – REVAND[3:0]** Revision Number  
 These bits will always return 0x0 when read.

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

**15.13.19 Component Identification 0**

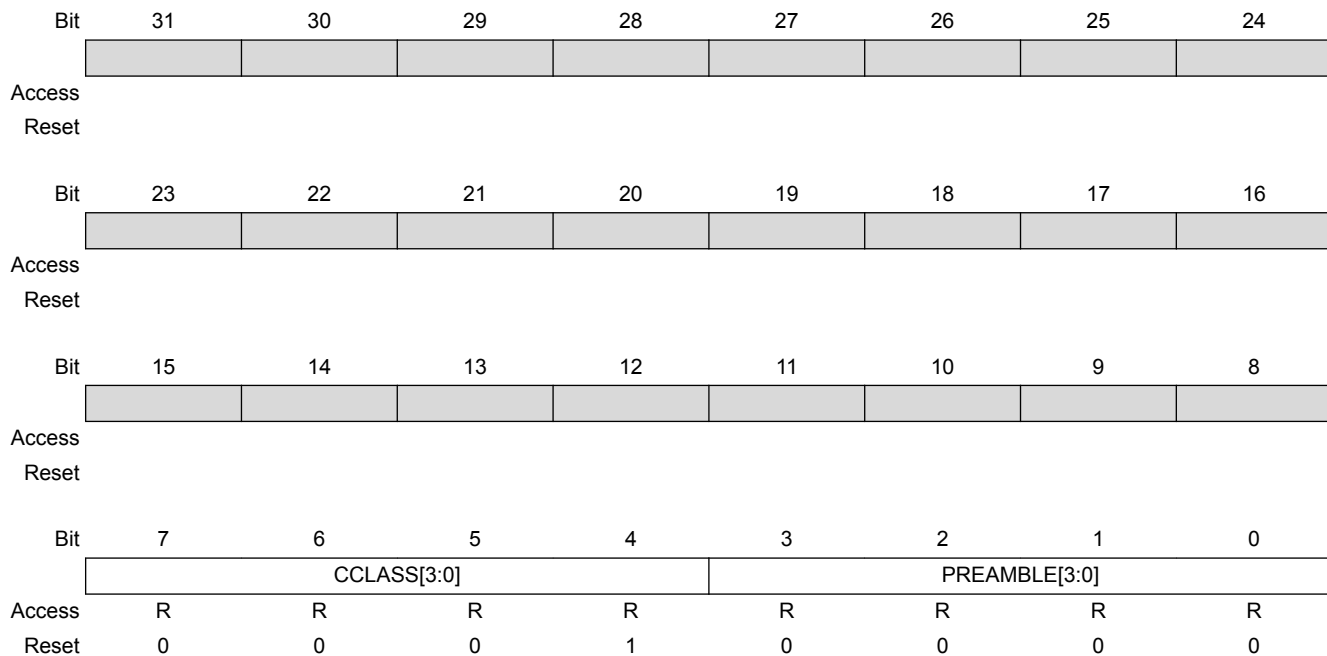
**Name:** CID0  
**Offset:** 0x1FF0  
**Reset:** 0x0000000D  
**Property:** -



**Bits 7:0 – PREAMBLEB0[7:0]** Preamble Byte 0  
 These bits will always return 0x0000000D when read.

**15.13.20 Component Identification 1**

**Name:** CID1  
**Offset:** 0x1FF4  
**Reset:** 0x00000010  
**Property:** -



**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>).

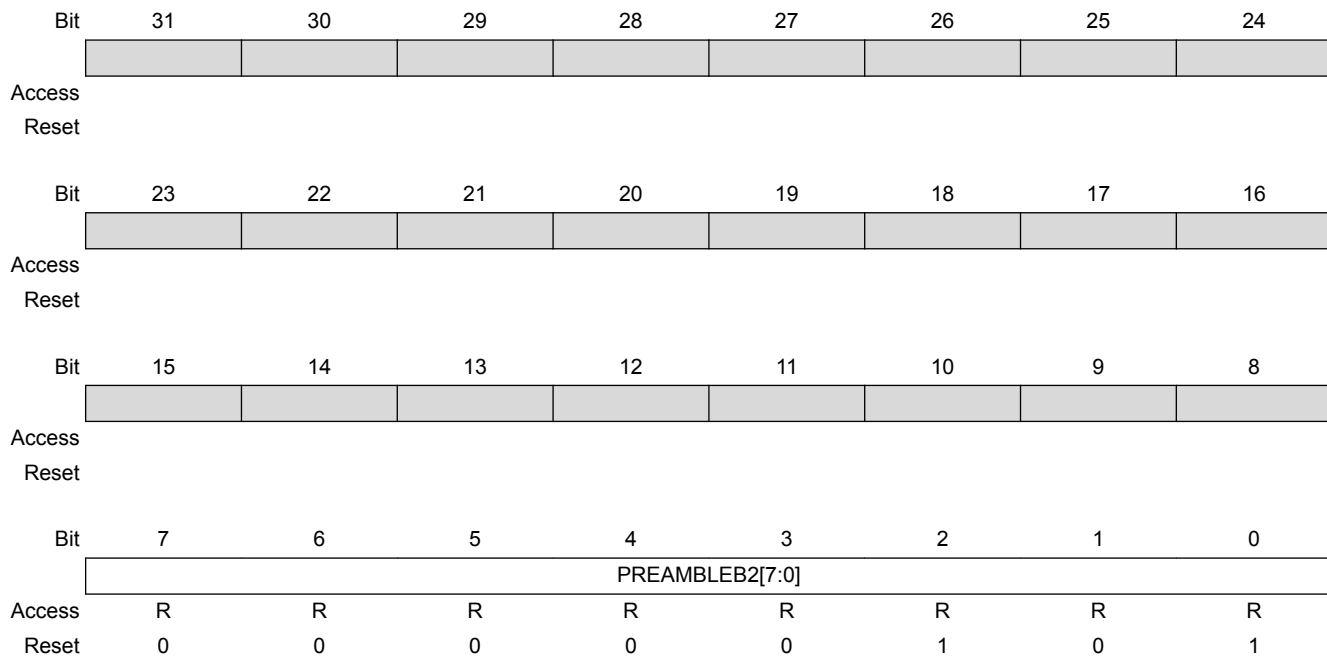
**Bits 3:0 – PREAMBLE[3:0] Preamble**

These bits will always return 0x00 when read.



**15.13.21 Component Identification 2**

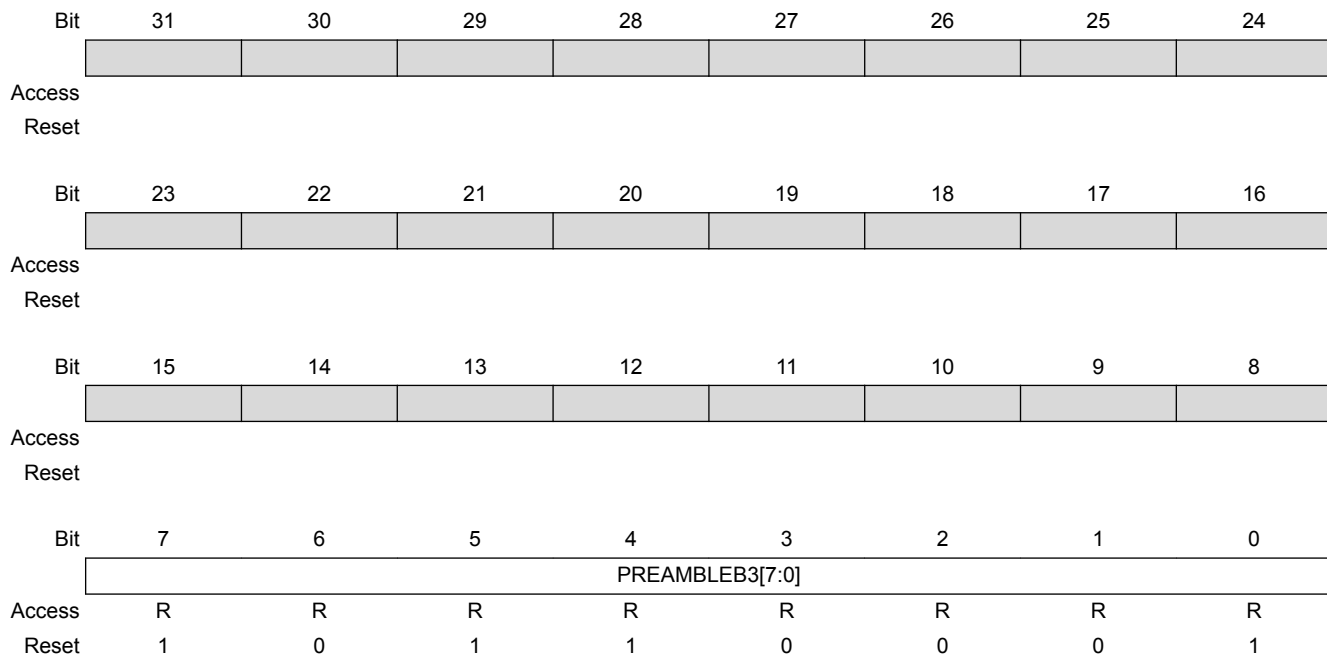
**Name:** CID2  
**Offset:** 0x1FF8  
**Reset:** 0x00000005  
**Property:** -



**Bits 7:0 – PREAMBLEB2[7:0]** Preamble Byte 2  
 These bits will always return 0x00000005 when read.

**15.13.22 Component Identification 3**

**Name:** CID3  
**Offset:** 0x1FFC  
**Reset:** 0x000000B1  
**Property:** -



**Bits 7:0 – PREAMBLEB3[7:0]** Preamble Byte 3  
 These bits will always return 0x000000B1 when read.

## 16. Clock System

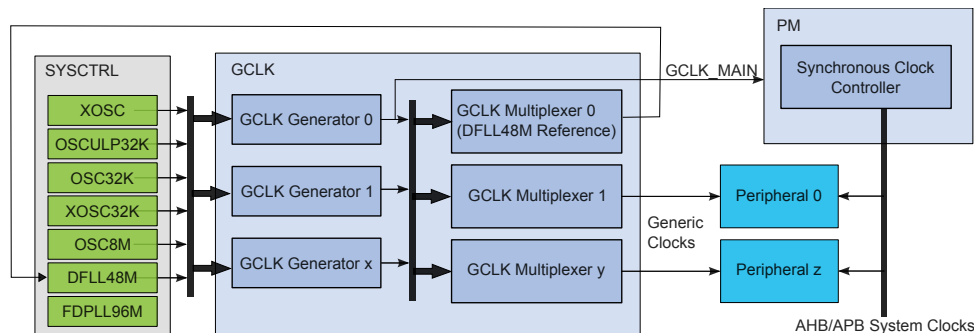
This chapter summarizes the clock distribution and terminology in the SAMHA0/1 device. It will not explain every detail of its configuration. For in-depth documentation, see the respective peripherals descriptions and the *Generic Clock* documentation.

### Related Links

[GCLK - Generic Clock Controller](#)

### 16.1 Clock Distribution

Figure 16-1. Clock distribution



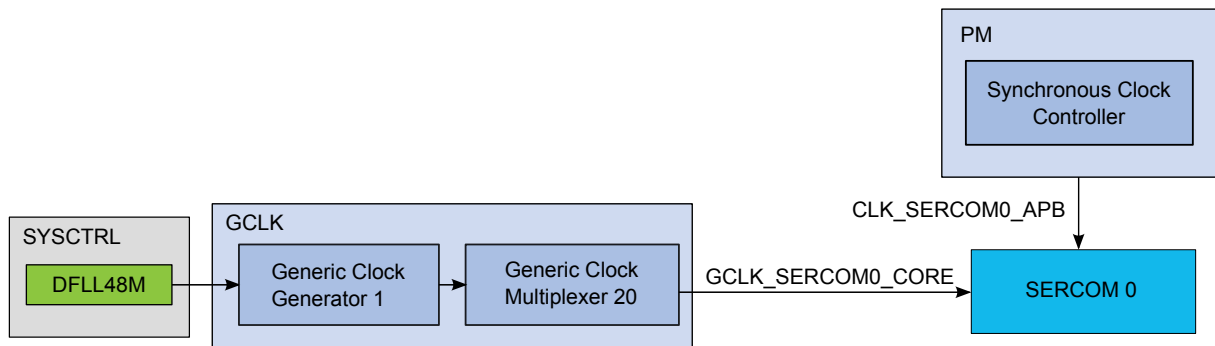
The clock system on the SAMHA0/1 consists of:

- *Clock sources*, controlled by SYSCTRL
  - A clock source provides a time base that is used by other components, such as Generic Clock Generators. 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*: These are programmable prescalers that can use any of the system clock sources as a time base. The Generic Clock Generator 0 generates the clock signal GCLK\_MAIN, which is used by the Power Manager, which in turn generates synchronous clocks.
  - *Generic Clocks*: These are clock signals generated by Generic Clock Generators and output by the Generic Clock Multiplexer, and serve as clocks for the peripherals of the system. Multiple instances of a peripheral will typically have a separate Generic Clock for each instance. Generic Clock 0 serves as the clock source for the DFLL48M clock input (when multiplying another clock source).
- Power Manager (PM)
  - The PM generates and controls the 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.

The next figure 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 feeds into Peripheral Channel 20. The Generic Clock 20, also called GCLK\_SERCOM0\_CORE, is connected to

SERCOM0. The SERCOM0 interface, clocked by CLK\_SERCOM0\_APB, has been unmasked in the APBC Mask register in the PM.

**Figure 16-2. Example of SERCOM clock**



## 16.2 Synchronous and Asynchronous Clocks

As the CPU and the peripherals can be in different clock domains, i.e. they are clocked from different clock sources and/or with different clock speeds, some peripheral accesses by the CPU need to be synchronized. In this case the peripheral includes a SYNCBUSY status register that can be used to check if a sync operation is in progress.

For a general description, see [Register Synchronization](#). Some peripherals have specific properties described in their individual sub-chapter “Synchronization”.

In the datasheet, references to Synchronous Clocks are referring to the CPU and bus clocks, while asynchronous clocks are generated by the Generic Clock Controller (GCLK).

## 16.3 Register Synchronization

There are two different register synchronization schemes implemented on this device: *common synchronizer register synchronization* and *distributed synchronizer register synchronization*.

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

The modules adopting a distributed synchronizer register synchronization are: SERCOM USART, SERCOM SPI, SERCOM I2C.

### 16.3.1 Common Synchronizer Register Synchronization

#### 16.3.1.1 Overview

All peripherals are composed of one digital bus interface connected to the APB or AHB bus and running from a corresponding clock in the Main Clock domain, and one peripheral core running from the peripheral Generic Clock (GCLK).

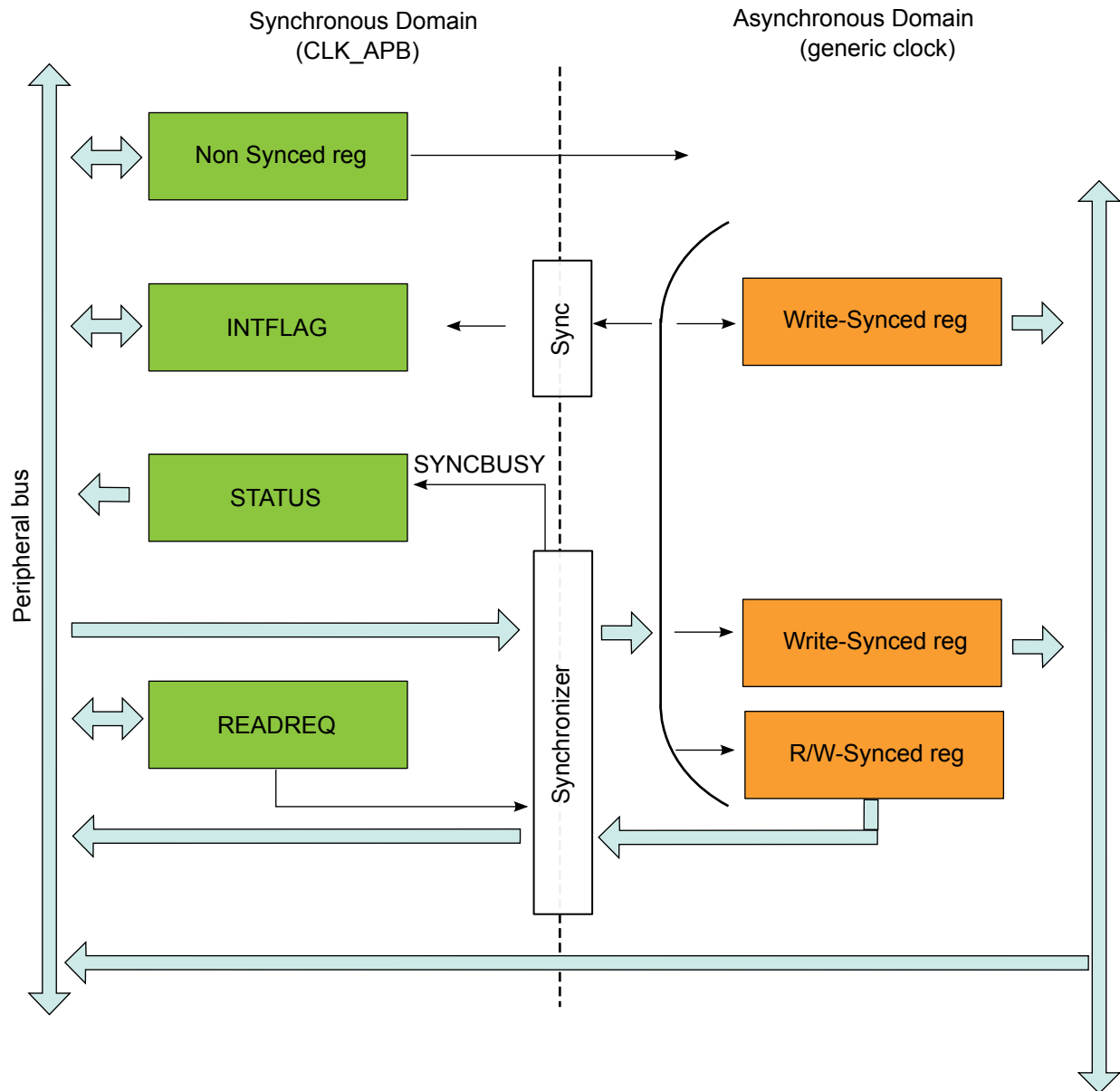
Communication between these clock domains must be synchronized. This mechanism is implemented in hardware, so the synchronization process takes place even if the peripheral generic clock is running from the same clock source and on the same frequency as the bus interface.

All registers in the bus interface are accessible without synchronization. All registers in the peripheral core are synchronized when written. Some registers in the peripheral core are synchronized when read.

Each individual register description will have the properties "Read-Synchronized" and/or "Write-Synchronized" if a register is synchronized.

As shown in the figure below, the common synchronizer is used for all registers in one peripheral. Therefore, status register (STATUS) of each peripheral can be synchronized at a time.

**Figure 16-3. Synchronization**



**16.3.1.2 Write-Synchronization**

Write-Synchronization is triggered by writing to a register in the peripheral clock domain. 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](#) 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 peripheral core register
- Reading a read-synchronized peripheral core register
- Reading the register that is being written (and thus triggered the synchronization)

Peripheral 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.

#### 16.3.1.3 Read-Synchronization

Reading a read-synchronized peripheral 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](#) for details on the synchronization delay. Note that reading a read-synchronized peripheral 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.

#### 16.3.1.4 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.

#### 16.3.1.5 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 peripheral core registers, the read request mechanism can be used.

##### Basic Read Request

Writing a '1' 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 '1' 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 '1' 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:

The 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.

### 16.3.1.6 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 peripheral core register
- Writing an APB register
- Reading a read-synchronized peripheral core register

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

### 16.3.1.7 Software Reset Write-Synchronization

Writing a '1' to the Software Reset bit in CTRL (CTRL.SWRST) will also trigger write-synchronization and set STATUS.SYNCBUSY. When writing a '1' to the CTRL.SWRST bit it will immediately read as '1'. 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 '1'), 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 peripheral 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.

### 16.3.1.8 Synchronization Delay

The synchronization will delay write and read accesses by a certain amount. This delay  $D$  is within the range of:

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

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  $2 \times P_{APB}$ .

## 16.3.2 Distributed Synchronizer Register Synchronization

### 16.3.2.1 Overview

All peripherals are composed of one digital bus interface connected to the APB or AHB bus and running from a corresponding clock in the Main Clock domain, and one peripheral core running from the peripheral Generic Clock (GCLK).

Communication between these clock domains must be synchronized. This mechanism is implemented in hardware, so the synchronization process takes place even if the peripheral generic clock is running from the same clock source and on the same frequency as the bus interface.

All registers in the bus interface are accessible without synchronization. All registers in the peripheral core are synchronized when written. Some registers in the peripheral core are synchronized when read. Registers that need synchronization has this denoted in each individual register description.

### 16.3.2.2 General Write synchronization

Write-Synchronization is triggered by writing to a register in the peripheral clock domain. The respective bit in the Synchronization Busy register (SYNCBUSY) will be set when the write-synchronization starts and cleared when the write-synchronization is complete. Refer to [Synchronization Delay](#) for details on the synchronization delay.

When write-synchronization is ongoing for a register, any subsequent write attempts to this register will be discarded, and an error will be reported.

Example:

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

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

Synchronization is per register, so multiple registers can be synchronized in parallel. Consequently, after REGA (8-bit access) was written, REGB (8-bit access) can be written immediately without error.

REGC (16-bit access) can be written without affecting REGA or REGB. If REGC is written to in two consecutive 8-bit accesses without waiting for synchronization, the second write attempt will be discarded and an error is generated.

A 32-bit access to offset 0x00 will write all three registers. Note that REGA, REGB and REGC can be updated at different times because of independent write synchronization.

### 16.3.2.3 General read synchronization

Read-synchronized registers are synchronized when the register value is updated. During synchronization the corresponding bit in SYNCBUSY will be set. Reading a read-synchronized register will return its value immediately and the corresponding bit in SYNCBUSY will not be set.

### 16.3.2.4 Completion of synchronization

In order to check if synchronization is complete, the user can either poll the relevant bits in SYNCBUSY or use the Synchronisation Ready interrupt (if available). The Synchronization Ready interrupt flag will be set when all ongoing synchronizations are complete, i.e. when all bits in SYNCBUSY are '0'.

### 16.3.2.5 Enable Write-Synchronization

Setting the Enable bit in a module's Control register (CTRL.ENABLE) will also trigger write-synchronization and set SYNCBUSY.ENABLE. CTRL.ENABLE will read its new value immediately after being written. SYNCBUSY.ENABLE will be cleared by hardware when the operation is complete. The Synchronisation Ready interrupt (if available) cannot be used for Enable write-synchronization.

### 16.3.2.6 Software Reset Write-Synchronization

Setting the Software Reset bit in CTRLA (CTRLA.SWRST=1) will trigger write-synchronization and set SYNCBUSY.SWRST. When writing a '1' to the CTRLA.SWRST bit it will immediately read as '1'. CTRLA.SWRST and SYNCBUSY.SWRST will be cleared by hardware when the peripheral has been reset. Writing a '0' to the CTRLA.SWRST bit has no effect. The Ready interrupt (if available) cannot be used for Software Reset write-synchronization.



### 16.3.2.7 Synchronization Delay

The synchronization will delay write and read accesses by a certain amount. This delay  $D$  is within the range of:

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

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  $2 \times P_{APB}$ .

## 16.4 Enabling a Peripheral

In order to enable a peripheral that is clocked by a Generic Clock, the following parts of the system needs to be configured:

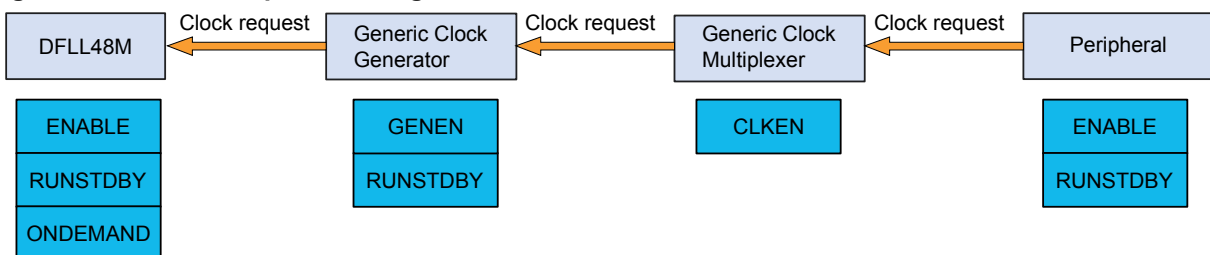
- A running Clock Source.
- A clock from the Generic Clock Generator must be configured to use one of the running Clock Sources, and the Generator must be enabled.
- The Generic Clock Multiplexer that provides the Generic Clock signal to the peripheral must be configured to use a running Generic Clock 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 all 0's and any writing attempts to the peripheral will be discarded.

## 16.5 Disabling a Peripheral

When disabling a peripheral and if a pin change interrupt is enabled on pins driven by the respective peripheral, a wake condition may be generated. If this happen the interrupt flag will not be set. As a consequence the system will not be able to identify the wake source. To avoid this, the interrupt enable register of the peripheral must be cleared (or the Nested Vectored Interrupt Controller (NVIC) Enable for the peripheral must be cleared) before disabling the peripheral.

## 16.6 On-demand, Clock Requests

**Figure 16-4. Clock request routing**



All clock sources in the system can be run in an on-demand mode: the clock source is in a stopped state unless a peripheral is 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 has an active request, the clock source will be stopped until requested again.

The clock request can reach the clock source only if the peripheral, the generic clock and the clock from the Generic Clock Generator in-between are 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  $T_{start}$  from a clock request until the clock is available for the peripheral is between:

$$T_{start\_max} = \text{Clock source startup time} + 2 \times \text{clock source periods} + 2 \times \text{divided clock source periods}$$

$$T_{start\_min} = \text{Clock source startup time} + 1 \times \text{clock source period} + 1 \times \text{divided clock source period}$$

The time between the last active clock request stopped and the clock is shut down,  $T_{stop}$ , is between:

$$T_{stop\_min} = 1 \times \text{divided clock source period} + 1 \times \text{clock source period}$$

$$T_{stop\_max} = 2 \times \text{divided clock source periods} + 2 \times \text{clock source periods}$$

The On-Demand function can be disabled individually for each clock source by clearing the ONDEMAND bit located in each clock source controller. Consequently, the clock will always run whatever the clock request status is. This has the effect of removing the clock source startup time at the cost of power consumption.

The clock request mechanism can be configured to work in standby mode by setting the RUNSDTBY bits of the modules, see [Figure 16-4](#).

### 16.7 Power Consumption vs. Speed

When targeting for either a low-power or a fast acting system, some considerations have to be taken into account due to the nature of the asynchronous clocking of the peripherals:

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 take longer with a slower peripheral clock. This will cause worse response times and longer synchronization delays.

### 16.8 Clocks after Reset

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

- OSC8M is enabled and divided by 8
- Generic Generator 0 uses OSC8M as source and generates GCLK\_MAIN
- CPU and BUS clocks are undivided

On a Power Reset, the GCLK module starts to its initial state:

- All Generic Clock Generators are disabled except
  - Generator 0 is using OSC8M as source without division and generates GCLK\_MAIN
  - Generator 2 uses OSCULP32K as source without division
- All Generic Clocks are disabled except:
  - WDT Generic Clock uses the Generator 2 as source

On a User Reset the GCLK module starts to its initial state, except for:

- Generic Clocks that are write-locked, i.e., the according WRTLOCK is set to 1 prior to Reset or WDT Generic Clock if the WDT Always-On at power on bit set in the NVM User Row
- Generic Clock is 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.

## 17. GCLK - Generic Clock Controller

### 17.1 Overview

Depending on the application, peripherals may require specific clock frequencies to operate correctly. The Generic Clock controller GCLK provides nine Generic Clock Generators that can provide a wide range of clock frequencies.

Generators can be set to use different external and internal oscillators as source. The clock of each Generator can be divided. The outputs from the Generators are used as sources for the Generic Clock Multiplexers, which provide the Generic Clock (GCLK\_PERIPHERAL) to the peripheral modules, as shown in Generic Clock Controller Block Diagram. The number of Peripheral Clocks depends on how many peripherals the device has.

**Note:** The Generator 0 is always the direct source of the GCLK\_MAIN signal.

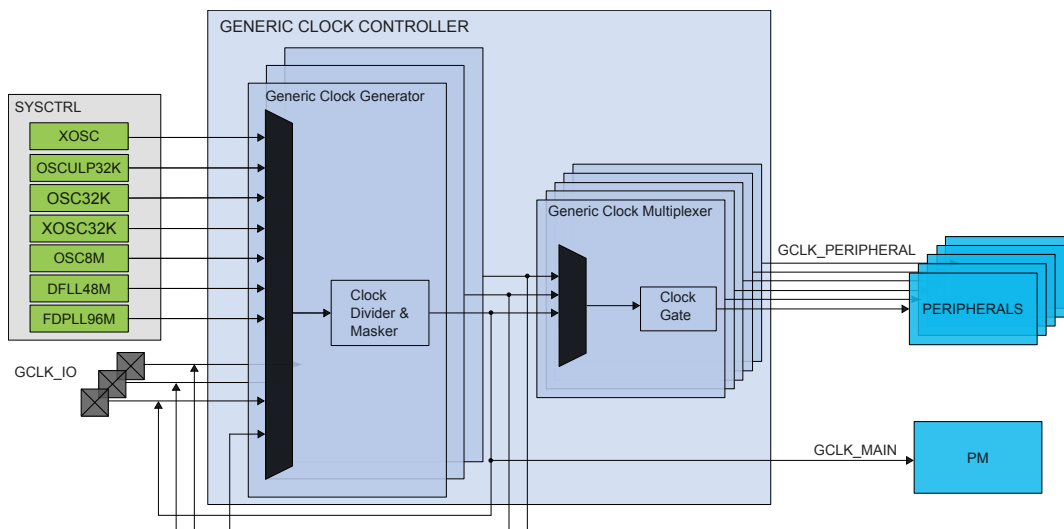
### 17.2 Features

- Provides Generic Clocks
- Wide frequency range
- Clock source for the generator can be changed on the fly

### 17.3 Block Diagram

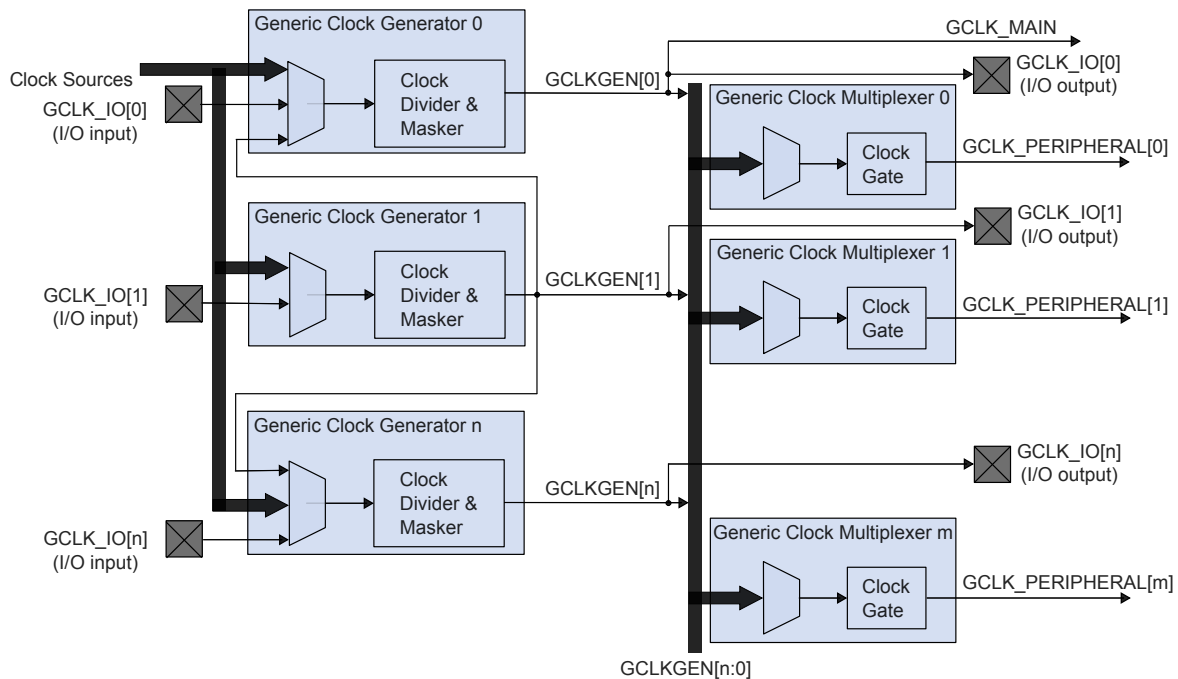
The generation of Peripheral Clock signals (GCLK\_PERIPHERAL) and the Main Clock (GCLK\_MAIN) can be seen in the figure below.

**Figure 17-1. Device Clocking Diagram**



The GCLK block diagram is shown in the next figure.

**Figure 17-2. Generic Clock Controller Block Diagram<sup>(1)</sup>**



**Note:** 1. If GENCTRL.SRC=0x01(GCLKIN), the GCLK\_IO is set as an input.

## 17.4 Signal Description

**Table 17-1. Signal Description**

Signal Name	Type	Description
GCLK_IO[7:0]	Digital I/O	Clock source for Generators when input Generic Clock signal when output

Refer to PORT Function Multiplexing table in I/O Multiplexing and Considerations for details on the pin mapping for this peripheral.

**Note:** One signal can be mapped on several pins.

### Related Links

[I/O Multiplexing and Considerations](#)

## 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

Using the GCLK I/O lines requires the I/O pins to be configured.

### Related Links

[PORT - I/O Pin Controller](#)

### 17.5.2 Power Management

The GCLK can operate in sleep modes, if required. Refer to the sleep mode description in the Power Manager (PM) section.

#### Related Links

[PM – Power Manager](#)

### 17.5.3 Clocks

The GCLK 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 of PM – Power Manager.

#### Related Links

[PM – Power Manager](#)

### 17.5.4 DMA

Not applicable.

### 17.5.5 Interrupts

Not applicable.

### 17.5.6 Events

Not applicable.

### 17.5.7 Debug Operation

Not applicable.

### 17.5.8 Register Access Protection

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

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

Write-protection does not apply for accesses through an external debugger.

#### Related Links

[PAC - Peripheral Access Controller](#)

### 17.5.9 Analog Connections

Not applicable.

## 17.6 Functional Description

### 17.6.1 Principle of Operation

The GCLK module is comprised of eight Generic Clock Generators (Generators) sourcing m Generic Clock Multiplexers.

A clock source selected as input to a Generator can either be used directly, or it can be prescaled in the Generator. A generator output is used as input to one or more the Generic Clock Multiplexers to provide a peripheral (GCLK\_PERIPHERAL). A generic clock can act as the clock to one or several of peripherals.

## 17.6.2 Basic Operation

### 17.6.2.1 Initialization

Before a Generator is enabled, the corresponding clock source should be enabled. The Peripheral 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 by setting the ID bit group (GENDIV.ID).
  - The division factor must be selected by the DIV bit group (GENDIV.DIV)  
**Note:** Refer to *Generic Clock Generator Division register (GENDIV)* 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 will be selected as the source of the generic clock by the ID bit group (GENCTRL.ID)
  - The Generic Clock generator must be enabled (GENCTRL.GENEN=1)  
**Note:** Refer to *Generic Clock Generator Control register (GENCTRL)* 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 via the ID bit group (CLKCTRL.ID)
  - The Generic Clock Generator used as the source of the generic clock by writing the GEN bit group (CLKCTRL.GEN)  
**Note:** Refer to *Generic Clock Control register (CLKCTRL)* for details.

#### Related Links

[GENDIV](#)

[GENCTRL](#)

[CLKCTRL](#)

### 17.6.2.2 Enabling, Disabling and Resetting

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

The GCLK is reset by setting the Software Reset bit in the Control register (CTRL.SWRST) to 1. All registers in the GCLK will be reset to their initial state, except for Generic Clocks Multiplexer and associated Generators that have their Write Lock bit set to 1 (CLKCTRL.WRTLOCK). For further details, refer to [Configuration Lock](#).

### 17.6.2.3 Generic Clock Generator

Each Generator (GCLK\_GEN) 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] is the only Generator that can be selected as source to other Generators but can not act as source to itself.

Each generator GCLKGEN[x] can be connected to one specific pin GCLK\_IO[x]. 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].

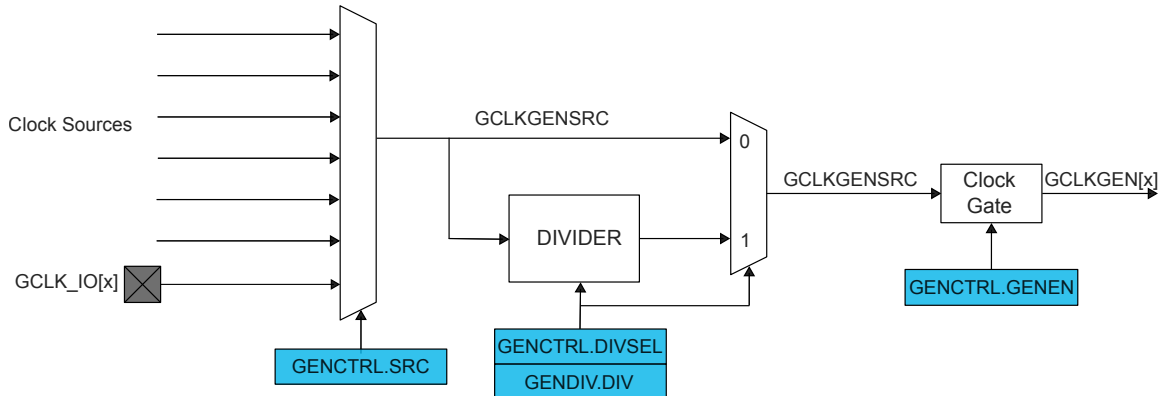
The selected source can be divided. Each Generator can be enabled or disabled independently.

Each GCLKGEN clock signal can then be used as clock source for Generic Clock Multiplexers. Each Generator output 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* for details on the synchronous clock generation.

**Figure 17-3. Generic Clock Generator**



### Related Links

[PM – Power Manager](#)

#### 17.6.2.4 Enabling a Generic Clock Generator

A Generator is enabled by setting the Generic Clock Generator Enable bit in the Generic Clock Generator Control register (GENCTRL.GENEN=1).

#### 17.6.2.5 Disabling a Generic Clock Generator

A Generator is disabled by clearing GENCTRL.GENEN. When GENCTRL.GENEN=0, the GCLKGEN clock is disabled and clock gated.

#### 17.6.2.6 Selecting a Clock Source for the Generic Clock Generator

Each Generator can individually select a clock source by setting the Source Select bit group in GENCTRL (GENCTRL.SRC).

Changing from one clock source, for example A, to another clock source, B, can be done on the fly: If clock source B is not ready, the 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 operation, the 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). Only GCLKGEN[1] can be used as a common source for all other generators except Generator 1.

#### 17.6.2.7 Changing Clock Frequency

The selected source (GENCLKSRC) for a Generator can be divided by writing a division value in the Division Factor bit group in the Generic Clock Generator Division register (GENDIV.DIV). How the actual division factor is calculated is depending on the Divide Selection bit in GENCTRL (GENCTRL.DIVSEL), it can be interpreted in two ways by the integer divider.

**Note:** The number of DIV bits for each Generator is device dependent.

#### 17.6.2.8 Duty Cycle

When dividing a clock with an odd division factor, the duty-cycle will not be 50/50. Writing the Improve Duty Cycle bit in GENCTRL (GENCTRL.IDC=1) will result in a 50/50 duty cycle.

#### 17.6.2.9 Generic Clock Output on I/O Pins

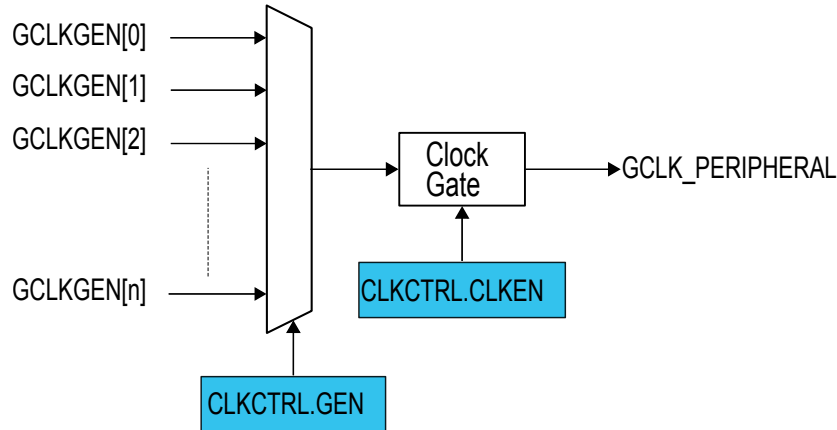
Each Generator's output can be directed to a GCLK\_IO pin. If the Output Enable bit in GENCTRL is '1' (GENCTRL.OE=1) and the Generator is enabled (GENCTRL.GENEN=1), the Generator requests its

clock source and the GCLKGEN clock is output to a GCLK\_IO pin. If GENCTRL.OE=0, 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=1, the output clock will be high when Generator is turned off.

In standby mode, if the clock is output (GENCTRL.OE=1), 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=1, the GCLKGEN clock is kept running and output to GCLK\_IO.

### 17.6.3 Generic Clock

**Figure 17-4. Generic Clock Multiplexer**



#### 17.6.3.1 Enabling a Generic Clock

Before a generic clock is enabled, one of the 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 Generator has been selected, the generic clock is enabled by setting the Clock Enable bit in CLKCTRL (CLKCTRL.CLKEN=1). 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.

#### 17.6.3.2 Disabling a Generic Clock

A generic clock is disabled by writing CLKCTRL.CLKEN=0. The SYNCBUSY bit will be cleared when this write-synchronization is complete. CLKCTRL.CLKEN will stay in its previous state until the synchronization is complete. The generic clock is gated when disabled.

#### 17.6.3.3 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:

1. Write CLKCTRL.CLKEN=0
2. Assert that CLKCTRL.CLKEN reads '0'
3. Change the source of the generic clock by writing CLKCTRL.GEN
4. Re-enable the generic clock by writing CLKCTRL.CLKEN=1

#### 17.6.3.4 Configuration Lock

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



The Generator source of a locked generic clock are also locked, too: The corresponding GENCTRL and GENDIV are locked, and can be unlocked only by a Power Reset.

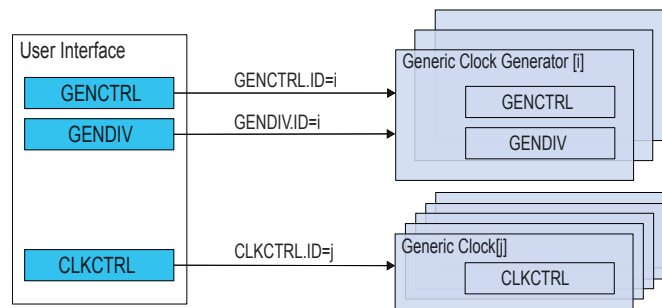
There is one exception concerning the GCLKGEN[0]. As it is used as GCLK\_MAIN, it can not be locked. It is reset by any Reset and will start up in a known configuration. The software reset (CTRL.SWRST) can not unlock the registers.

## 17.6.4 Additional Features

### 17.6.4.1 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 the next figure.

**Figure 17-5. 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:

1. Do an 8-bit write of the *i* value to GENCTRL.ID
2. Read the value of GENCTRL

### 17.6.4.2 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 Generators and generic clocks after Reset is device-dependent.

Refer to *GENCTRL.ID* for details on GENCTRL reset.

Refer to *GENDIV.ID* for details on GENDIV reset.

Refer to *CLKCTRL.ID* for details on CLKCTRL reset.

#### Related Links

[GENDIV](#)

[GENCTRL](#)

[CLKCTRL](#)

## 17.6.5 Sleep Mode Operation

### 17.6.5.1 Sleep Walking

The GCLK module supports the Sleep Walking feature. If the system is in a sleep mode where the Generic Clocks are stopped, a peripheral that needs its clock in order to execute a process must request it from the Generic Clock Controller.

The Generic Clock Controller receives this request, determines which Generic Clock Generator is involved and which clock source needs to be awakened. It then wakes up the respective clock source, enables the Generator and generic clock stages successively, and delivers the clock to the peripheral.

### 17.6.5.2 Run in Standby Mode

In standby mode, the GCLK can continuously output the generator output to GCLK\_IO.

When set, the GCLK can continuously output the generator output to GCLK\_IO.

Refer to [Generic Clock Output on I/O Pins](#) for details.

### 17.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

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=1, 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 are synchronized when written:

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

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

#### Related Links

[Register Synchronization](#)

## 17.7 Register Summary

Offset	Name	Bit Pos.								
0x00	<a href="#">CTRL</a>	7:0								SWRST
0x01	<a href="#">STATUS</a>	7:0	SYNCBUSY							
0x02	<a href="#">CLKCTRL</a>	7:0							ID[5:0]	
		15:8	WRTLOCK	CLKEN					GEN[3:0]	
0x04	<a href="#">GENCTRL</a>	7:0							ID[3:0]	
		15:8							SRC[4:0]	
		23:16			RUNSTDBY	DIVSEL	OE	OOV	IDC	GENEN
		31:24								
0x08	<a href="#">GENDIV</a>	7:0							ID[3:0]	
		15:8						DIV[7:0]		
		23:16						DIV[15:8]		
		31:24								

## 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 require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Refer to [Register Access Protection](#) for details.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Refer to [Synchronization](#) for details.

**17.8.1 Control**

**Name:** CTRL  
**Offset:** 0x0  
**Reset:** 0x00  
**Property:** Write-Protected, Write-Synchronized

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

**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 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 *GENCTRL.ID* for details on GENCTRL reset.

Refer to *GENDIV.ID* for details on GENDIV reset.

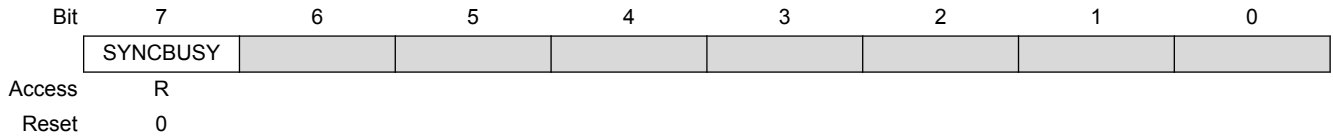
Refer to *CLKCTRL.ID* for details on CLKCTRL reset.

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.

Value	Description
0	There is no reset operation ongoing.
1	There is a reset operation ongoing.

**17.8.2 Status**

**Name:** STATUS  
**Offset:** 0x1  
**Reset:** 0x00  
**Property:** -



**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.

### 17.8.3 Generic Clock Control

**Name:** CLKCTRL  
**Offset:** 0x2  
**Reset:** 0x0000  
**Property:** Write-Protected

Bit	15	14	13	12	11	10	9	8
	WRTLOCK	CLKEN			GEN[3:0]			
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0
Bit	7	6	5	4	3	2	1	0
			ID[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

**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.

Value	Description
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.

Value	Description
0	The generic clock is disabled.
1	The generic clock is enabled.

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

**Table 17-2. Generic Clock Generator**

GEN[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

GEN[3:0]	Name	Description
0x6	GCLKGEN6	Generic clock generator 6
0x7	GCLKGEN7	Generic clock generator 7
0x8	GCLKGEN8	Generic clock generator 8
0x9-0xF	-	Reserved

**Bits 5:0 – ID[5:0]** Generic Clock Selection ID

These bits select the generic clock that will be configured. The value of the ID bit group versus module instance is shown in the table below.

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 the next table.

**Table 17-3. Generic Clock Selection ID and CLKCTRL value after Power Reset**

Module Instance	Reset Value after 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 the table below.

**Table 17-4. Generic Clock Selection ID and CLKCTRL Value after 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	No change

# ATSAMHAXEXXA

## GCLK - Generic Clock Controller

Module Instance	Reset Value after a User Reset		
	CLKCTRL.GEN	CLCTRL.CLKEN	CLKCTRL.WRTLOCK
		Row written to zero If WRTLOCK=1 no change	
Others	0x00 if WRTLOCK=0 No change if WRTLOCK=1	0x00 if WRTLOCK=0 No change if WRTLOCK=1	No change

Value	Name	Description
0x00	GCLK_DFLL48M_REF	DFLL48M Reference
0x01	GCLK_DPLL	FDPLL96M input clock source for reference
0x02	GCLK_DPLL_32K	FDPLL96M 32kHz clock for FDPLL96M internal lock timer
0x03	GCLK_WDT	WDT
0x04	GCLK_RTC	RTC
0x05	GCLK_EIC	EIC
0x06		
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_EVSYS_CHANNEL_6	EVSYS_CHANNEL_6
0x0E	GCLK_EVSYS_CHANNEL_7	EVSYS_CHANNEL_7
0x0F	GCLK_EVSYS_CHANNEL_8	EVSYS_CHANNEL_8
0x10	GCLK_EVSYS_CHANNEL_9	EVSYS_CHANNEL_9
0x11	GCLK_EVSYS_CHANNEL_10	EVSYS_CHANNEL_10
0x12	GCLK_EVSYS_CHANNEL_11	EVSYS_CHANNEL_11
0x13	GCLK_SERCOMx_SLOW	SERCOMx_SLOW
0x14	GCLK_SERCOM0_CORE	SERCOM0_CORE
0x15	GCLK_SERCOM1_CORE	SERCOM1_CORE
0x16	GCLK_SERCOM2_CORE	SERCOM2_CORE
0x17	GCLK_SERCOM3_CORE	SERCOM3_CORE
0x18	GCLK_SERCOM4_CORE	SERCOM4_CORE
0x19	GCLK_SERCOM5_CORE	SERCOM5_CORE
0x1A	GCLK_TCC0, GCLK_TCC1	TCC0,TCC1
0x1B	GCLK_TCC2, GCLK_TC3	TCC2,TC3
0x1C	GCLK_TC4, GCLK_TC5	TC4,TC5
0x1D	GCLK_TC6, GCLK_TC7	TC6,TC7
0x1E	GCLK_ADC	ADC
0x1F	GCLK_AC_DIG	AC_DIG
0x20	-	-
0x21	GCLK_AC_ANA	AC_ANA
0x22	-	-
0x23	GCLK_DAC	DAC
0x24	GCLK_PTC	PTC



# ATSAMHAXEXXA

## GCLK - Generic Clock Controller

Value	Name	Description
0x25	-	Reserved
0x26	-	Reserved
0x27-0x3F	-	Reserved

### 17.8.4 Generic Clock Generator Control

**Name:** GENCTRL  
**Offset:** 0x4  
**Reset:** 0x00000000  
**Property:** Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
			RUNSTDBY	DIVSEL	OE	OOV	IDC	GENEN
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
				SRC[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
					ID[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### 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.

Value	Description
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.

Value	Description
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.

Value	Description
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.

Value	Description
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.

Value	Description
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.

Value	Description
0	The generic clock generator is disabled.
1	The generic clock generator is enabled.

### Bits 12:8 – SRC[4:0] Source Select

These bits define the clock source to be used as the source for the generic clock generator, as shown in the table below.

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

### 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 the next table.

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 the next table.

GCLK Generator ID	Reset Value after a Power Reset
0x00	0x00010600
0x01	0x00000001
0x02	0x00010302
0x03	0x00000003
0x04	0x00000004
0x05	0x00000005
0x06	0x00000006
0x07	0x00000007
0x08	0x00000008

After a user reset, the reset value of the GENCTRL register is as shown in the table below.

GCLK Generator ID	Reset Value after a User Reset
0x00	0x00010600
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	0x00010302 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

# ATSAMHAXEXXA

## GCLK - Generic Clock Controller

GCLK Generator ID	Reset Value after a User Reset
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
0x06	0x00000006 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
0x07	0x00000007 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
0x08	0x00000008 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

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	GCLKGEN6	Generic clock generator 6
0x7	GCLKGEN7	Generic clock generator 7
0x8	GCLKGEN8	Generic clock generator 8
0x9-0xF	Reserved	

### 17.8.5 Generic Clock Generator Division

**Name:** GENDIV  
**Offset:** 0x8  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	[Greyed out bits 31:24]							
Access								
Reset								
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
	[Greyed out bits 7:4]				ID[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### 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 the next table. Writes to bits above the specified number will be ignored.

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 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 the table below.

Values	Description
0x0	Generic clock generator 0
0x1	Generic clock generator 1
0x2	Generic clock generator 2

Values	Description
0x3	Generic clock generator 3
0x4	Generic clock generator 4
0x5	Generic clock generator 5
0x6	Generic clock generator 6
0x7	Generic clock generator 7
0x8	Generic clock generator 8
0x9-0xF	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 the next table.

GCLK Generator ID	Reset Value after a Power Reset
0x00	0x00000000
0x01	0x00000001
0x02	0x00000002
0x03	0x00000003
0x04	0x00000004
0x05	0x00000005
0x06	0x00000006
0x07	0x00000007
0x08	0x00000008

After a user reset, the reset value of the GENDIV register is as shown in next table.

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

# ATSAMHAXEXXA

## GCLK - Generic Clock Controller

GCLK Generator ID	Reset Value after a User Reset
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
0x06	0x00000006 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
0x07	0x00000007 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
0x08	0x00000008 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



## **18. PM – Power Manager**

### **18.1 Overview**

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

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.

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 are provided in order to fit power consumption requirements. This enables the PM to stop unused modules in order 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 device from a sleep mode to active mode.

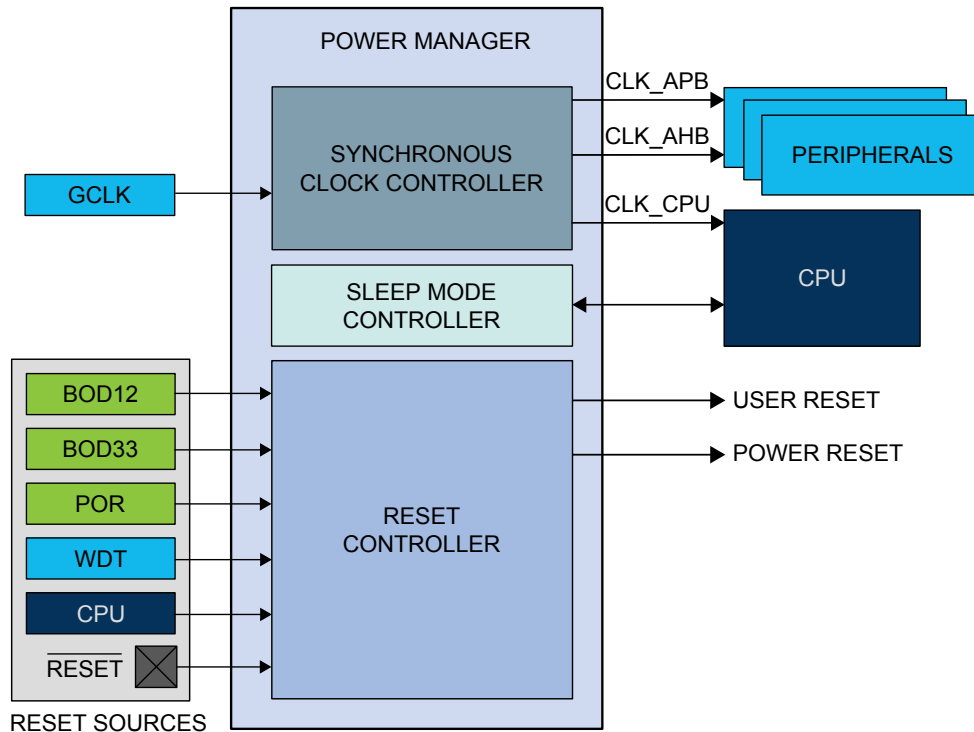
The PM also contains a reset controller to collect all possible reset sources. It issues a device reset and sets the device to its initial state, and allows the reset source to be identified by software.

### **18.2 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
- Power management control
  - Sleep modes: IDLE, STANDBY
  - SleepWalking support on GCLK clocks

### 18.3 Block Diagram

Figure 18-1. PM Block Diagram



### 18.4 Signal Description

Signal Name	Type	Description
RESET	Digital input	External reset

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

#### Related Links

[I/O Multiplexing and Considerations](#)

### 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

Not applicable.

### 18.5.3 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 *Peripheral Clock Default State* table in the *Peripheral Clock Masking* section. 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 reconfigured by the user if needed. Refer to *GCLK – Generic Clock Controller* for details.

#### Related Links

[Peripheral Clock Masking](#)

[GCLK - Generic Clock Controller](#)

#### 18.5.3.1 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.

#### 18.5.3.2 CPU Clock

The CPU clock (CLK\_CPU) is routed to the CPU. Halting the CPU clock inhibits the CPU from executing instructions.

#### 18.5.3.3 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.

#### 18.5.3.4 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.

### 18.5.4 DMA

Not applicable.

### 18.5.5 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* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 18.5.6 Events

Not applicable.

### 18.5.7 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.

### 18.5.8 Register Access Protection

Registers with write-access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag register (INTFLAG).
- Reset Cause register (RCAUSE).

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

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

#### Related Links

[PAC - Peripheral Access Controller](#)

### 18.5.9 Analog Connections

Not applicable.

## 18.6 Functional Description

### 18.6.1 Principle of Operation

#### 18.6.1.1 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 18-4](#)).

#### 18.6.1.2 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.

#### 18.6.1.3 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 18-4](#)).

### 18.6.2 Basic Operation

#### 18.6.2.1 Initialization

After a power-on reset, the PM is enabled and the Reset Cause register indicates the POR source (RCAUSE.POR). 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 18-1](#)).

### 18.6.2.2 Enabling, Disabling and Resetting

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

### 18.6.2.3 Selecting the Main Clock Source

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

#### Related Links

[GCLK - Generic Clock Controller](#)

### 18.6.2.4 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:

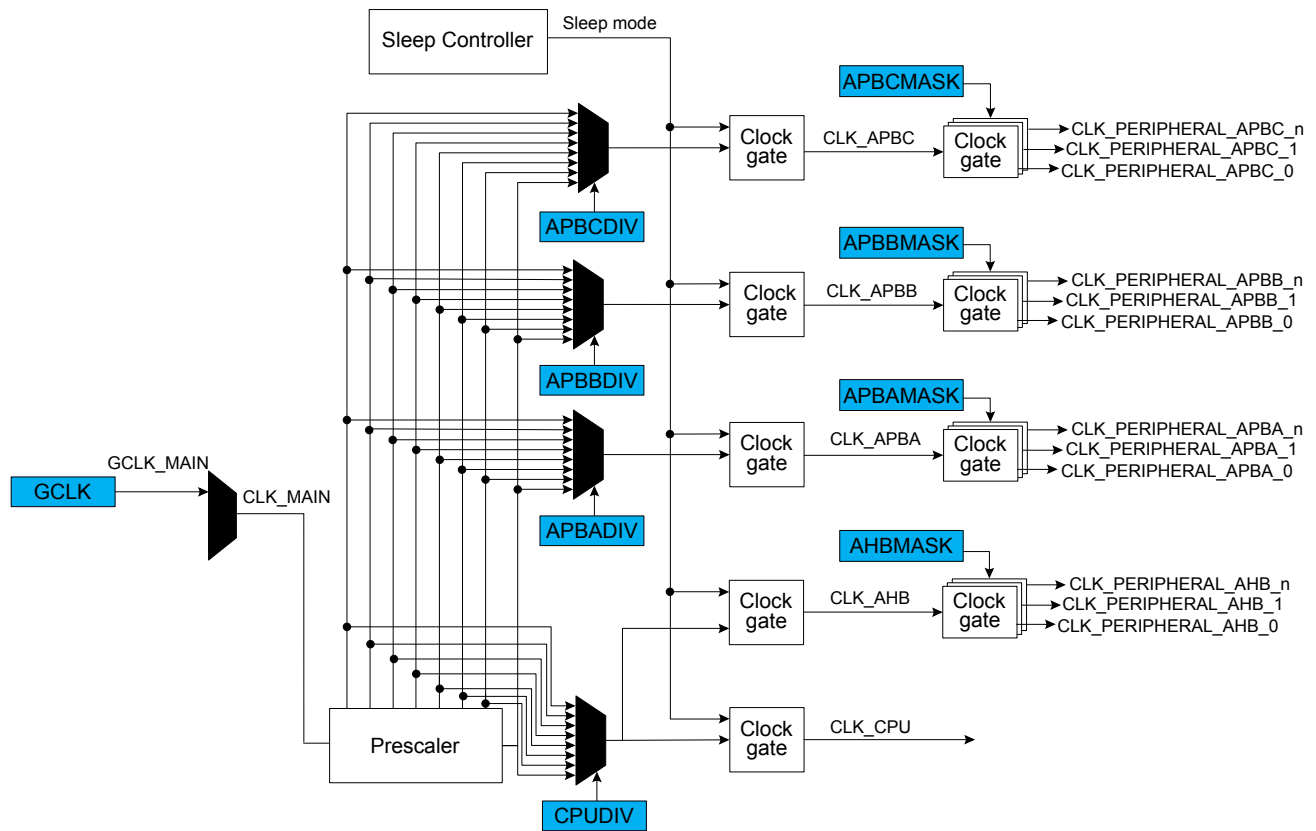
$$f_{\text{CPU}} = \frac{f_{\text{main}}}{2^{\text{CPUDIV}}}$$

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  $f_{\text{CPU}} \geq f_{\text{APBx}}$ . Also, frequencies must never exceed the specified maximum frequency for each clock domain.

**Note:** 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 18-2. Synchronous Clock Selection and Prescaler**



### 18.6.2.5 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.

### 18.6.2.6 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) to zero or one. Refer to the table below for the default state of each of the peripheral clocks.

**Table 18-1. 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

Peripheral Clock	Default State
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_AC_APB	Disabled
CLK_DAC_APB	Disabled
CLK_PTC_APB	Disabled
CLK_DMAC_APB	Enabled
CLK_TCCx_APB	Disabled

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:** 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.

### 18.6.2.7 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 18-2. 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

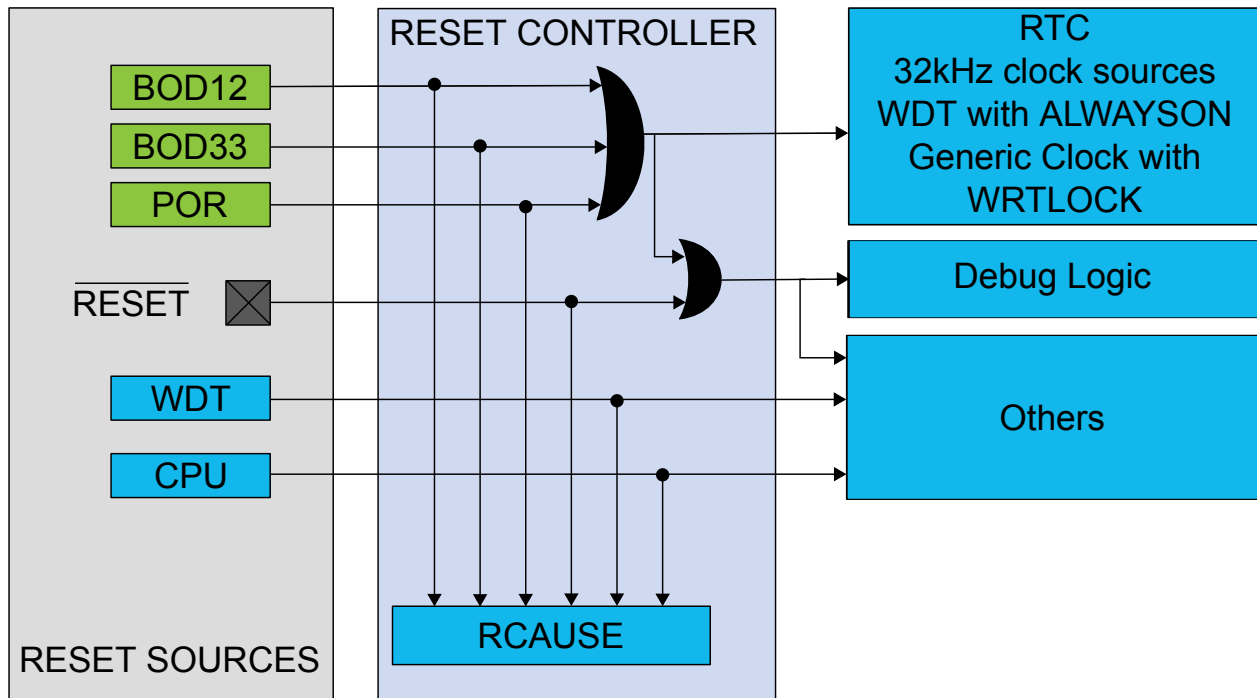
The external reset is generated when pulling the  $\overline{\text{RESET}}$  pin low. This pin has an internal pull-up, and does not need to be driven externally during normal operation.

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 18-3. Reset Controller**





### 18.6.2.8 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 18-3. Sleep Mode Entry and Exit Table**

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

**Note:**

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

**Table 18-4. 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.

### 18.6.3 SleepWalking

SleepWalking is the capability for a device to temporarily wake-up clocks for the 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 awakened by an interrupt (from a peripheral involved in SleepWalking) or enter into STANDBY sleep mode again.

In this device, SleepWalking is supported only on GCLK clocks by using the on-demand clock principle of the clock sources. Refer to *On-demand, Clock Requests* for more details.

#### Related Links

[On-demand, Clock Requests](#)

### 18.6.4 DMA Operation

Not applicable.

### 18.6.5 Interrupts

The peripheral has the following interrupt sources:

- Clock Ready flag

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* 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.

### Related Links

[Nested Vector Interrupt Controller](#)

#### 18.6.6 Events

Not applicable.

#### 18.6.7 Sleep Mode Operation

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

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

## 18.7 Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0								
0x01	SLEEP	7:0							IDLE[1:0]	
0x02	Reserved									
...										
0x07										
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	Reserved									
...										
0x13										
0x14		AHBMASK	7:0			DMAC	NVMCTRL	DSU	HPB2	HPB1
	15:8									
	23:16									
	31:24									
0x18	APBAMASK	7:0	EIC	RTC	WDT	GCLK	SYSCTRL	PM	PAC0	
		15:8								
		23:16								
		31:24								
0x1C	APBBMASK	7:0			DMAC	PORT	NVMCTRL	DSU	PAC1	
		15:8								
		23:16								
		31:24								
0x20	APBCMASK	7:0	SERCOM5	SERCOM4	SERCOM3	SERCOM2	SERCOM1	SERCOM0	EVSYS	PAC2
		15:8	TC7	TC6	TC5	TC4	TC3	TCC2	TCC1	TCC0
		23:16					PTC	DAC	AC	ADC
		31:24								
0x24	Reserved									
...										
0x33										
0x34		INTENCLR	7:0							
0x35	INTENSET	7:0								CKRDY
0x36	INTFLAG	7:0								CKRDY
0x37	Reserved									
0x38	RCAUSE	7:0		SYST	WDT	EXT		BOD33	BOD12	POR

## 18.8 Register Description

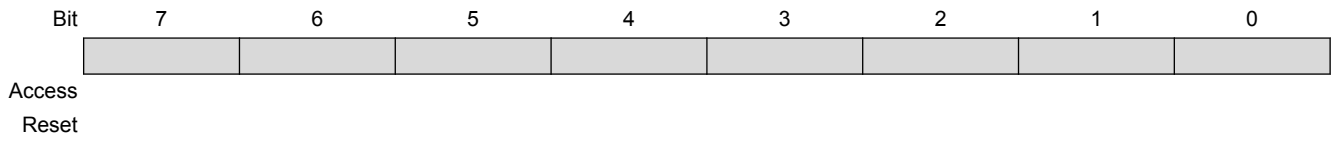
Registers can be 8, 16, or 32 bits wide. Atomic 8-, 16-, and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, and the 8-bit halves of a 16-bit register can be accessed directly.

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

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

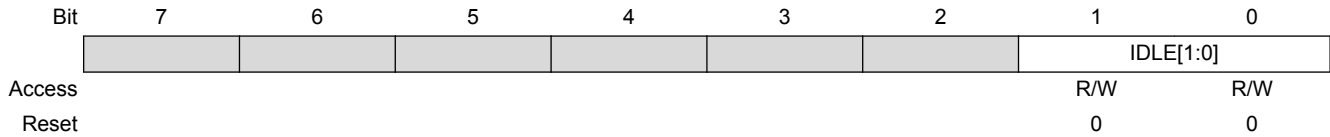
**18.8.1 Control**

**Name:** CTRL  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** Write-Protected



**18.8.2 Sleep Mode**

**Name:** SLEEP  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** Write-Protected



**Bits 1:0 – IDLE[1:0] Idle Mode Configuration**

These bits select the Idle mode configuration after a WFI instruction.

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

### 18.8.3 CPU Clock Select

**Name:** CPUSEL  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** Write-Protected

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

**Bits 2:0 – CPUDIV[2:0]** CPU Prescaler Selection

These bits define the division ratio of the main clock prescaler ( $2^n$ ).

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



**18.8.4 APBA Clock Select**

**Name:** APBASEL  
**Offset:** 0x09  
**Reset:** 0x00  
**Property:** Write-Protected

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

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

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

**18.8.5 APBB Clock Select**

**Name:** APBBSEL  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** Write-Protected

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

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

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

**18.8.6 APBC Clock Select**

**Name:** APBCSEL  
**Offset:** 0x0B  
**Reset:** 0x00  
**Property:** Write-Protected

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

**Bits 2:0 – APBCDIV[2:0] APBC Prescaler Selection**

These bits define the division ratio of the APBC clock prescaler ( $2^n$ ).

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

### 18.8.7 AHB Mask

**Name:** AHBMASK  
**Offset:** 0x14  
**Reset:** 0x0000007F  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24									
	<table border="1" style="width: 100%; height: 15px; background-color: #cccccc;"> <tr> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> </tr> </table>																
Access																	
Reset																	
Bit	23	22	21	20	19	18	17	16									
	<table border="1" style="width: 100%; height: 15px; background-color: #cccccc;"> <tr> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> </tr> </table>																
Access																	
Reset																	
Bit	15	14	13	12	11	10	9	8									
	<table border="1" style="width: 100%; height: 15px; background-color: #cccccc;"> <tr> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> </tr> </table>																
Access																	
Reset																	
Bit	7	6	5	4	3	2	1	0									
	<table border="1" style="width: 100%; height: 15px; background-color: #cccccc;"> <tr> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%; text-align: center;">DMAC</td> <td style="width: 12.5%; text-align: center;">NVMCTRL</td> <td style="width: 12.5%; text-align: center;">DSU</td> <td style="width: 12.5%; text-align: center;">HPB2</td> <td style="width: 12.5%; text-align: center;">HPB1</td> <td style="width: 12.5%; text-align: center;">HPB0</td> <td style="width: 12.5%;"></td> </tr> </table>										DMAC	NVMCTRL	DSU	HPB2	HPB1	HPB0	
		DMAC	NVMCTRL	DSU	HPB2	HPB1	HPB0										
Access			R/W	R/W	R/W	R/W	R/W	R/W									
Reset			1	1	1	1	1	1									

#### Bit 5 – DMAC DMAC AHB Clock Mask

Value	Description
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

Value	Description
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

Value	Description
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

Value	Description
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

Value	Description
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

Value	Description
0	The AHB clock for the HPB0 is stopped.
1	The AHB clock for the HPB0 is enabled.

### 18.8.8 APBA Mask

**Name:** APBAMASK  
**Offset:** 0x18  
**Reset:** 0x0000007F  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24									
	<table border="1" style="width: 100%; height: 15px; border-collapse: collapse;"> <tr> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> </tr> </table>																
Access																	
Reset																	
Bit	23	22	21	20	19	18	17	16									
	<table border="1" style="width: 100%; height: 15px; border-collapse: collapse;"> <tr> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> </tr> </table>																
Access																	
Reset																	
Bit	15	14	13	12	11	10	9	8									
	<table border="1" style="width: 100%; height: 15px; border-collapse: collapse;"> <tr> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> <td style="width: 12.5%;"></td> </tr> </table>																
Access																	
Reset																	
Bit	7	6	5	4	3	2	1	0									
	<table border="1" style="width: 100%; height: 15px; border-collapse: collapse;"> <tr> <td style="width: 12.5%;"></td> <td style="width: 12.5%; text-align: center;">EIC</td> <td style="width: 12.5%; text-align: center;">RTC</td> <td style="width: 12.5%; text-align: center;">WDT</td> <td style="width: 12.5%; text-align: center;">GCLK</td> <td style="width: 12.5%; text-align: center;">SYSCTRL</td> <td style="width: 12.5%; text-align: center;">PM</td> <td style="width: 12.5%; text-align: center;">PAC0</td> <td style="width: 12.5%;"></td> </tr> </table>									EIC	RTC	WDT	GCLK	SYSCTRL	PM	PAC0	
	EIC	RTC	WDT	GCLK	SYSCTRL	PM	PAC0										
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W									
Reset		1	1	1	1	1	1	1									

#### Bit 6 – EIC EIC APB Clock Enable

Value	Description
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

Value	Description
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

Value	Description
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

Value	Description
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

Value	Description
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

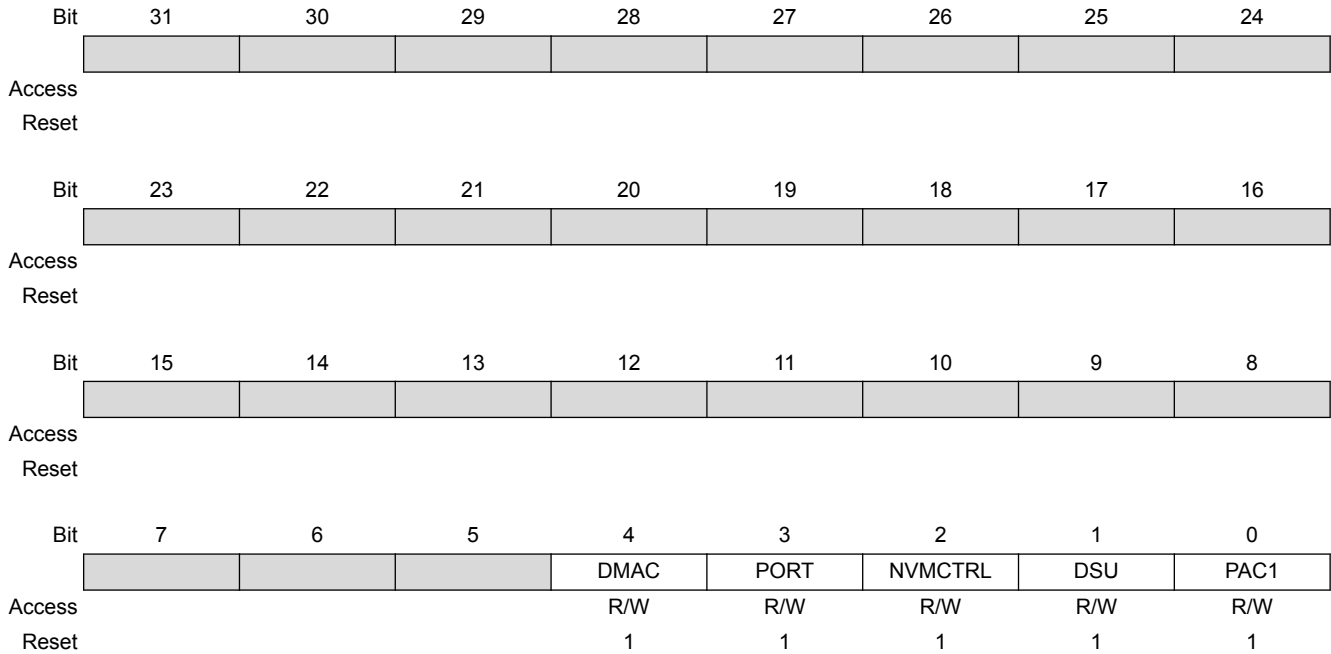
Value	Description
0	The APBA clock for the PM is stopped.
1	The APBA clock for the PM is enabled.

**Bit 0 – PAC0** PAC0 APB Clock Enable

Value	Description
0	The APBA clock for the PAC0 is stopped.
1	The APBA clock for the PAC0 is enabled.

### 18.8.9 APBB Mask

**Name:** APBBMASK  
**Offset:** 0x1C  
**Reset:** 0x0000007F  
**Property:** Write-Protected



#### Bit 4 – DMAC DMAC APB Clock Enable

Value	Description
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

Value	Description
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

Value	Description
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

Value	Description
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

Value	Description
0	The APBB clock for the PAC1 is stopped.
1	The APBB clock for the PAC1 is enabled.

### 18.8.10 APBC Mask

**Name:** APBCMASK  
**Offset:** 0x20  
**Reset:** 0x00010000  
**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		
							PTC	DAC	AC	ADC
Access	R	R	R	R	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0	0	1		
Bit	15	14	13	12	11	10	9	8		
	TC7	TC6	TC5	TC4	TC3	TCC2	TCC1	TCC0		
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		
	SERCOM5	SERCOM4	SERCOM3	SERCOM2	SERCOM1	SERCOM0	EVSYS	PAC2		
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 19 – PTC PTC APB Clock Enable

Value	Description
0	The APBC clock for the PTC is stopped.
1	The APBC clock for the PTC is enabled.

#### Bit 18 – DAC DAC APB Clock Enable

Value	Description
0	The APBC clock for the DAC is stopped.
1	The APBC clock for the DAC is enabled.

#### Bit 17 – AC AC APB Clock Enable

Value	Description
0	The APBC clock for the AC is stopped.
1	The APBC clock for the AC is enabled.

#### Bit 16 – ADC ADC APB Clock Enable

Value	Description
0	The APBC clock for the ADC is stopped.
1	The APBC clock for the ADC is enabled.

**Bit 15 – TC7** TC7 APB Clock Enable

Value	Description
0	The APBC clock for the TC7 is stopped.
1	The APBC clock for the TC7 is enabled.

**Bit 14 – TC6** TC6 APB Clock Enable

Value	Description
0	The APBC clock for the TC6 is stopped.
1	The APBC clock for the TC6 is enabled.

**Bit 13 – TC5** TC5 APB Clock Enable

Value	Description
0	The APBC clock for the TC5 is stopped.
1	The APBC clock for the TC5 is enabled.

**Bit 12 – TC4** TC4 APB Clock Enable

Value	Description
0	The APBC clock for the TC4 is stopped.
1	The APBC clock for the TC4 is enabled.

**Bit 11 – TC3** TC3 APB Clock Enable

Value	Description
0	The APBC clock for the TC3 is stopped.
1	The APBC clock for the TC3 is enabled.

**Bit 10 – TCC2** TCC2 APB Clock Enable

Value	Description
0	The APBC clock for the TCC2 is stopped.
1	The APBC clock for the TCC2 is enabled.

**Bit 9 – TCC1** TCC1 APB Clock Enable

Value	Description
0	The APBC clock for the TCC1 is stopped.
1	The APBC clock for the TCC1 is enabled.

**Bit 8 – TCC0** TCC0 APB Clock Enable

Value	Description
0	The APBC clock for the TCC0 is stopped.
1	The APBC clock for the TCC0 is enabled.

**Bit 7 – SERCOM5** SERCOM5 APB Clock Enable

Value	Description
0	The APBC clock for the SERCOM5 is stopped.
1	The APBC clock for the SERCOM5 is enabled.

---

**Bit 6 – SERCOM4** SERCOM4 APB Clock Enable

Value	Description
0	The APBC clock for the SERCOM4 is stopped.
1	The APBC clock for the SERCOM4 is enabled.

**Bit 5 – SERCOM3** SERCOM3 APB Clock Enable

Value	Description
0	The APBC clock for the SERCOM3 is stopped.
1	The APBC clock for the SERCOM3 is enabled.

**Bit 4 – SERCOM2** SERCOM2 APB Clock Enable

Value	Description
0	The APBC clock for the SERCOM2 is stopped.
1	The APBC clock for the SERCOM2 is enabled.

**Bit 3 – SERCOM1** SERCOM1 APB Clock Enable

Value	Description
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

Value	Description
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

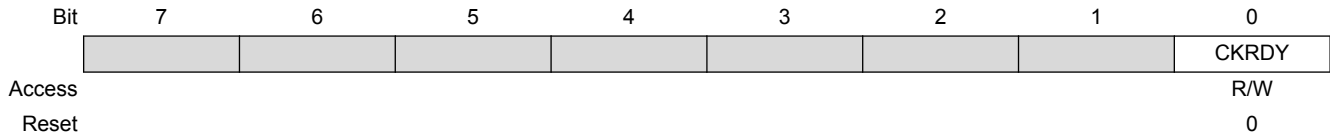
Value	Description
0	The APBC clock for the EVSYS is stopped.
1	The APBC clock for the EVSYS is enabled.

**Bit 0 – PAC2** PAC2 APB Clock Enable

Value	Description
0	The APBC clock for the PAC2 is stopped.
1	The APBC clock for the PAC2 is enabled.

**18.8.11 Interrupt Enable Clear**

**Name:** INTENCLR  
**Offset:** 0x34  
**Reset:** 0x00  
**Property:** Write-Protected



**Bit 0 – CKRDY** Clock Ready Interrupt Enable

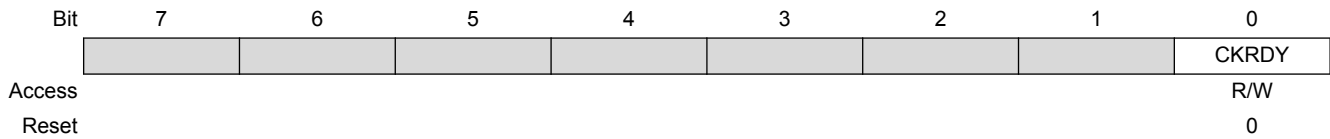
Writing a zero to this bit has no effect.

Writing a one to this bit will clear the Clock Ready Interrupt Enable bit and the corresponding interrupt request.

Value	Description
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.

**18.8.12 Interrupt Enable Set**

**Name:** INTENSET  
**Offset:** 0x35  
**Reset:** 0x00  
**Property:** Write-Protected



**Bit 0 – CKRDY** Clock Ready Interrupt Enable

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.

Value	Description
0	The Clock Ready interrupt is disabled.
1	The Clock Ready interrupt is enabled.

**18.8.13 Interrupt Flag Status and Clear**

**Name:** INTFLAG  
**Offset:** 0x36  
**Reset:** 0x00  
**Property:** -

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

**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.

**18.8.14 Reset Cause**

**Name:** RCAUSE  
**Offset:** 0x38  
**Reset:** 0x01  
**Property:** -

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

**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 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.



## 19. SYSCTRL – System Controller

### 19.1 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). 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.

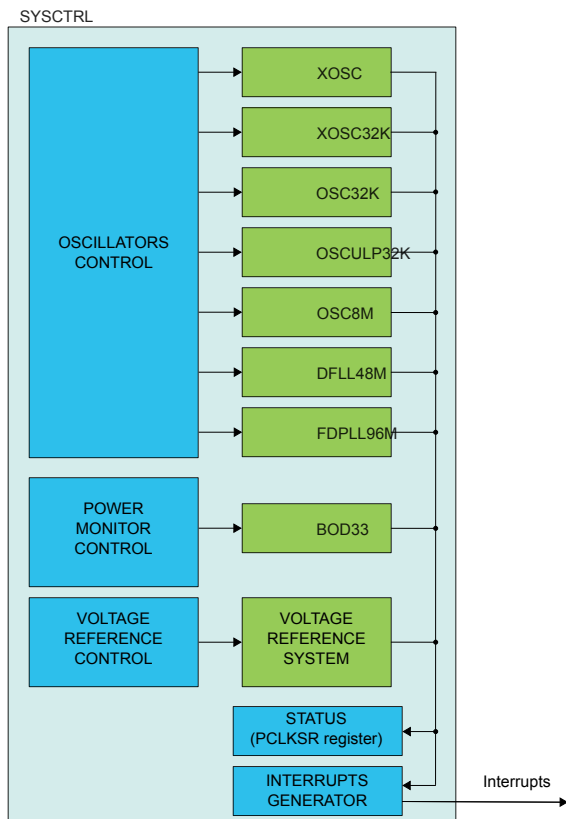
### 19.2 Features

- 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
- Internal Voltage Regulator system (VREG)
  - Operating modes:
    - Normal mode
    - Low-power mode
  - With an internal non-configurable Brown-out detector (BOD12)
- 1.2V Brown-Out Detector (BOD12)
  - Programmable threshold
  - Threshold value loaded from Flash User Calibration at start-up
  - 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
  - Bandgap calibration value loaded from Flash Factory Calibration at start-up

### 19.3 Block Diagram

Figure 19-1. SYSCTRL Block Diagram



### 19.4 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. Refer to *Oscillator Pinout*.

**Related Links**

[I/O Multiplexing and Considerations](#)

### 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

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

### 19.5.2 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 the different sleep modes.

#### Related Links

[PM – Power Manager](#)

### 19.5.3 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, DFLL48M and FDPPLL96M.

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.

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](#) for further details.

#### Related Links

[Peripheral Clock Masking](#)

### 19.5.4 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* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 19.5.5 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 and BOD12 resets will be masked. The BOD resets keep running under hot-plugging. This allows to correct a BOD33 user level too high for the available supply.

### 19.5.6 Register Access Protection

Registers with write-access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag Status and Clear register (INTFLAG)

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

Write-protection does not apply for accesses through an external debugger.

### 19.5.7 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* for details.

#### Related Links

[Electrical Characteristics](#)

## 19.6 Functional Description

### 19.6.1 Principle of Operation

XOSC, XOSC32K, OSC32K, OSCULP32K, OSC8M, DFLL48M, FDPLL96M, BOD33, BOD12, 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 the table below.

**Table 19-1. 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, 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.

The next table shows the behavior in the different sleep modes, depending on the settings of ONDEMAND and RUNSTDBY.

**Table 19-2. 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:** This does not apply to the OSCULP32K oscillator, which is always running and cannot be disabled.

### 19.6.2 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*.

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 start-up 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 start-up: Wait for the OSCxRDY bit in SYSCTRL.PCLKSR register to be set before going into standby mode.

#### Related Links

[GCLK - Generic Clock Controller](#)

### 19.6.3 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*.

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 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. XOSC32K.EN32K is 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 start-up: Wait for the OSCxRDY bit in SYSCTRL.PCLKSR register to be set before going into standby mode.

#### Related Links

[GCLK - Generic Clock Controller](#)

### 19.6.4 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*.

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 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 Area Mapping*). 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.

#### Related Links

[GCLK - Generic Clock Controller](#)

[NVM Software Calibration Area Mapping](#)

### 19.6.5 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*.

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*. The calibration value can be overridden by the user by writing to OSCULP32K.CALIB.

**Related Links**

[Electrical Characteristics](#)

[GCLK - Generic Clock Controller](#)

### 19.6.6 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*.

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*.

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* for details.

OSC8M is automatically switched off in certain sleep modes to reduce power consumption, as described in the *PM – Power Manager*.

**Related Links**

[PM – Power Manager](#)

[Electrical Characteristics](#)

[GCLK - Generic Clock Controller](#)

### 19.6.7 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*.

**Related Links**

[GCLK - Generic Clock Controller](#)

### 19.6.7.1 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 *NVM Software Calibration Area Mapping* 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.

#### Related Links

[NVM Software Calibration Area Mapping](#)

#### 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 (GCLK\_DFLL48M\_REF). Refer to *GCLK – Generic Clock Controller* 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 DFLLMUL.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.
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 ( $F_{\text{clkdfll48m}}$ ) is given by:

$$F_{\text{clkdfll48m}} = \text{DFLLMUL} \cdot \text{MUL} \times F_{\text{clkdfll48mref}}$$

where  $F_{\text{clkdfll48mref}}$  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 *NVM Software Calibration Area Mapping* 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 *NVM User Row Mapping* 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.

**Related Links**

[GCLK - Generic Clock Controller](#)

[NVM Software Calibration Area Mapping](#)

**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.DFLLLOCKC or INTENSET.DFLLLOCKF 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*.

**Related Links**

[Electrical Characteristics](#)

**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:

$$\text{ERROR} = \frac{\text{DIFF}}{\text{MUL}}$$

**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<sub>MAX</sub>)), 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 217 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.

### 19.6.7.2 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  $F_{\text{clkdfll48m}}$ . 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  $F_{\text{clkdfll48m}}$  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.

### 19.6.8 FDPLL96M – Fractional Digital Phase-Locked Loop Controller (DFLL96M)

#### 19.6.8.1 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 XOSC32K, XOSC and GCLK\_DPLL. The path between XOSC and input multiplexer integrates a clock divider. 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* 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 output clock of the FDPLL96M is CLK\_FDPLL96M. The state of the CLK\_FDPLL96M clock only depends on the FDPLL96M internal control of the final clock gater CG.

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* for details.

**Table 19-3. Generic Clock Input for FDPLL96M**

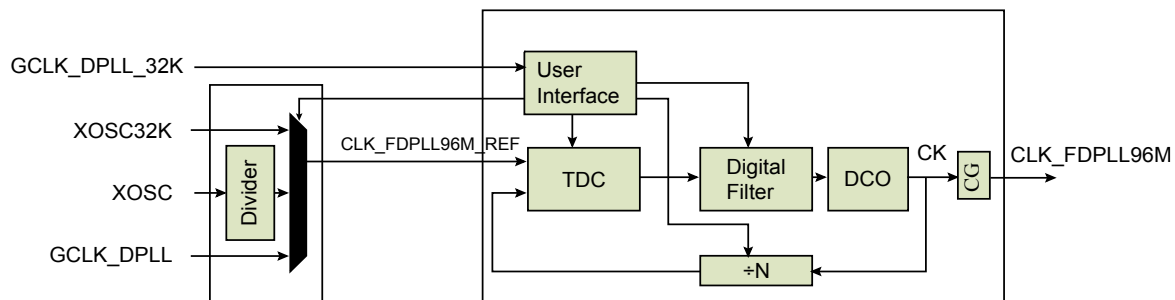
Generic Clock	FDPLL96M
FDPLL96M 32kHz clock	GCLK_DPLL_32K for internal lock timer
FDPLL96M	GCLK_DPLL for CLK_FDPLL96M_REF

**Related Links**

[GCLK - Generic Clock Controller](#)

**19.6.8.2 Block Diagram**

**Figure 19-2. FDPLL96M Block Diagram**



**19.6.8.3 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; XOSC32K, XOSC and GCLK\_DPLL. When the FDPLL96M is enabled, the relationship between the reference clock (CLK\_FDPLL96M\_REF) frequency and the output clock (CLK\_FDPLL96M) frequency is defined below.

$$f_{clk\_fdpll96m} = f_{clk\_fdpll96m\_ref} \times \left( LDR + 1 + \frac{LDRFRAC}{16} \right)$$

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 XOSC and CLK\_FDPLL96M\_REF. The frequency between the two clocks is defined below.

$$f_{clk\_fdpll96m\_ref} = f_{xosc} \times \left( \frac{1}{2 \times (DIV + 1)} \right)$$

When the FDPLL96M is disabled, the output clock is reset. If the loop divider ratio fractional part (DPLLATIO.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  $f_{ckr} = 32\text{kHz}$  and  $f_{ck} = 48\text{MHz}$ , the multiplication ratio is 1500. It means that LDR shall be set to 1499.

Example (fractional mode): assuming  $f_{ckr} = 32\text{kHz}$  and  $f_{ck} = 48.006\text{MHz}$ , the multiplication ratio is 1500.1875 ( $1500 + 3/16$ ). Thus LDR is set to 1499 and LDRFRAC to 3.

#### 19.6.8.4 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 saving 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 CLK\_FDPLL96M.

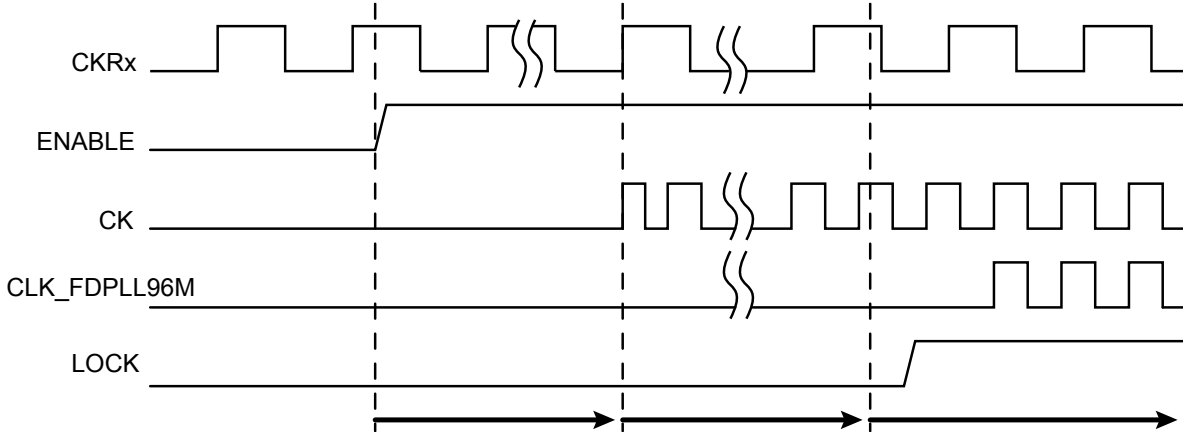
**Table 19-4. CLK\_FDPLL96M behavior from start-up to first edge detection.**

WUF	LTIME	CLK_FDPLL96M 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 (start-up time)

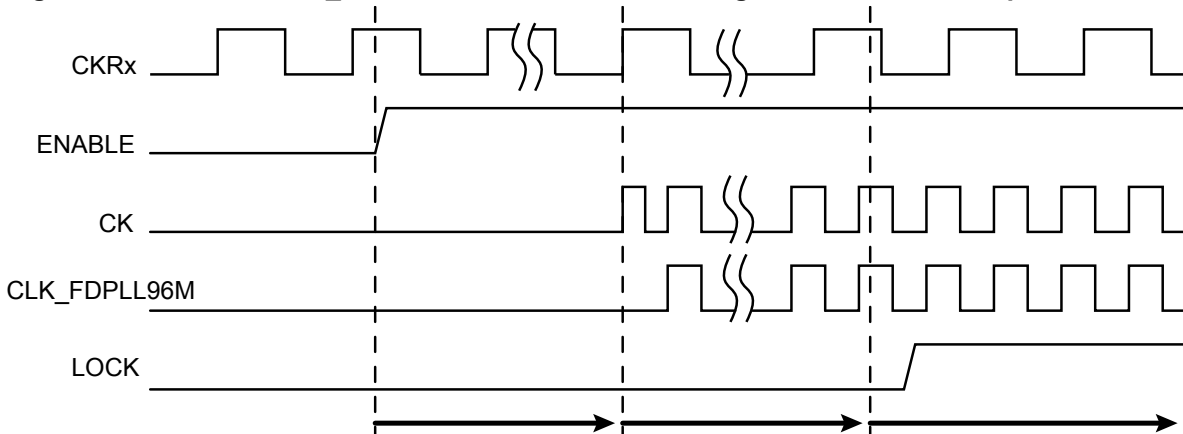
**Table 19-5. CLK\_FDPLL96M behavior after First Edge detection.**

LBYPASS	CLK_FDPLL96M Behavior
0	Normal Mode: the CLK_FDPLL96M is turned off when lock signal is low.
1	Lock Bypass Mode: the CLK_FDPLL96M is always running, lock is irrelevant.

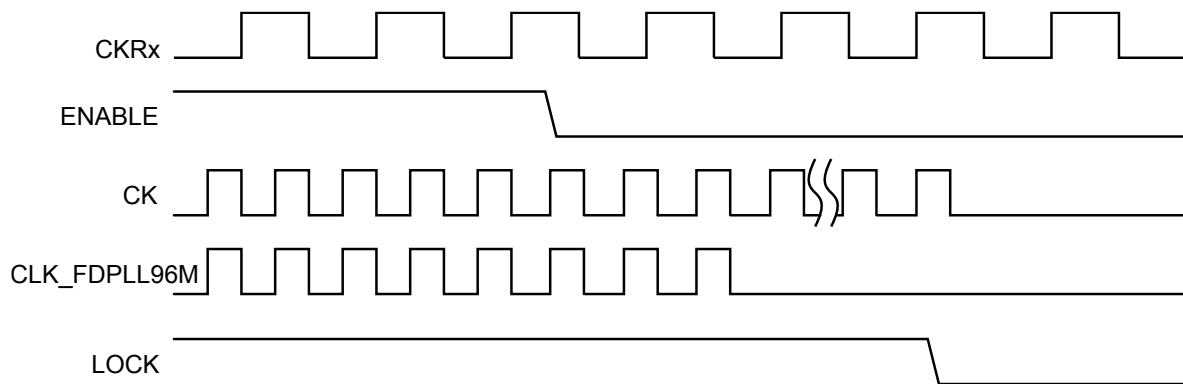
**Figure 19-3. CK and CLK\_FDPLL96M Off Mode to Running Mode**



**Figure 19-4. CK and CLK\_FDPLL96M Off Mode to Running Mode when Wake-Up Fast is Activated**



**Figure 19-5. CK and CLK\_FDPLL96M Running Mode to Off Mode**



**19.6.8.5 Reference Clock Switching**

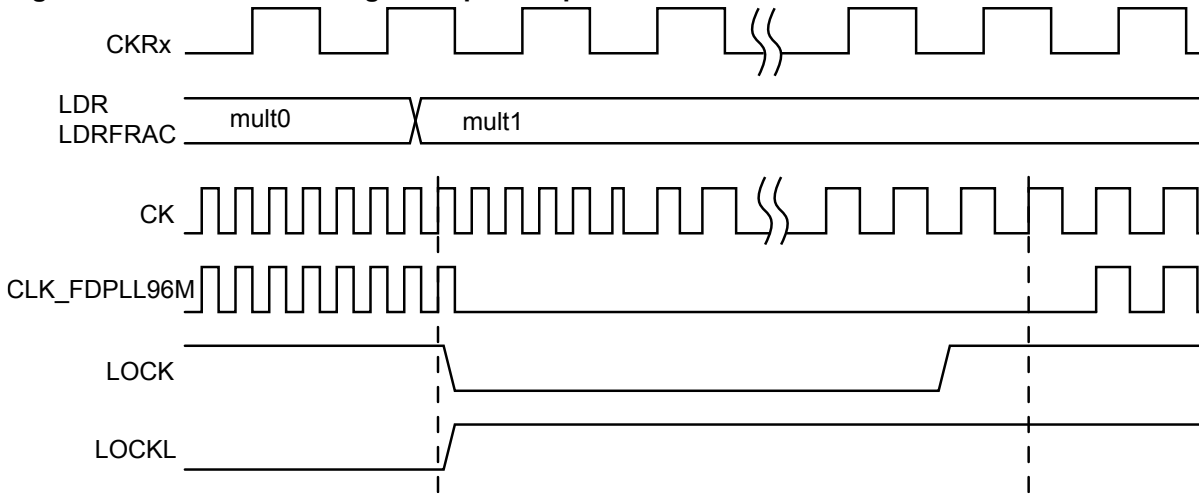
When a software operation requires reference clock switching, the normal operation is to disable the FDPLL96M, modify the DPLLCTRLB.REFCLK to select the desired reference source and activate the FDPLL96M again.

**19.6.8.6 Loop Divider Ratio updates**

The FDPLL96M supports on-the-fly update of the DPLLRTIO 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 19-6. RATIOCTRL Register Update Operation**



**19.6.8.7 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.

**19.6.9 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.

**19.6.9.1 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 start-up or at power-on reset (POR), the BOD33 register values are loaded from the Flash User Row. Refer to *NVM User Row Mapping* for more details.

**19.6.9.2 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.



### 19.6.9.3 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.

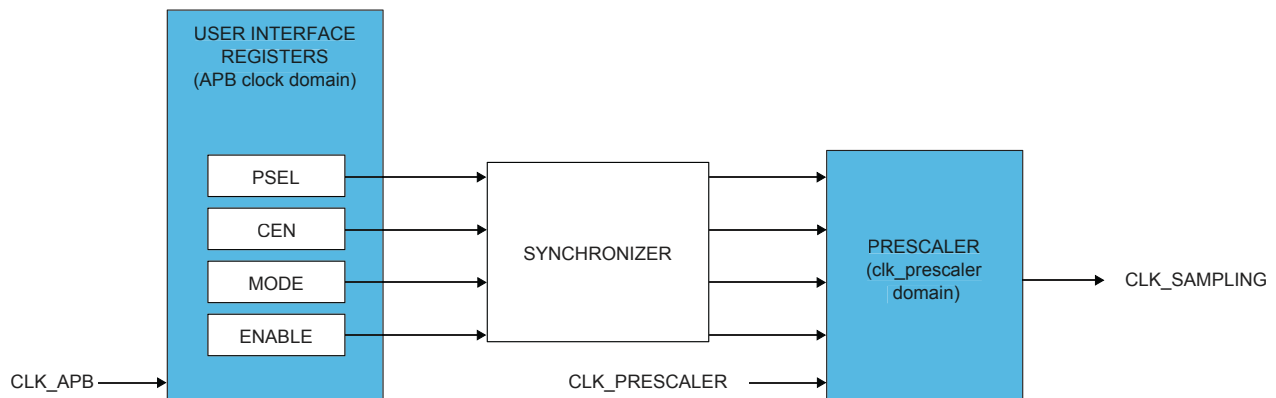
Sampling mode is enabled by writing one to BOD33.MODE for BOD33, and by writing one to BOD12.MODE for BOD12. The frequency of the clock ticks ( $F_{\text{clksampling}}$ ) is controlled by the BOD33 Prescaler Select bit group (BOD33.PSEL) in the BOD33 register and Prescaler Select bit group (BOD12.PSEL) in the BOD12 register for BOD33 and BOD12, respectively.

$$F_{\text{clksampling}} = \frac{F_{\text{clkprescaler}}}{2^{(\text{PSEL}+1)}}$$

The prescaler signal ( $F_{\text{clkprescaler}}$ ) is a 1kHz clock, output from the 32kHz Ultra Low Power Oscillator, OSCULP32K.

As the sampling mode clock is different from the APB clock domain, synchronization among the clocks is necessary. The next figure 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 19-7. 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.

### 19.6.9.4 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.

### 19.6.10 Voltage Reference System Operation

The Voltage Reference System (VREF) consists of a Bandgap Reference Voltage Generator.

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.

**Related Links**

[Electrical Characteristics](#)

**19.6.10.1 User Control of the Voltage Reference System**

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.

**19.6.11 Voltage Regulator System Operation**

The embedded Voltage Regulator (VREG) is an internal voltage regulator that provides the core logic supply (VDDCORE).

**19.6.12 DMA Operation**

Not applicable.

**19.6.13 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. This is an asynchronous interrupt and can be used to wake-up the device from any sleep mode.
- 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* for details. The user must read the INTFLAG register to determine which interrupt condition is present.

**Note:** Interrupts must be globally enabled for interrupt requests to be generated. Refer to *Nested Vector Interrupt Controller* for details.

### Related Links

[Nested Vector Interrupt Controller](#)

### 19.6.14 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.

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.

### 19.7 Register Summary

Offset	Name	Bit Pos.									
0x00	INTENCLR	7:0	DFLLCKC	DFLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY	
		15:8	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS	
		23:16							DPLLLTO	DPLLLCKF	
		31:24									
0x04	INTENSET	7:0	DFLLCKC	DFLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY	
		15:8	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS	
		23:16							DPLLLTO	DPLLLCKF	
		31:24									
0x08	INTFLAG	7:0	DFLLCKC	DFLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY	
		15:8	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS	
		23:16							DPLLLTO	DPLLLCKF	
		31:24									
0x0C	PCLKSR	7:0	DFLLCKC	DFLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY	
		15:8	DPLLLCKR				B33SRDY	BOD33DET	BOD33RDY	DFLLRCS	
		23:16							DPLLLTO	DPLLLCKF	
		31:24									
0x10	XOSC	7:0	ONDEMAND	RUNSTDBY				XTALEN	ENABLE		
		15:8	STARTUP[3:0]				AMPGC	GAIN[2:0]			
0x12	Reserved										
0x13											
0x14	XOSC32K	7:0	ONDEMAND	RUNSTDBY	AAMPEN		EN32K	XTALEN	ENABLE		
		15:8				WRTLOCK		STARTUP[2:0]			
0x16	Reserved										
0x17											
0x18	OSC32K	7:0	ONDEMAND	RUNSTDBY				EN32K	ENABLE		
		15:8				WRTLOCK		STARTUP[2:0]			
		23:16	CALIB[6:0]								
		31:24									
0x1C	OSCULP32K	7:0	WRTLOCK			CALIB[4:0]					
0x1D	Reserved										
0x1F											
0x20	OSC8M	7:0	ONDEMAND	RUNSTDBY					ENABLE		
		15:8							PRESC[1:0]		
		23:16	CALIB[7:0]								
		31:24	FRANGE[1:0]					CALIB[11:8]			
0x24	DFLLCTRL	7:0	ONDEMAND			LLAW	STABLE	MODE	ENABLE		
		15:8					WAITLOCK	BPLCKC	QLDIS	CCDIS	

# ATSAMHAXEXXA

## SYSCTRL – System Controller

Offset	Name	Bit Pos.								
0x26 ... 0x27	Reserved									
0x28	DFLLVAL	7:0	FINE[7:0]							
		15:8	COARSE[5:0]						FINE[9:8]	
		23:16	DIFF[7:0]							
		31:24	DIFF[15:8]							
0x2C	DFLLMUL	7:0	MUL[7:0]							
		15:8	MUL[15:8]							
		23:16	FSTEP[7:0]							
		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	
		15:8	PSEL[3:0]						CEN	MODE
		23:16	LEVEL[5:0]							
		31:24								
0x38 ... 0x3B	Reserved									
0x3C	VREG	7:0		RUNSTDBY						
		15:8			FORCELDO					
0x3E ... 0x3F	Reserved									
0x40	VREF	7:0						BGOUTEN		
		15:8								
		23:16	CALIB[7:0]							
		31:24	CALIB[10:8]							
0x44	DPLLCTRLA	7:0	ONDEMAND					ENABLE		
0x45 ... 0x47	Reserved									
0x48	DPLLRTIO	7:0	LDR[7:0]							
		15:8							LDR[11:8]	
		23:16	LDRFRAC[3:0]							
		31:24								
0x4C	DPLLCTRLB	7:0			REFCLK[1:0]	WUF	LPEN	FILTER[1:0]		
		15:8			LBPASS		LTIME[2:0]			
		23:16	DIV[7:0]							
		31:24	DIV[10:8]							
0x50	DPLLSTATUS	7:0				DIV	ENABLE	CLKRDY	LOCK	

## **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 require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

### 19.8.1 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x00  
**Reset:** 0x00000000  
**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
							DPLLTO	DPLLCKF
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
	DPLLCKR				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
	DFLLCKC	DFLLCKF	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

#### Bit 17 – DPLLTO DPLL Lock Timeout Interrupt Enable

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 Timeout interrupt.

Value	Description
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.

#### Bit 16 – DPLLCKF DPLL Lock Fall Interrupt Enable

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.

Value	Description
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.



**Bit 15 – DPLLLCKR** DPLL Lock Rise Interrupt Enable

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.

Value	Description
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.

**Bit 11 – B33SRDY** BOD33 Synchronization Ready Interrupt Enable

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.

Value	Description
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.

**Bit 10 – BOD33DET** BOD33 Detection Interrupt Enable

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.

Value	Description
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.

**Bit 9 – BOD33RDY** BOD33 Ready Interrupt Enable

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.

Value	Description
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.

**Bit 8 – DFLLRCS** DFLL Reference Clock Stopped Interrupt Enable

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.

Value	Description
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.

**Bit 7 – DFLLCKC** DFLL Lock Coarse Interrupt Enable

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.

Value	Description
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.

**Bit 6 – DFLLCKF** DFLL Lock Fine Interrupt Enable

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.

Value	Description
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.

**Bit 5 – DFLL0OB** DFLL Out Of Bounds Interrupt Enable

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.

Value	Description
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.

**Bit 4 – DFLLRDY** DFLL Ready Interrupt Enable

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.

Value	Description
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.

**Bit 3 – OSC8MRDY** OSC8M Ready Interrupt Enable

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.

Value	Description
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.

**Bit 2 – OSC32KRDY** OSC32K Ready Interrupt Enable

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.

Value	Description
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.

**Bit 1 – XOSC32KRDY** XOSC32K Ready Interrupt Enable

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.

Value	Description
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.

**Bit 0 – XOSCRDY** XOSC Ready Interrupt Enable

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.

Value	Description
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.

### 19.8.2 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x04  
**Reset:** 0x00000000  
**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
							DPLLTO	DPLLCKF
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
	DPLLCKR				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
	DFLLCKC	DFLLCKF	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

**Bit 17 – DPLLTO** DPLL Lock Timeout Interrupt Enable

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.

Value	Description
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.

**Bit 16 – DPLLCKF** DPLL Lock Fall Interrupt Enable

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.

Value	Description
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.

**Bit 15 – DPLLLCKR** DPLL Lock Rise Interrupt Enable

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.

Value	Description
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.

**Bit 11 – B33SRDY** BOD33 Synchronization Ready Interrupt Enable

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.

Value	Description
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.

**Bit 10 – BOD33DET** BOD33 Detection Interrupt Enable

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.

Value	Description
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.

**Bit 9 – BOD33RDY** BOD33 Ready Interrupt Enable

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.

Value	Description
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.

**Bit 8 – DFLLRCS** DFLL Reference Clock Stopped Interrupt Enable

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.

Value	Description
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.

**Bit 7 – DFLLCKC** DFLL Lock Coarse Interrupt Enable

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.

Value	Description
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.

**Bit 6 – DFLLCKF** DFLL Lock Fine Interrupt Enable

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.

Value	Description
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.

**Bit 5 – DFLL0OB** DFLL Out Of Bounds Interrupt Enable

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.

Value	Description
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.

**Bit 4 – DFLLRDY** DFLL Ready Interrupt Enable

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.

Value	Description
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.

**Bit 3 – OSC8MRDY** OSC8M Ready Interrupt Enable

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.

Value	Description
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.

**Bit 2 – OSC32KRDY** OSC32K Ready Interrupt Enable

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.

Value	Description
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.

**Bit 1 – XOSC32KRDY** XOSC32K Ready Interrupt Enable

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.

Value	Description
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.

**Bit 0 – XOSCRDY** XOSC Ready Interrupt Enable

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.

Value	Description
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.

### 19.8.3 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** -

**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.

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
							DPLLLO	DPLLCKF
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
	DPLLCKR				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
	DFLLCKC	DFLLCKF	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

**Bit 17 – DPLLLO** 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.DPLLLO) and will generate an interrupt request if INTENSET.DPLLLO 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 – DPLLCKF** 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.DPLLCKF) and will generate an interrupt request if INTENSET.DPLLCKF 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 – DPLLCKR** 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.

**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.DFLLLCKC) 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 – DFLL0OB** 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.DFLL0OB) and will generate an interrupt request if INTENSET.DFLL0OB 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.

### 19.8.4 Power and Clocks Status

**Name:** PCLKSR  
**Offset:** 0x0C  
**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
							DPLLTO	DPLLCKF
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
	DPLLCKR				B33SRDY	BOD33DET	BOD33RDY	DLLRCS
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
	DFLLCKC	DFLLCKF	DFLLOOB	DFLLRDY	OSC8MRDY	OSC32KRDY	XOSC32KRDY	XOSCRDY
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bit 17 – DPLLTO DPLL Lock Timeout

Value	Description
0	DPLL Lock time-out not detected.
1	DPLL Lock time-out detected.

#### Bit 16 – DPLLCKF DPLL Lock Fall

Value	Description
0	DPLL Lock fall edge not detected.
1	DPLL Lock fall edge detected.

#### Bit 15 – DPLLCKR DPLL Lock Rise

Value	Description
0	DPLL Lock rise edge not detected.
1	DPLL Lock fall edge detected.

#### Bit 11 – B33SRDY BOD33 Synchronization Ready

Value	Description
0	BOD33 synchronization is complete.
1	BOD33 synchronization is ongoing.

**Bit 10 – BOD33DET** BOD33 Detection

Value	Description
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

Value	Description
0	BOD33 is not ready.
1	BOD33 is ready.

**Bit 8 – DFLLRCS** DFLL Reference Clock Stopped

Value	Description
0	DFLL reference clock is running.
1	DFLL reference clock has stopped.

**Bit 7 – DFLLLCKC** DFLL Lock Coarse

Value	Description
0	No DFLL coarse lock detected.
1	DFLL coarse lock detected.

**Bit 6 – DFLLLCKF** DFLL Lock Fine

Value	Description
0	No DFLL fine lock detected.
1	DFLL fine lock detected.

**Bit 5 – DFLLOOB** DFLL Out Of Bounds

Value	Description
0	No DFLL Out Of Bounds detected.
1	DFLL Out Of Bounds detected.

**Bit 4 – DFLLRDY** DFLL Ready

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.

Value	Description
0	The Synchronization is ongoing.
1	The Synchronization is complete.

**Bit 3 – OSC8MRDY** OSC8M Ready

Value	Description
0	OSC8M is not ready.
1	OSC8M is stable and ready to be used as a clock source.

**Bit 2 – OSC32KRDY** OSC32K Ready

---

---

Value	Description
0	OSC32K is not ready.
1	OSC32K is stable and ready to be used as a clock source.

**Bit 1 – XOSC32KRDY** XOSC32K Ready

Value	Description
0	XOSC32K is not ready.
1	XOSC32K is stable and ready to be used as a clock source.

**Bit 0 – XOSCRDY** XOSC Ready

Value	Description
0	XOSC is not ready.
1	XOSC is stable and ready to be used as a clock source.

### 19.8.5 External Multipurpose Crystal Oscillator (XOSC) Control

**Name:** XOSC  
**Offset:** 0x10  
**Reset:** 0x0080  
**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/W	R/W	
Reset	1	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.

STARTUP[3:0]	Number of OSCULP32K Clock Cycles	Number of XOSC Clock Cycles	Approximate Equivalent Time <sup>(1)(2)(3)</sup>
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

**Note:**

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

This bit must be set only after the XOSC has settled, indicated by the XOSC Ready flag in the PCLKSR register (PCLKSR.XOSCRDY).

Value	Description
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.

GAIN[2:0]	Recommended Max Frequency
0x0	2MHz
0x1	4MHz
0x2	8MHz
0x3	16MHz
0x4	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.

Value	Description
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:



Value	Description
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.

**Bit 2 – XTALEN** Crystal Oscillator Enable

This bit controls the connections between the I/O pads and the external clock or crystal oscillator:

Value	Description
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

Value	Description
0	The oscillator is disabled.
1	The oscillator is enabled.

**19.8.6 32kHz External Crystal Oscillator (XOSC32K) Control**

**Name:** XOSC32K  
**Offset:** 0x14  
**Reset:** 0x0080  
**Property:** Write-Protected

	Bit	15	14	13	12	11	10	9	8
					WRTLOCK		STARTUP[2:0]		
Access					R/W		R/W	R/W	R/W
Reset					0		0	0	0
	Bit	7	6	5	4	3	2	1	0
		ONDEMAND	RUNSTDBY	AAMPEN		EN32K	XTALEN	ENABLE	
Access		R/W	R/W	R/W		R/W	R/W	R/W	
Reset		1	0	0		0	0	0	

**Bit 12 – WRTLOCK** Write Lock

This bit locks the XOSC32K register for future writes to fix the XOSC32K configuration.

Value	Description
0	The XOSC32K configuration is not locked.
1	The XOSC32K configuration is locked.

**Bits 10:8 – STARTUP[2:0]** Oscillator Start-Up Time

These bits select the start-up time for the oscillator.

The OSCULP32K oscillator is used to clock the start-up counter.

**Table 19-6. 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) <sup>(1)(2)(3)</sup>
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.

Value	Description
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:

Value	Description
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

Value	Description
0	The automatic amplitude control for the crystal oscillator is disabled.
1	The automatic amplitude control for the crystal oscillator is enabled.

**Bit 3 – EN32K** 32kHz Output Enable

This bit controls the connections between the I/O pads and the external clock or crystal oscillator:

Value	Description
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:

Value	Description
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

# ATSAMHAXEXXA

## SYSCTRL – System Controller

---

---

Value	Description
0	The oscillator is disabled.
1	The oscillator is enabled.

**19.8.7 32kHz Internal Oscillator (OSC32K) Control**

**Name:** OSC32K  
**Offset:** 0x18  
**Reset:** 0x003F0080  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	CALIB[6:0]							
Access	R/W		R/W		R/W		R/W	
Reset	0		1		1		1	
Bit	15	14	13	12	11	10	9	8
			WRTLOCK		STARTUP[2:0]			
Access			R/W		R/W		R/W	
Reset			0		0		0	
Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY					EN32K	ENABLE
Access	R/W		R/W				R/W	
Reset	1		0				0	

**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.

**Bit 12 – WRTLOCK** Write Lock

This bit locks the OSC32K register for future writes to fix the OSC32K configuration.

Value	Description
0	The OSC32K configuration is not locked.
1	The OSC32K configuration is locked.

**Bits 10:8 – STARTUP[2:0]** Oscillator Start-Up Time

These bits select start-up time for the oscillator.

The OSCULP32K oscillator is used as input clock to the startup counter.

**Table 19-7. Start-Up Time for 32kHz Internal Oscillator**

STARTUP[2:0]	Number of OSC32K clock cycles	Approximate Equivalent Time (OSCULP= 32 kHz) <sup>(1)(2)(3)</sup>
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.

Value	Description
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:

Value	Description
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.

**Bit 2 – EN32K** 32kHz Output Enable

---

---

Value	Description
0	The 32kHz output is disabled.
1	The 32kHz output is enabled.
0	The oscillator is disabled.
1	The oscillator is enabled.

**Bit 1 – ENABLE** Oscillator Enable

Value	Description
0	The oscillator is disabled.
1	The oscillator is enabled.

**19.8.8 32kHz Ultra Low Power Internal Oscillator (OSCULP32K) Control**

**Name:** OSCULP32K  
**Offset:** 0x1C  
**Reset:** 0xXX  
**Property:** Write-Protected

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

**Bit 7 – WRTLOCK** Write Lock

This bit locks the OSCULP32K register for future writes to fix the OSCULP32K configuration.

Value	Description
0	The OSCULP32K configuration is not locked.
1	The OSCULP32K configuration is locked.

**Bits 4:0 – CALIB[4:0]** Oscillator Calibration

These bits control the oscillator calibration.

These bits are loaded from Flash Calibration at startup.



**19.8.9 8MHz Internal Oscillator (OSC8M) Control**

**Name:** OSC8M  
**Offset:** 0x20  
**Reset:** 0xFFFF0382  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
	FRANGE[1:0]				CALIB[11:8]			
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	x	x			0	0	0	0
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	0	0	x
Bit	15	14	13	12	11	10	9	8
							PRESC[1:0]	
Access							R/W	R/W
Reset							1	1
Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY					ENABLE	
Access	R/W	R/W					R/W	
Reset	1	0					1	

**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.

FRANGE[1:0]	Description
0x0	4 to 6MHz
0x1	6 to 8MHz
0x2	8 to 11MHz
0x3	11 to 15MHz

**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 9:8 – PRESC[1:0] Oscillator Prescaler**

These bits select the oscillator prescaler factor setting according to the table below.

PRESC[1:0]	Description
0x0	1
0x1	2
0x2	4
0x3	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.

Value	Description
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:

Value	Description
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.

**Bit 1 – ENABLE** Oscillator Enable

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.

Value	Description
0	The oscillator is disabled or being enabled.
1	The oscillator is enabled or being disabled.

**19.8.10 DFLL48M Control**

**Name:** DFLLCTRL  
**Offset:** 0x24  
**Reset:** 0x0080  
**Property:** Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
					WAITLOCK	BPLCKC	QLDIS	CCDIS
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	ONDEMAND			LLAW	STABLE	MODE	ENABLE	
Access	R/W			R/W	R/W	R/W	R/W	
Reset	1			0	0	0	0	

**Bit 11 – WAITLOCK** Wait Lock

This bit controls the DFLL output clock, depending on lock status:

Value	Description
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:

Value	Description
0	Bypass coarse lock is disabled.
1	Bypass coarse lock is enabled.

**Bit 9 – QLDIS** Quick Lock Disable

Value	Description
0	Quick Lock is enabled.
1	Quick Lock is disabled.

**Bit 8 – CCDIS** Chill Cycle Disable

Value	Description
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.

Value	Description
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 4 – LLAW Lose Lock After Wake

Value	Description
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

Value	Description
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

Value	Description
0	The DFLL operates in open-loop operation.
1	The DFLL operates in closed-loop operation.

#### Bit 1 – ENABLE DFLL Enable

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.

Value	Description
0	The DFLL oscillator is disabled.
1	The DFLL oscillator is enabled.

**19.8.11 DFLL48M Value**

**Name:** DFLLVAL  
**Offset:** 0x28  
**Reset:** 0x00000000  
**Property:** Read-Synchronized, 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.

**19.8.12 DFLL48M Multiplier**

**Name:** DFLLMUL  
**Offset:** 0x2C  
**Reset:** 0x00000000  
**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**

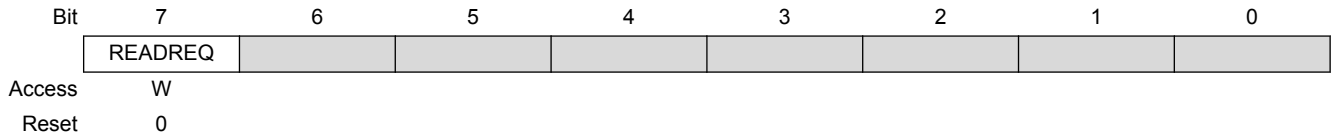
This bit group indicates the maximum step size allowed during fine adjustment in closed-loop mode. When adjusting to a new frequency, the expected output frequency overshoot depends on this step size.

**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.

**19.8.13 DFLL48M Synchronization**

**Name:** DFLLSYNC  
**Offset:** 0x30  
**Reset:** 0x00  
**Property:** Write-Protected



**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.

**19.8.14 3.3V Brown-Out Detector (BOD33) Control**

**Name:** BOD33  
**Offset:** 0x34  
**Reset:** 0x00XX00XX  
**Property:** Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	[Greyed out bits 31-24]							
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	LEVEL[5:0]							
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			x	x	x	x	x	x
Bit	15	14	13	12	11	10	9	8
	PSEL[3:0]				[Greyed out bits 11-10]		CEN	MODE
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0
Bit	7	6	5	4	3	2	1	0
	[Greyed out bit 7]	RUNSTDBY	[Greyed out bit 5]	ACTION[1:0]		HYST	ENABLE	[Greyed out bit 0]
Access		R/W		R/W	R/W	R/W	R/W	
Reset		0		x	x	x	x	

**Bits 21:16 – LEVEL[5:0] BOD33 Threshold Level**

This field sets the triggering voltage threshold for the BOD33. See the *Electrical Characteristics* 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 start-up. Refer to *NVM User Row Mapping* 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.

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



PSEL[3:0]	Name	Description
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

**Bit 9 – CEN** Clock Enable

Writing a zero to this bit will stop the BOD33 sampling clock.

Writing a one to this bit will start the BOD33 sampling clock.

Value	Description
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.

**Bit 8 – MODE** Operation Mode

Value	Description
0	The BOD33 operates in continuous mode.
1	The BOD33 operates in sampling mode.

**Bit 6 – RUNSTDBY** Run in Standby

Value	Description
0	The BOD33 is disabled in standby sleep mode.
1	The BOD33 is enabled in standby sleep mode.

**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 start-up.

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:

This bit is loaded from Flash User Row at start-up. Refer to *NVM User Row Mapping* for more details.

Value	Description
0	No hysteresis.
1	Hysteresis enabled.

### Bit 1 – ENABLE Enable

This bit is loaded from Flash User Row at startup. Refer to *NVM User Row Mapping* for more details.

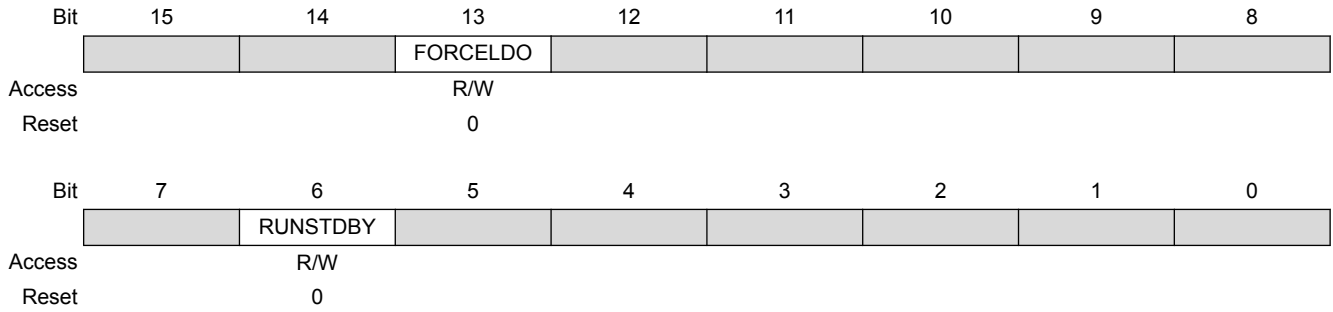
Value	Description
0	BOD33 is disabled.
1	BOD33 is enabled.

### Related Links

[Electrical Characteristics](#)

**19.8.15 Voltage Regulator System (VREG) Control**

**Name:** VREG  
**Offset:** 0x3C  
**Reset:** 0x0X00  
**Property:** Write-Protected



**Bit 13 – FORCELDO** Force LDO Voltage Regulator

Value	Description
0	The voltage regulator is in low power and low drive configuration in standby sleep mode.
1	The voltage regulator is in low power and high drive configuration in standby sleep mode.

**Bit 6 – RUNSTDBY** Run in Standby

Value	Description
0	The voltage regulator is in low power configuration in standby sleep mode.
1	The voltage regulator is in normal configuration in standby sleep mode.

**19.8.16 Voltage References System (VREF) Control**

**Name:** VREF  
**Offset:** 0x40  
**Reset:** 0x0XXX0000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
							CALIB[10:8]	
Access						R/W	R/W	R/W
Reset						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								
Reset								
Bit	7	6	5	4	3	2	1	0
						BGOUTEN		
Access						R/W		
Reset						0		

**Bits 26:16 – CALIB[10:0] Bandgap Voltage Generator Calibration**

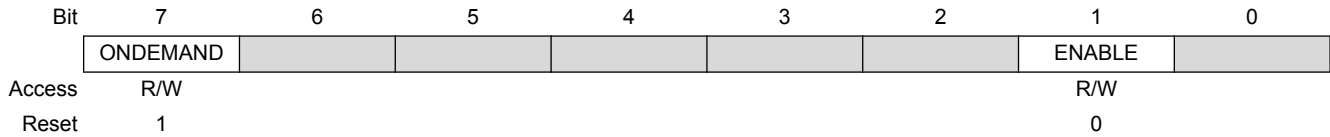
These bits are used to calibrate the output level of the bandgap voltage reference. These bits are loaded from Flash Calibration Row at startup.

**Bit 2 – BGOUTEN Bandgap Output Enable**

Value	Description
0	The bandgap output is not available as an ADC input channel.
1	The bandgap output is routed to an ADC input channel.

**19.8.17 DPLL Control A**

**Name:** DPLLCTRLA  
**Offset:** 0x44  
**Reset:** 0x80  
**Property:** Write-Protected



**Bit 7 – ONDEMAND** On Demand Clock Activation

Value	Description
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.

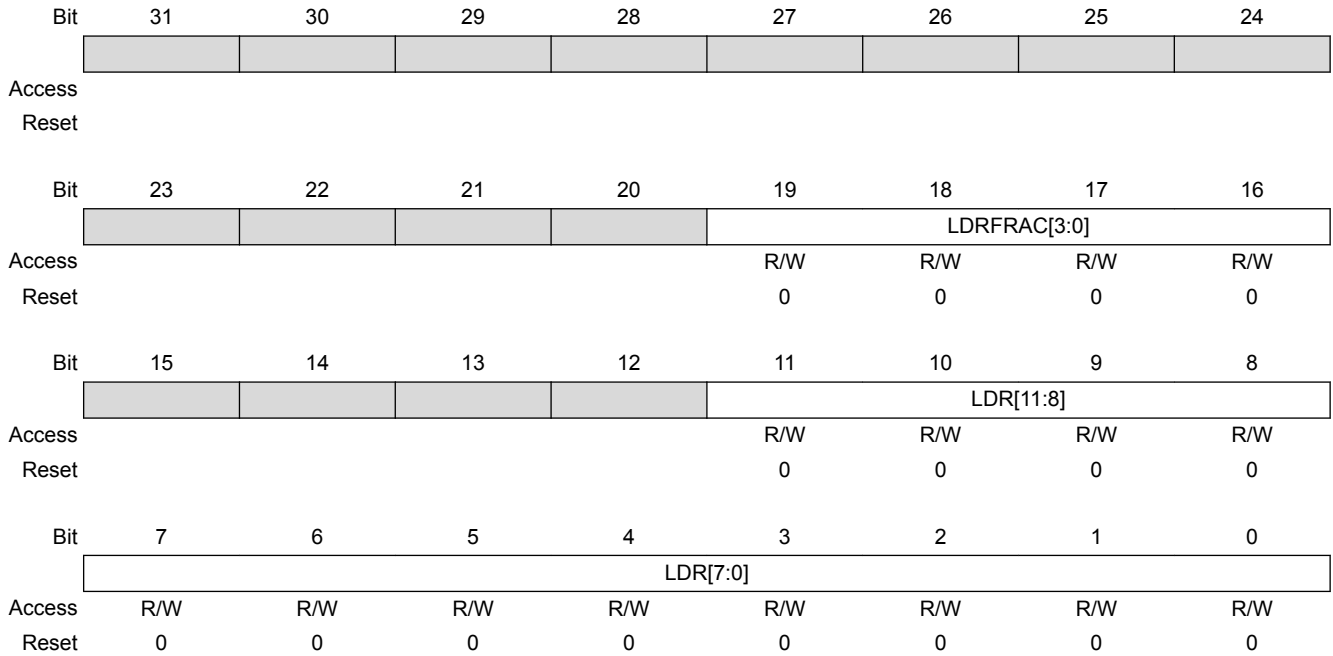
**Bit 1 – ENABLE** DPLL Enable

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.

Value	Description
0	The DPLL is disabled.
1	The DPLL is enabled.

**19.8.18 DPLL Ratio Control**

**Name:** DPLL RATIO  
**Offset:** 0x48  
**Reset:** 0x00000000  
**Property:** Write-Protected



**Bits 19:16 – LDRFRAC[3:0]** Loop Divider Ratio Fractional Part  
Write this field with the fractional part of the frequency multiplier.

**Bits 11:0 – LDR[11:0]** Loop Divider Ratio  
Write this field with the integer part of the frequency multiplier.

**19.8.19 DPLL Control B**

**Name:** DPLLCTRLB  
**Offset:** 0x4C  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
	DIV[10:8]							
Access						R/W	R/W	R/W
Reset						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/W		R/W	R/W	R/W
Reset				0		0	0	0
Bit	7	6	5	4	3	2	1	0
	REFCLK[1:0]		WUF		LPEN		FILTER[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

**Bits 26:16 – DIV[10:0] Clock Divider**

These bits are used to set the XOSC clock source division factor. Refer to [Principle of Operation](#).

**Bit 12 – LBYPASS Lock Bypass**

Value	Description
0	Normal Mode: the CLK_FDPLL96M is turned off when lock signal is low.
1	Lock Bypass Mode: the CLK_FDPLL96M is always running, lock is irrelevant.

**Bits 10:8 – LTIME[2:0] Lock Time**

These bits select Lock Timeout.

LTIME[2:0]	Name	Description
0x0	DEFAULT	No time-out
0x1-0x3		Reserved
0x4	8MS	Time-out if no lock within 8 ms
0x5	9MS	Time-out if no lock within 9 ms
0x6	10MS	Time-out if no lock within 10 ms
0x7	11MS	Time-out if no lock within 11 ms

**Bits 5:4 – REFCLK[1:0]** Reference Clock Selection  
 These bits select the CLK\_FDPLL96M\_REF source.

REFCLK[1:0]	Name	Description
0x0	XOSC32	XOSC32 clock reference
0x1	XOSC	XOSC clock reference
0x2	GCLK_DPLL	GCLK_DPLL clock reference
0x3		Reserved

**Bit 3 – WUF** Wake Up Fast

Value	Description
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

Value	Description
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.

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



**19.8.20 DPLL Status**

**Name:** DPLLSTATUS  
**Offset:** 0x50  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
					DIV	ENABLE	CLKRDY	LOCK
Access					R	R	R	R
Reset					0	0	0	0

**Bit 3 – DIV** Divider Enable

Value	Description
0	The reference clock divider is disabled.
1	The reference clock divider is enabled.

**Bit 2 – ENABLE** DPLL Enable

Value	Description
0	The DPLL is disabled.
1	The DPLL is enabled.

**Bit 1 – CLKRDY** Output Clock Ready

Value	Description
0	The DPLL output clock is off
1	The DPLL output clock in on.

**Bit 0 – LOCK** DPLL Lock Status

Value	Description
0	The DPLL Lock signal is cleared.
1	The DPLL Lock signal is asserted.

## **20. WDT – Watchdog Timer**

### **20.1 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.

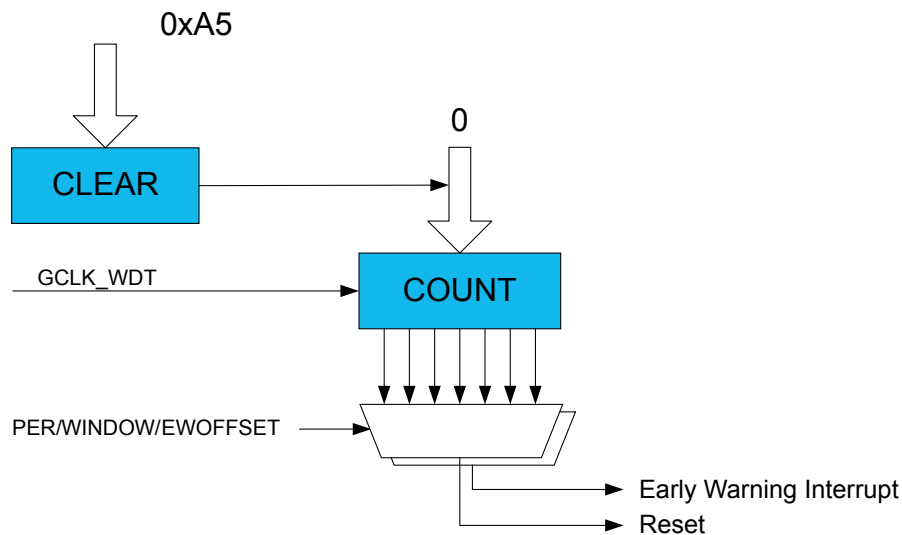
When enabled, the WDT will run in active mode and all sleep modes. It is asynchronous and runs from a CPU-independent clock source. The WDT will continue operation and issue a system reset or interrupt even if the main clocks fail.

### **20.2 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
  - From 16 cycles to 32,000 cycles in window mode
- Always-on capability

## 20.3 Block Diagram

Figure 20-1. WDT Block Diagram



## 20.4 Signal Description

Not applicable.

## 20.5 Product Dependencies

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

### 20.5.1 I/O Lines

Not applicable.

### 20.5.2 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.

#### Related Links

[PM – Power Manager](#)

### 20.5.3 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* 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* 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* for further details.

GCLK\_WDT is intended to be sourced from the clock of the internal ultra-low-power (ULP) oscillator. Due to the ultralow- 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*.

GCLK\_WDT can also be clocked from other sources if a more accurate clock is needed, but at the cost of higher power consumption.

### Related Links

[PM – Power Manager](#)

[GCLK - Generic Clock Controller](#)

[32kHz Ultra Low Power Internal Oscillator \(OSCULP32K\) Operation Synchronization](#)

### 20.5.4 DMA

Not applicable.

### 20.5.5 Interrupts

The interrupt request line is connected to the interrupt controller. Using the WDT interrupt(s) requires the interrupt controller to be configured first.

### Related Links

[Nested Vector Interrupt Controller](#)

### 20.5.6 Events

Not applicable.

### 20.5.7 Debug Operation

When the CPU is halted in debug mode the WDT will halt normal operation.

### 20.5.8 Register Access Protection

Registers with write-access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag Status and Clear register (INTFLAG)

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

### Related Links

[PAC - Peripheral Access Controller](#)

### 20.5.9 Analog Connections

Not applicable.

## 20.6 Functional Description

### 20.6.1 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 set back, or else, a system Reset is issued.

The WDT has two modes of operation, Normal mode and Window mode. Both modes offer the option of Early Warning interrupt generation. The description for each of the basic modes is given below. The settings in the Control register (CTRL) and the Interrupt Enable register (handled by INTENCLR/SET) determine the mode of operation:

**Table 20-1. 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

## 20.6.2 Basic Operation

### 20.6.2.1 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:

- 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=1) 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. The WDT is configured by defining the required Time-Out Period bits in the Configuration register (CONFIG.PER). If window-mode operation is required, the Window Enable bit in the Control register (CTRL.WEN) must be written to one and the Window Period bits in the Configuration register (CONFIG.WINDOW) must be defined.

#### 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).

#### 20.6.2.2 Configurable Reset Values

After a Power-on Reset, some registers will be loaded with initial values from the NVM User Row. Refer to *NVM User Row Mapping* 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*.

#### Related Links

[NVM User Row Mapping](#)

#### 20.6.2.3 Enabling and Disabling

The WDT is enabled by writing a '1' to the Enable bit in the Control register (CTRL.ENABLE). The WDT is disabled by writing a '0' to CTRL.ENABLE.

The WDT can be disabled only if the Always-On bit in the Control register (CTRL.ALWAYSON) is '0'.

#### 20.6.2.4 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 '1' to the Enable bit in the Control register (CTRL.ENABLE). Once enabled, the WDT will issue a system reset if a time-out occurs. This can be prevented by clearing the WDT at any time during the time-out period.

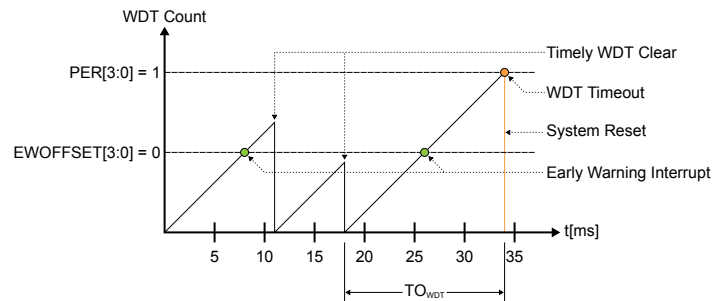
The WDT is cleared and a new WDT time-out period is started by writing 0xA5 to the Clear register (CLEAR). Writing any other value than 0xA5 to CLEAR will issue an immediate system reset.

There are 12 possible WDT time-out ( $TO_{WDT}$ ) periods, selectable from 8ms to 16s.

By default, the early warning interrupt is disabled. If it is desired, the Early Warning Interrupt Enable bit in the Interrupt Enable register (INTENSET.EW) must be written to '1'. The Early Warning Interrupt is disabled again by writing a '1' to the Early Warning Interrupt bit in the Interrupt Enable Clear register

(INTENCLR.EW). If the Early Warning Interrupt is enabled, an interrupt is generated prior to a WDT time-out condition. In Normal mode, the Early Warning Offset bits in the Early Warning Interrupt Control register, EWCTRL.EWOFFSET, define the time when the early warning interrupt occurs. The Normal mode operation is illustrated in the figure Normal-Mode Operation.

**Figure 20-2. Normal-Mode Operation**

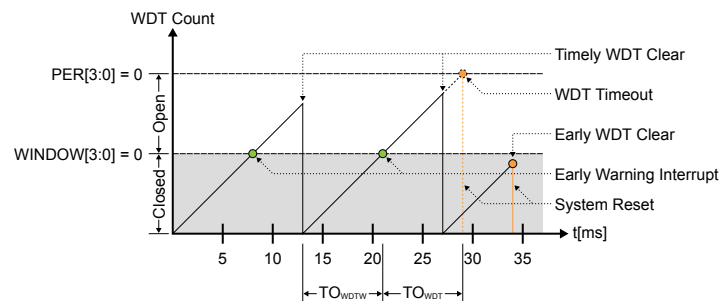


### 20.6.2.5 Window Mode

In Window mode operation, the WDT uses two different time specifications: the WDT can only be cleared by writing 0xA5 to the CLEAR register *after* the closed window time-out period ( $TO_{WDTW}$ ), during the subsequent Normal time-out period ( $TO_{WDT}$ ). If the WDT is cleared before the time window opens (before  $TO_{WDTW}$  is over), the WDT will issue a system reset. Both parameters  $TO_{WDTW}$  and  $TO_{WDT}$  are periods in a range from 8ms to 16s, so the total duration of the WDT time-out period is the sum of the two parameters. The closed window period is defined by the Window Period bits in the Configuration register (CONFIG.WINDOW), and the open window period is defined by the Period bits in the Configuration register (CONFIG.PER).

By default, the Early Warning interrupt is disabled. If it is desired, the Early Warning Interrupt Enable bit in the Interrupt Enable register (INTENSET.EW) must be written to '1'. The Early Warning Interrupt is disabled again by writing a '1' to the Early Warning Interrupt bit in the Interrupt Enable Clear (INTENCLR.EW) register. If the Early Warning interrupt is enabled in Window mode, the interrupt is generated at the start of the open window period, i.e. after  $TO_{WDTW}$ . The Window mode operation is illustrated in figure Window-Mode Operation.

**Figure 20-3. Window-Mode Operation**



## 20.6.3 Additional Features

### 20.6.3.1 Always-On Mode

The Always-On mode is enabled by setting the Always-On bit in the Control register (CTRLA.ALWAYSON=1). 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 (CTRLA.WEN) is allowed while in Always-On mode, but note that CONFIG.PER cannot be changed.

The CTRL.ALWAYSON bit must never be set to one by software if any of the following conditions is true:

1. The GCLK\_WDT is disabled
2. The clock generator for the GCLK\_WDT is disabled
3. The source clock of the clock generator for the GCLK\_WDT is disabled or off

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 20-2. WDT Operating Modes With Always-On**

WEN	Interrupt enable	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

### 20.6.4 Interrupts

The WDT has the following interrupt source:

- Early Warning (EW)

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 '1' to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by writing a '1' 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 WDT is reset. See the INTFLAG register description 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 the INTFLAG register to determine which interrupt condition is present.

**Note:** Interrupts must be globally enabled for interrupt requests to be generated.

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.

### 20.6.5 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

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='1', 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 are synchronized when written:

- Control register (CTRL)
- Clear register (CLEAR)

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

#### Related Links

[Register Synchronization](#)

## 20.7 Register Summary

Offset	Name	Bit Pos.									
0x00	<a href="#">CTRL</a>	7:0	ALWAYSON					WEN	ENABLE		
0x01	<a href="#">CONFIG</a>	7:0	WINDOW[3:0]				PER[3:0]				
0x02	<a href="#">EWCTRL</a>	7:0					EWOFFSET[3:0]				
0x03	Reserved										
0x04	<a href="#">INTENCLR</a>	7:0									EW
0x05	<a href="#">INTENSET</a>	7:0									EW
0x06	<a href="#">INTFLAG</a>	7:0									EW
0x07	<a href="#">STATUS</a>	7:0	SYNCBUSY								
0x08	<a href="#">CLEAR</a>	7:0	CLEAR[7:0]								

## 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 require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

### 20.8.1 Control

**Name:** CTRL  
**Offset:** 0x0  
**Reset:** N/A - Loaded from NVM User Row at start-up  
**Property:** Write-Protected, Enable-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	ALWAYSON					WEN	ENABLE	
Access	R/W					R/W	R/W	
Reset	x					x	x	

#### 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.

This bit is not enable-protected.

These bits are loaded from NVM User Row at start-up. Refer to *NVM User Row Mapping* for more details.

Value	Description
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).

#### Bit 2 – WEN Watchdog Timer Window Mode Enable

The initial value of this bit is loaded from Flash Calibration.

This bit is loaded from NVM User Row at start-up. Refer to *NVM User Row Mapping* for more details.

Value	Description
0	Window mode is disabled (normal operation).
1	Window mode is enabled.

#### Bit 1 – ENABLE Enable

This bit enables or disables the WDT. Can only be written while CTRL.ALWAYSON is zero.

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 start-up. Refer to *NVM User Row Mapping* for more details.

Value	Description
0	The WDT is disabled.
1	The WDT is enabled.

#### Related Links

[NVM User Row Mapping](#)

## 20.8.2 Configuration

**Name:** CONFIG  
**Offset:** 0x1  
**Reset:** N/A - Loaded from NVM User Row at startup  
**Property:** Write-Protected, Enable-Protected, Write-Synchronized

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.

These bits are loaded from NVM User Row at start-up. Refer to *NVM User Row Mapping* for more details.

Value	Description
0x0	8 clock cycles
0x1	16 clock cycles
0x2	32 clock cycles
0x3	64 clock cycles
0x4	128 clock cycles
0x5	256 clocks cycles
0x6	512 clocks cycles
0x7	1024 clock cycles
0x8	2048 clock cycles
0x9	4096 clock cycles
0xA	8192 clock cycles
0xB	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.

These bits are loaded from NVM User Row at start-up. Refer to *NVM User Row Mapping* for more details.

Value	Description
0x0	8 clock cycles
0x1	16 clock cycles
0x2	32 clock cycles
0x3	64 clock cycles
0x4	128 clock cycles
0x5	256 clocks cycles
0x6	512 clocks cycles
0x7	1024 clock cycles
0x8	2048 clock cycles

# ATSAMHAXEXXA

## WDT – Watchdog Timer

Value	Description
0x9	4096 clock cycles
0xA	8192 clock cycles
0xB	16384 clock cycles
0xC-0xF	Reserved

### Related Links

[NVM User Row Mapping](#)

### 20.8.3 Early Warning Interrupt Control

**Name:** EWCTRL  
**Offset:** 0x2  
**Reset:** N/A - Loaded from NVM User Row at start-up  
**Property:** Write-Protected, Enable-Protected

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

**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. These bits are loaded from NVM User Row at start-up. Refer to *NVM User Row Mapping* for more details.

Value	Description
0x0	8 clock cycles
0x1	16 clock cycles
0x2	32 clock cycles
0x3	64 clock cycles
0x4	128 clock cycles
0x5	256 clocks cycles
0x6	512 clocks cycles
0x7	1024 clock cycles
0x8	2048 clock cycles
0x9	4096 clock cycles
0xA	8192 clock cycles
0xB	16384 clock cycles
0xC-0xF	Reserved

**Related Links**

[NVM User Row Mapping](#)

**20.8.4 Interrupt Enable Clear**

**Name:** INTENCLR  
**Offset:** 0x4  
**Reset:** 0x00  
**Property:** Write-Protected

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

**Bit 0 – EW** Early Warning Interrupt Enable  
 Writing a zero to this bit has no effect.  
 Writing a one to this bit disables the Early Warning interrupt.

Value	Description
0	The Early Warning interrupt is disabled.
1	The Early Warning interrupt is enabled.



**20.8.5 Interrupt Enable Set**

**Name:** INTENSET  
**Offset:** 0x5  
**Reset:** 0x00  
**Property:** Write-Protected

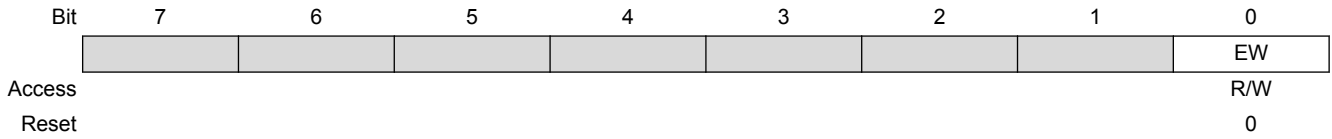
Bit	7	6	5	4	3	2	1	0
								EW
Access								R/W
Reset								0

**Bit 0 – EW** Early Warning Interrupt Enable  
 Writing a zero to this bit has no effect.  
 Writing a one to this bit enables the Early Warning interrupt.

Value	Description
0	The Early Warning interrupt is disabled.
1	The Early Warning interrupt is enabled.

**20.8.6 Interrupt Flag Status and Clear**

**Name:** INTFLAG  
**Offset:** 0x6  
**Reset:** 0x00  
**Property:** –



**Bit 0 – EW** Early Warning

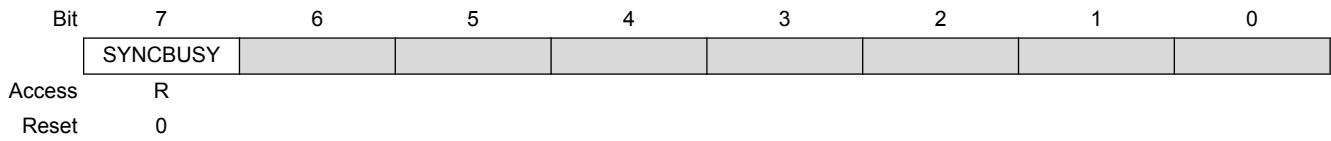
This flag is set when an Early Warning interrupt occurs, as defined by the EWOFFSET bit group in EWCTRL.

Writing a zero to this bit has no effect.

Writing a one to this bit clears the Early Warning interrupt flag.

**20.8.7 Status**

**Name:** STATUS  
**Offset:** 0x7  
**Reset:** 0x00  
**Property:** –



**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.

**20.8.8 Clear**

**Name:** CLEAR  
**Offset:** 0x8  
**Reset:** 0x00  
**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.

## **21. RTC – Real-Time Counter**

### **21.1 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 can be reset on the occurrence of an alarm/compare match. This allows periodic interrupts and peripheral events at very long and accurate intervals.

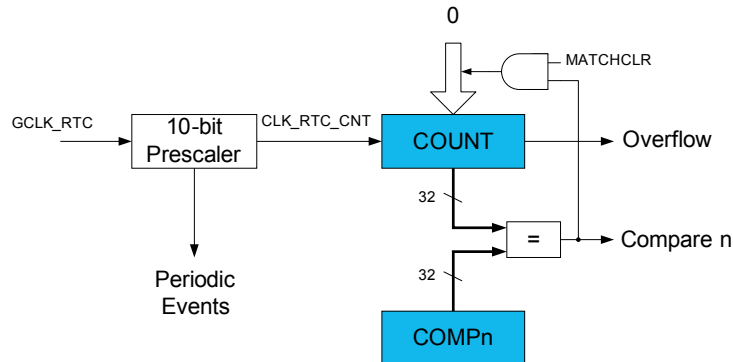
The 10-bit programmable prescaler can scale down the clock source. By this, a wide range of resolutions and time-out periods can be configured. With a 32.768kHz clock source, the minimum counter tick interval is 30.5 $\mu$ s, and time-out periods can range up to 36 hours. For a counter tick interval of 1s, the maximum time-out period is more than 136 years.

### **21.2 Features**

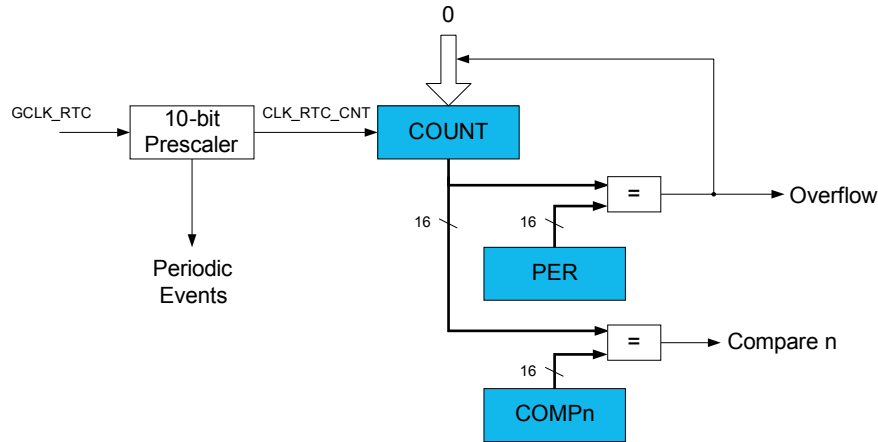
- 32-bit counter with 10-bit prescaler
- Multiple clock sources
- 32-bit or 16-bit Counter mode
  - One 32-bit or two 16-bit compare values
- Clock/Calendar mode
  - Time in seconds, minutes and hours (12/24)
  - Date in day of month, month and year
  - Leap year correction
- Digital prescaler correction/tuning for increased accuracy
- Overflow, alarm/compare match and prescaler interrupts and events
  - Optional clear on alarm/compare match

**21.3 Block Diagram**

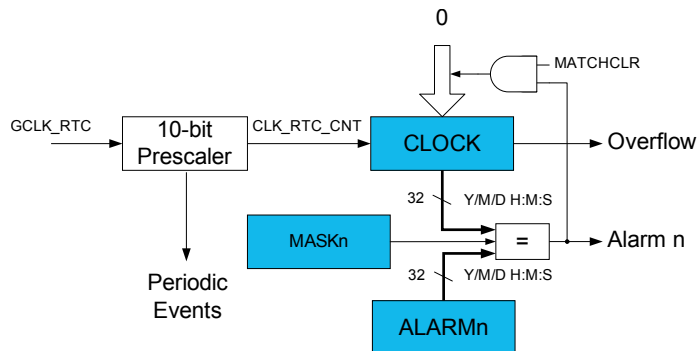
**Figure 21-1. RTC Block Diagram (Mode 0 — 32-Bit Counter)**



**Figure 21-2. RTC Block Diagram (Mode 1 — 16-Bit Counter)**



**Figure 21-3. RTC Block Diagram (Mode 2 — Clock/Calendar)**



**21.4 Signal Description**

Not applicable.

**21.5 Product Dependencies**

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

### 21.5.1 I/O Lines

Not applicable.

### 21.5.2 Power Management

The RTC will continue to operate in any sleep mode where the selected source clock is running. The RTC 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 the *Power Manager* for details on the different sleep modes.

The RTC will be reset only at power-on (POR) or by setting the Software Reset bit in the Control register (CTRL.SWRST=1).

#### Related Links

[PM – Power Manager](#)

### 21.5.3 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.

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* 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](#) for further details.

The RTC should not work with the Generic Clock Generator 0.

#### Related Links

[Peripheral Clock Masking](#)

[GCLK - Generic Clock Controller](#)

### 21.5.4 DMA

Not applicable.

### 21.5.5 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* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 21.5.6 Events

The events are connected to the *Event System*.

#### Related Links

[EVSYS – Event System](#)

### 21.5.7 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 Debug Control (DBGCTRL) register for details.

### 21.5.8 Register Access Protection

Registers with write-access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag Status and Clear register (INTFLAG)
- Read Request register (READREQ)
- Status register (STATUS)
- Debug register (DBGCTRL)

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

Write-protection does not apply for accesses through an external debugger.

### 21.5.9 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* for details.

#### Related Links

[Electrical Characteristics](#)

## 21.6 Functional Description

### 21.6.1 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.

The RTC can function in one of these modes:

- Mode 0 - COUNT32: RTC serves as 32-bit counter
- Mode 1 - COUNT16: RTC serves as 16-bit counter
- Mode 2 - CLOCK: RTC serves as clock/calendar with alarm functionality

### 21.6.2 Basic Operation

#### 21.6.2.1 Initialization

The following bits are enable-protected, meaning that they can only be written when the RTC is disabled (CTRL.ENABLE=0):

- 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=1) 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-Protected" property in the register description.

Before the RTC is enabled, it must be configured, as outlined by the following steps:

1. RTC operation mode must be selected by writing the Operating Mode bit group in the Control register (CTRL.MODE)
2. Clock representation must be selected by writing the Clock Representation bit in the Control register (CTRL.CLKREP)
3. Prescaler value must be selected by writing the Prescaler bit group in the Control register (CTRL.PRESCALER)

The RTC prescaler divides the source clock for the RTC counter.

**Note:** In Clock/Calendar mode, the prescaler must be configured to provide a 1Hz clock to the counter for correct operation.

The frequency of the RTC clock (CLK\_RTC\_CNT) is given by the following formula:

$$f_{\text{CLK\_RTC\_CNT}} = \frac{f_{\text{GCLK\_RTC}}}{2^{\text{PRESCALER}}}$$

The frequency of the generic clock, GCLK\_RTC, is given by  $f_{\text{GCLK\_RTC}}$ , and  $f_{\text{CLK\_RTC\_CNT}}$  is the frequency of the internal prescaled RTC clock, CLK\_RTC\_CNT.

### 21.6.2.2 Enabling, Disabling and Resetting

The RTC is enabled by setting the Enable bit in the Control register (CTRL.ENABLE=1). The RTC is disabled by writing CTRL.ENABLE=0.

The RTC is reset by setting the Software Reset bit in the Control register (CTRL.SWRST=1). All registers in the RTC, except DEBUG, will be reset to their initial state, and the RTC will be disabled. The RTC must be disabled before resetting it.

### 21.6.3 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).

#### 21.6.3.1 32-Bit Counter (Mode 0)

When the RTC Operating Mode bits in the Control register are zero (CTRL.MODE=00), the counter operates in 32-bit Counter mode. The block diagram of this mode is shown in [Figure 21-1](#). 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 register (COMP0). When a compare match occurs, the Compare 0 interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMP0) 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 '1', the counter is cleared on the next counter cycle when a compare match with COMP0 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 '1', INTFLAG.CMP0 and INTFLAG.OVF will both be set simultaneously on a compare match with COMP0.

### 21.6.3.2 16-Bit Counter (Mode 1)

When the RTC Operating Mode bits in the Control register (CTRL.MODE) are 1, the counter operates in 16-bit Counter mode as shown in [Figure 21-2](#). 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.

### 21.6.3.3 Clock/Calendar (Mode 2)

When CTRL.MODE is two, the counter operates in Clock/Calendar mode, as shown in [Figure 21-3](#). 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 register (ALARM0). When an alarm match occurs, the Alarm 0 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. E.g. For a 1Hz clock counter, it means the Alarm 0 Interrupt flag is set with a delay of 1s after the occurrence of alarm match. A valid alarm match depends on the setting of the Alarm Mask Selection bits in the Alarm

A valid alarm match depends on the setting of the Alarm Mask Selection bits in the Alarm 0 Mask register (MASK0.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 ALARM0 occurs. This allows the RTC to generate periodic interrupts or events with longer periods than are possible with the prescaler events (see [Periodic Events](#)).

Note that when CTRL.MATCHCLR is '1', INTFLAG.ALARM0 and INTFLAG.OVF will both be set simultaneously on an alarm match with ALARM0.

### 21.6.4 DMA Operation

Not applicable.

### 21.6.5 Interrupts

The RTC has the following interrupt sources which are asynchronous interrupts and can wake-up the device from any sleep mode.:

- Overflow (INTFLAG.OVF): Indicates that the counter has reached its top value and wrapped to zero.
- Compare n (INTFLAG.CMPn): Indicates a match between the counter value and the compare register.
- Alarm n (INTFLAG.ALARMn): Indicates a match between the clock value and the alarm register.
- Synchronization Ready (INTFLAG.SYNCRDY): Indicates an operation requires synchronization.

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 setting the corresponding bit in the Interrupt Enable Set register (INTENSET=1), and disabled by setting the corresponding bit in the Interrupt Enable Clear register (INTENCLR=1). An interrupt request is generated when the interrupt flag is raised and the corresponding interrupt is enabled. The interrupt request remains active until either the interrupt flag is cleared, the interrupt is disabled or the RTC is reset. See the description of the INTFLAG registers 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 the Nested Vector Interrupt Controller for details. The user must read the INTFLAG register to determine which interrupt condition is present.

**Note:** Interrupts must be globally enabled for interrupt requests to be generated. Refer to the Nested Vector Interrupt Controller for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 21.6.6 Events

The RTC can generate the following output events, which are generated in the same way as the corresponding interrupts:

- Overflow (OVF): Indicates that the counter has reached its top value and wrapped to zero.
- Period n (PERn): The corresponding bit in the prescaler has toggled. Refer to [Periodic Events](#) for details.
- Compare n (CMPn): Indicates a match between the counter value and the compare register.
- Alarm n (ALARMn): Indicates a match between the clock value and the alarm register.

Setting the Event Output bit in the Event Control Register (EVCTRL.xxxEO=1) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. Refer to the *EVSYS - Event System* for details on configuring the event system.

#### Related Links

[EVSYS – Event System](#)

### 21.6.7 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. 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 any interrupt. In this case, the CPU will continue executing right from the first instruction that followed 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 *Event System* for more information.

#### Related Links

[EVSYS – Event System](#)

### 21.6.8 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

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. 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 are synchronized when written:

- Software Reset bit in the Control register (CTRL.SWRST)
- Enable bit in the Control register (CTRL.ENABLE)

The following registers are synchronized when written:

- Counter Value register (COUNT)
- Clock Value register (CLOCK)
- Counter Period register (PER)
- Compare n Value registers (COMPn)
- Alarm n Value registers (ALARMn)
- Frequency Correction register (FREQCORR)
- Alarm n Mask register (MASKn)

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

The following registers are synchronized when read:

- The Counter Value register (COUNT)
- The Clock Value register (CLOCK)

Required read-synchronization is denoted by the "Read-Synchronized" property in the register description.

#### Related Links

Register Synchronization

**21.6.9 Additional Features**

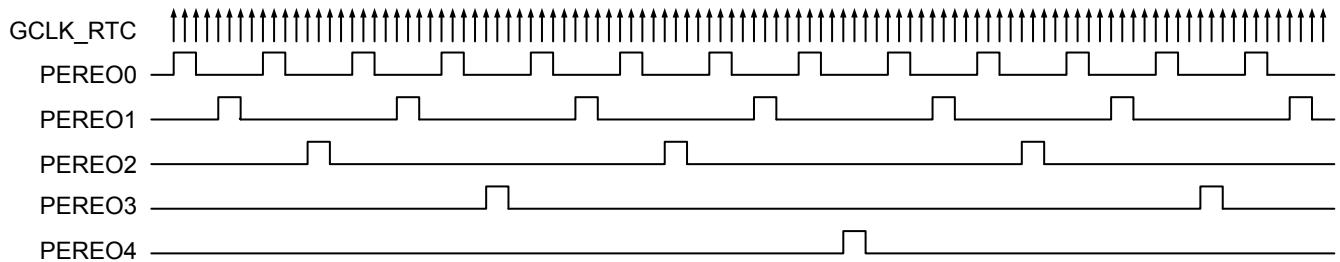
**21.6.9.1 Periodic Events**

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 eight Periodic Event Output bits in the Event Control register (EVCTRL.PERE0[n=0..7]) is '1', an event is generated on the 0-to-1 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.PERE0n bit. For example, PER0 will generate an event every eight CLK\_RTC\_OSC cycles, PER1 every 16 cycles, etc. This is shown in the figure below. 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 21-4. Example Periodic Events**



**21.6.9.2 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 1 ppm steps. Digital correction is achieved by adding or skipping a single count in the prescaler once every 4096 GCLK\_RTC\_OSC cycles. The Value bit group in the Frequency Correction register (FREQCORR.VALUE) determines the number of times the adjustment is applied over 240 of these periods. The resulting correction is as follows:

$$\text{Correction in ppm} = (\text{FREQCORR.VALUE} / 4096 * 240) * 10^{6ppm}$$

This results in a resolution of 1.017 PPM.

The Sign bit in the Frequency Correction register (FREQCORR.SIGN) determines the direction of the correction. A positive value will add counts and increase the period (reducing the frequency), and a negative value will reduce counts per period (speeding up 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.

## 21.7 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 21-1. 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 21-2. 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

# ATSAMHAXEXXA

## RTC – Real-Time Counter

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 21-3. 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	PERE07	PERE06	PERE05	PERE04	PERE03	PERE02	PERE01	PERE00
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]		
0x14	Reserved									

Offset	Name	Bit Pos.								
...										
0x17										
0x18	ALARM0	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]		

### 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.

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

Some registers require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.



### 21.8.1 Control - MODE0

**Name:** CTRL  
**Offset:** 0x00  
**Reset:** 0x0000  
**Property:** Enable-Protected, Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
					PRESCALER[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	MATCHCLR					MODE[1:0]		ENABLE
Access	R/W			R/W	R/W	R/W	R/W	W
Reset	0			0	0	0	0	0

#### 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.

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 is not synchronized.

Value	Description
0	The counter is not cleared on a Compare/Alarm 0 match.
1	The counter is cleared on a Compare/Alarm 0 match.

### Bits 3:2 – MODE[1:0] Operating Mode

These bits define the operating mode of the RTC.

These bits are not synchronized.

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

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.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled or being enabled.

### 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 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.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

### 21.8.2 Control - MODE1

**Name:** CTRL  
**Offset:** 0x00  
**Reset:** 0x0000  
**Property:** Enable-Protected, Write-Protected, Write-Synchronized

	Bit	15	14	13	12	11	10	9	8
		PRESCALER[3:0]							
Access						R/W	R/W	R/W	R/W
Reset						0	0	0	0
	Bit	7	6	5	4	3	2	1	0
		MODE[1:0]				ENABLE		SWRST	
Access						R/W	R/W	R/W	W
Reset						0	0	0	0

#### 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.

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 3:2 – MODE[1:0] Operating Mode

These bits define the operating mode of the RTC.

These bits are not synchronized.

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

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.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled or being enabled.

**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 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.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

### 21.8.3 Control - MODE2

**Name:** CTRL  
**Offset:** 0x00  
**Reset:** 0x0000  
**Property:** Enable-Protected, Write-Protected, Write-Synchronized

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

#### 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.

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.

This bit is not synchronized.

Value	Description
0	The counter is not cleared on a Compare/Alarm 0 match.
1	The counter is cleared on a Compare/Alarm 0 match.

### 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.

This bit is not synchronized.

Value	Description
0	24 Hour
1	12 Hour (AM/PM)

### Bits 3:2 – MODE[1:0] Operating Mode

These bits define the operating mode of the RTC.

These bits are not synchronized.

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

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.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled or being enabled.

### 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 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.

# ATSAMHAXEXXA

## RTC – Real-Time Counter

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

### 21.8.4 Read Request

**Name:** READREQ  
**Offset:** 0x02  
**Reset:** 0x0010  
**Property:** -

	Bit	15	14	13	12	11	10	9	8	
		RREQ	RCONT							
Access		W	R/W							
Reset		0	0							
	Bit	7	6	5	4	3	2	1	0	
				ADDR[5:0]						
Access				R	R	R	R	R	R	
Reset				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 an RTC register is written.

**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.



### 21.8.5 Event Control - MODE0

**Name:** EVCTRL  
**Offset:** 0x04  
**Reset:** 0x0000  
**Property:** Enable-Protected, Write-Protected

Bit	15	14	13	12	11	10	9	8
	OVFEO							CMPEO0
Access	R/W							R/W
Reset	0							0
Bit	7	6	5	4	3	2	1	0
	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx
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

Value	Description
0	Overflow event is disabled and will not be generated.
1	Overflow event is enabled and will be generated for every overflow.

**Bit 8 – CMPEO0** Compare 0 Event Output Enable

Value	Description
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 7,6,5,4,3,2,1,0 – PEREOx** Periodic Interval x Event Output Enable [x=7:0]

Value	Description
0	Periodic Interval x event is disabled and will not be generated.
1	Periodic Interval x event is enabled and will be generated.

### 21.8.6 Event Control - MODE1

**Name:** EVCTRL  
**Offset:** 0x04  
**Reset:** 0x0000  
**Property:** Enable-Protected, Write-Protected

Bit	15	14	13	12	11	10	9	8
	OVFEO						CMPEOx	CMPEOx
Access	R/W						R/W	R/W
Reset	0						0	0
Bit	7	6	5	4	3	2	1	0
	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx
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

Value	Description
0	Overflow event is disabled and will not be generated.
1	Overflow event is enabled and will be generated for every overflow.

#### Bits 9,8 – CMPEOx Compare x Event Output Enable [x=1:0]

Value	Description
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 7,6,5,4,3,2,1,0 – PEREOx Periodic Interval x Event Output Enable [x=7:0]

Value	Description
0	Periodic Interval x event is disabled and will not be generated.
1	Periodic Interval x event is enabled and will be generated.

**21.8.7 Event Control - MODE2**

**Name:** EVCTRL  
**Offset:** 0x04  
**Reset:** 0x0000  
**Property:** Enable-Protected, Write-Protected

Bit	15	14	13	12	11	10	9	8
	OVFEO							ALARMEO0
Access	R/W							R/W
Reset	0							0
Bit	7	6	5	4	3	2	1	0
	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx	PEREOx
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

Value	Description
0	Overflow event is disabled and will not be generated.
1	Overflow event is enabled and will be generated for every overflow.

**Bit 8 – ALARMEO0** Alarm 0 Event Output Enable

Value	Description
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 7,6,5,4,3,2,1,0 – PEREOx** Periodic Interval x Event Output Enable [x=7:0]

Value	Description
0	Periodic Interval x event is disabled and will not be generated.
1	Periodic Interval x event is enabled and will be generated.

### 21.8.8 Interrupt Enable Clear - MODE0

**Name:** INTENCLR  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						CMP0
Access	R/W	R/W						R/W
Reset	0	0						0

**Bit 7 – OVF** Overflow Interrupt Enable  
 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.

Value	Description
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.

**Bit 6 – SYNCRDY** Synchronization Ready Interrupt Enable  
 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.

Value	Description
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.

**Bit 0 – CMP0** Compare 0 Interrupt Enable  
 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.

Value	Description
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.

### 21.8.9 Interrupt Enable Clear - MODE1

**Name:** INTENCLR  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** Write-Protected

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

**Bit 7 – OVF** Overflow Interrupt Enable  
 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.

Value	Description
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.

**Bit 6 – SYNCRDY** Synchronization Ready Interrupt Enable  
 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.

Value	Description
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.

**Bits 1,0 – CMPx** Compare x Interrupt Enable [x=1:0]  
 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.

Value	Description
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.

### 21.8.10 Interrupt Enable Clear - MODE2

**Name:** INTENCLR  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						ALARM0
Access	R/W	R/W						R/W
Reset	0	0						0

**Bit 7 – OVF** Overflow Interrupt Enable  
 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.

Value	Description
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.

**Bit 6 – SYNCRDY** Synchronization Ready Interrupt Enable  
 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.

Value	Description
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.

**Bit 0 – ALARM0** Alarm 0 Interrupt Enable  
 Writing a zero to this bit has no effect.

Writing a one to this bit disables the Alarm 0 interrupt.

Value	Description
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.

### 21.8.11 Interrupt Enable Set - MODE0

**Name:** INTENSET  
**Offset:** 0x07  
**Reset:** 0x00  
**Property:** Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						CMP0
Access	R/W	R/W						R/W
Reset	0	0						0

**Bit 7 – OVF** Overflow Interrupt Enable  
 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.

Value	Description
0	The overflow interrupt is disabled.
1	The overflow interrupt is enabled.

**Bit 6 – SYNCRDY** Synchronization Ready Interrupt Enable  
 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.

Value	Description
0	The synchronization ready interrupt is disabled.
1	The synchronization ready interrupt is enabled.

**Bit 0 – CMP0** Compare 0 Interrupt Enable  
 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.

Value	Description
0	The compare 0 interrupt is disabled.
1	The compare 0 interrupt is enabled.

### 21.8.12 Interrupt Enable Set - MODE1

**Name:** INTENSET  
**Offset:** 0x07  
**Reset:** 0x00  
**Property:** Write-Protected

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

**Bit 7 – OVF** Overflow Interrupt Enable

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.

Value	Description
0	The overflow interrupt is disabled.
1	The overflow interrupt is enabled.

**Bit 6 – SYNCRDY** Synchronization Ready Interrupt Enable

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.

Value	Description
0	The synchronization ready interrupt is disabled.
1	The synchronization ready interrupt is enabled.

**Bits 1,0 – CMPx** Compare x Interrupt Enable [x=1:0]

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.

Value	Description
0	The compare x interrupt is disabled.
1	The compare x interrupt is enabled.



### 21.8.13 Interrupt Enable Set - MODE2

**Name:** INTENSET  
**Offset:** 0x07  
**Reset:** 0x00  
**Property:** Write-Protected

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						ALARM0
Access	R/W	R/W						R/W
Reset	0	0						0

**Bit 7 – OVF** Overflow Interrupt Enable  
 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.

Value	Description
0	The overflow interrupt is disabled.
1	The overflow interrupt is enabled.

**Bit 6 – SYNCRDY** Synchronization Ready Interrupt Enable  
 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.

Value	Description
0	The synchronization ready interrupt is disabled.
1	The synchronization ready interrupt is enabled.

**Bit 0 – ALARM0** Alarm 0 Interrupt Enable  
 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.

Value	Description
0	The alarm 0 interrupt is disabled.
1	The alarm 0 interrupt is enabled.

### 21.8.14 Interrupt Flag Status and Clear - MODE0

**Name:** INTFLAG  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						CMP0
Access	R/W	R/W						R/W
Reset	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 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 0 – CMP0** Compare 0

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.

### 21.8.15 Interrupt Flag Status and Clear - MODE1

**Name:** INTFLAG  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	OVF	SYNCRDY					CMPx	CMPx
Access	R/W	R/W					R/W	R/W
Reset	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 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.

**Bits 1,0 – CMPx** Compare x [x=1:0]

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.

### 21.8.16 Interrupt Flag Status and Clear - MODE2

**Name:** INTFLAG  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** -

	7	6	5	4	3	2	1	0
	OVF	SYNCRDY						ALARM0
Access	R/W	R/W						R/W
Reset	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 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 0 – ALARM0** Alarm 0

This flag is cleared by writing a one to the flag.

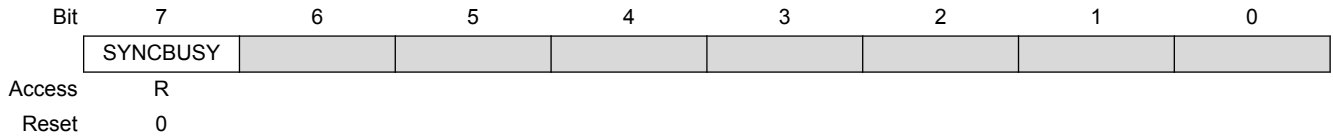
This flag is set on the next CLK\_RTC\_CNT cycle after a match with ALARM0 condition occurs, and an interrupt request will be generated if INTENCLR/SET.ALARM0 is also one.

Writing a zero to this bit has no effect.

Writing a one to this bit clears the Alarm 0 interrupt flag.

**21.8.17 Status**

**Name:** STATUS  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** -



**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.

**21.8.18 Debug Control**

**Name:** DBGCTRL  
**Offset:** 0x0B  
**Reset:** 0x00  
**Property:** -

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

**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.

### 21.8.19 Frequency Correction

**Name:** FREQCORR  
**Offset:** 0x0C  
**Reset:** 0x00  
**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

Value	Description
0	The correction value is positive, i.e., frequency will be decreased.
1	The correction value is negative, i.e., frequency will be increased.

#### Bits 6:0 – VALUE[6:0] Correction Value

These bits define the amount of correction applied to the RTC prescaler.

1–127: The RTC frequency is adjusted according to the value.

Value	Description
0	Correction is disabled and the RTC frequency is unchanged.

**21.8.20 Counter Value - MODE0**

**Name:** COUNT  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** Read-Synchronized, Write-Protected, Write-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	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.



**21.8.21 Counter Value - MODE1**

**Name:** COUNT  
**Offset:** 0x10  
**Reset:** 0x0000  
**Property:** Read-Synchronized, Write-Protected, Write-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 define the value of the 16-bit RTC counter.

### 21.8.22 Clock Value - MODE2

**Name:** CLOCK  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** Read-Synchronized, 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[4:4]	
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 21-4. Hour**

HOUR[4:0]	CLOCK.HOUR[4]	CLOCK.HOUR[3:0]	Description
0	0x00 - 0x17		Hour (0 - 23)
	0x18 - 0x1F		Reserved
1	0	0x0	Reserved
		0x1 - 0xC	AM Hour (1 - 12)
		0xD - 0xF	Reserved
	1	0x0	Reserved
		0x1 - 0xC	PM Hour (1 - 12)
		0xF - 0xF	Reserved

**Bits 11:6 – MINUTE[5:0]** Minute  
0 – 59.

**Bits 5:0 – SECOND[5:0]** Second  
0– 59.

**21.8.23 Counter Period - MODE1**

**Name:** PER  
**Offset:** 0x14  
**Reset:** 0x0000  
**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.

**21.8.24 Compare n Value - MODE0**

**Name:** COMP  
**Offset:** 0x18  
**Reset:** 0x00000000  
**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.

**21.8.25 Compare n Value - MODE1**

**Name:** COMPn  
**Offset:** 0x18+n\*0x2 [n=0..1]  
**Reset:** 0x0000  
**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.

**21.8.26 Alarm 0 Value - MODE2**

**Name:** ALARM0  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** Write-Protected, Write-Synchronized

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.

	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[4:4]	
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 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.

**21.8.27 Alarm n Mask - MODE2**

**Name:** MASK  
**Offset:** 0x1C  
**Reset:** 0x00  
**Property:** Write-Protected, Write-Synchronized

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

**Bits 2:0 – SEL[2:0]** Alarm Mask Selection  
 These bits define which bit groups of Alarm n are valid.

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



## 22. DMAC – Direct Memory Access Controller

### 22.1 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.

The DMA part of the DMAC has several DMA channels which all can receive different types of transfer triggers to generate transfer requests from the DMA channels to the arbiter, see also the [Block Diagram](#). The arbiter will grant one DMA channel at a time to act as the active channel. When an active channel has been granted, the fetch engine of the DMAC will fetch a transfer descriptor from the SRAM and store it in the internal memory of the active channel, which will execute the data transmission.

An ongoing data transfer of an active channel can be interrupted by a higher prioritized DMA channel. The DMAC will write back the updated transfer descriptor from the internal memory of the active channel to SRAM, and grant the higher prioritized channel to start transfer as the new active channel. Once a DMA channel is done with its transfer, interrupts and events can be generated optionally.

The DMAC has four bus interfaces:

- The *data transfer bus* is used for performing the actual DMA transfer.
- The *AHB/APB Bridge bus* is used when writing and reading the I/O registers of the DMAC.
- The *descriptor fetch bus* is used by the fetch engine to fetch transfer descriptors before data transfer can be started or continued.
- The *write-back bus* is used to write the transfer descriptor back to SRAM.

All buses are AHB master interfaces but the AHB/APB Bridge bus, which is an APB slave interface.

The CRC engine 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.

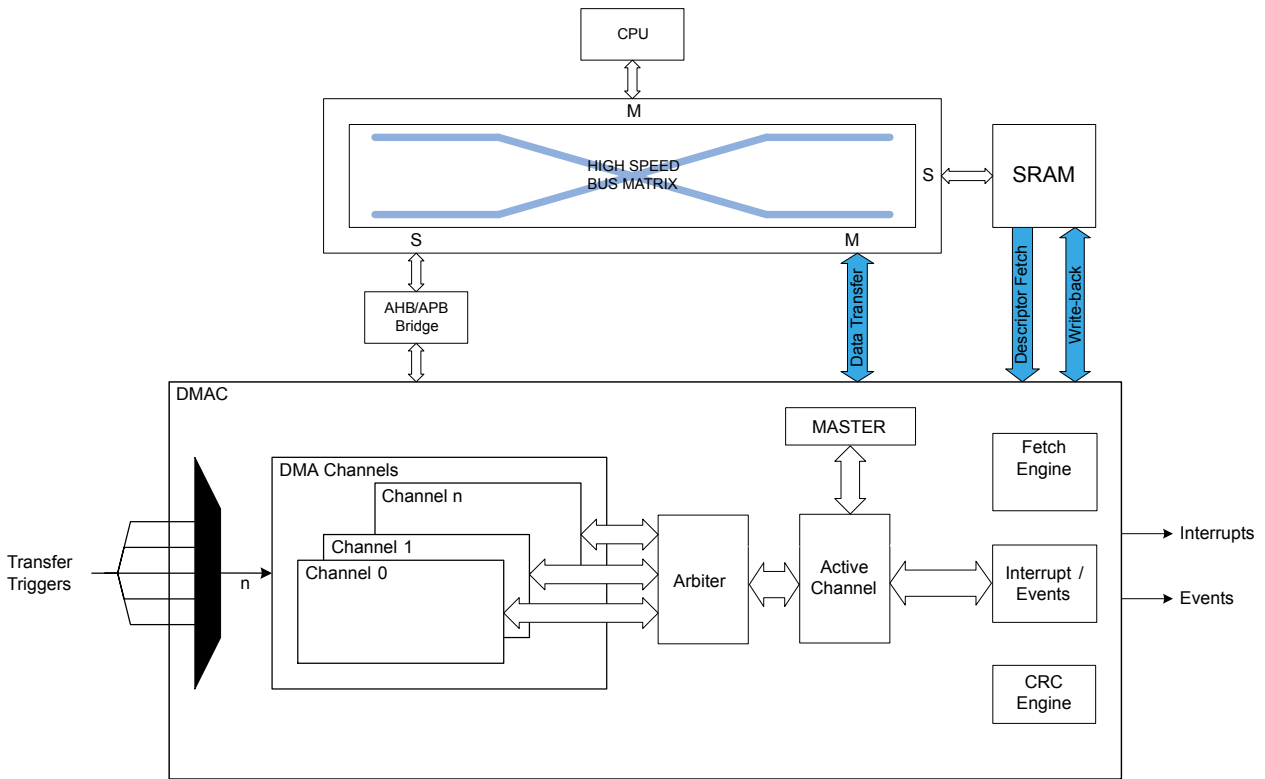
### 22.2 Features

- Data transfer from:
  - 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

- Up to 8 channels
  - Enable 8 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, 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)

22.3 Block Diagram

Figure 22-1. DMAC Block Diagram



22.4 Signal Description

Not applicable.

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

Not applicable.

22.5.2 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. On hardware or software reset, all registers are set to their reset value.

Related Links

[PM – Power Manager](#)

22.5.3 Clocks

The DMAC bus clock (CLK\_DMACH\_APB) must be configured and enabled in the Power Manager before using the DMAC.

---

An AHB clock (CLK\_DMACH\_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\_DMACH\_AHB can be found in *Peripheral Clock Masking*.

This bus clock (CLK\_DMACH\_APB) is always synchronous to the module clock (CLK\_DMACH\_AHB), but can be divided by a prescaler and may run even when the module clock is turned off.

**Related Links**

[Peripheral Clock Masking](#)

**22.5.4 DMA**

Not applicable.

**22.5.5 Interrupts**

The interrupt request line is connected to the interrupt controller. Using the DMAC interrupt requires the interrupt controller to be configured first.

**Related Links**

[Nested Vector Interrupt Controller](#)

**22.5.6 Events**

The events are connected to the event system.

**Related Links**

[EVSYS – Event System](#)

**22.5.7 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.

**22.5.8 Register Access Protection**

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

- Interrupt Pending register (INTPEND)
- Channel ID register (CHID)
- Channel Interrupt Flag Status and Clear register (CHINTFLAG)

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

PAC write-protection does not apply to accesses through an external debugger.

**Related Links**

[PAC - Peripheral Access Controller](#)

**22.5.9 Analog Connections**

Not applicable.

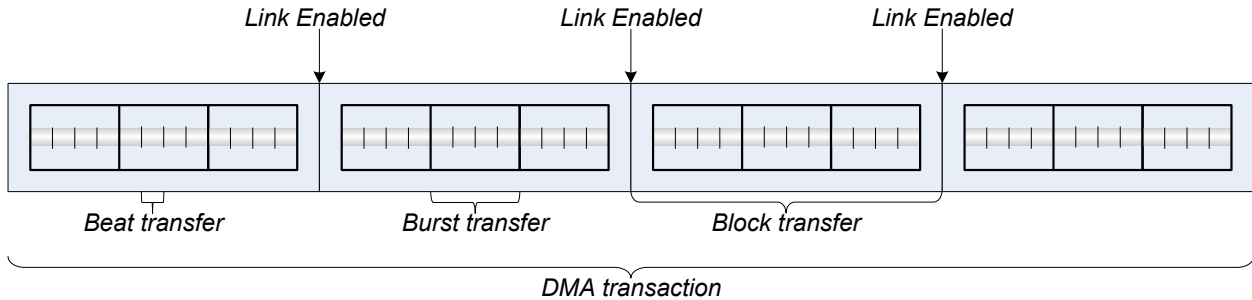
**22.6 Functional Description****22.6.1 Principle of Operation**

The DMAC consists of a DMA module and a CRC module.

22.6.1.1 DMA

The DMAC can transfer data between memories and peripherals without interaction from the CPU. The data transferred by the DMAC are called transactions, and these transactions can be split into smaller data transfers. The following figure shows the relationship between the different transfer sizes:

Figure 22-2. DMA Transfer Sizes



- Beat transfer: 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. A burst transfer is atomic, cannot be interrupted and the length of the burst is selected by writing the Burst Length bit group in each Channel n Control A register (CHCTRLA.BURSTLEN).
- Block transfer: The amount of data one transfer descriptor can transfer, and the amount can range from 1 to 64k beats. A block transfer can be interrupted, in contrast to the burst transfer.
- Transaction: The DMAC can link several transfer descriptors by having the first descriptor pointing to the second and so forth, as shown in the figure above. A DMA transaction is the complete transfer of all blocks within a linked list.

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](#).

The figure above shows several block transfers linked together, which are called linked descriptors. For further information about linked descriptors, refer to [Linked Descriptors](#).

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. If there are several DMA channels with pending transfer requests, the arbiter chooses which channel is granted 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. A current transaction can be interrupted by a higher prioritized channel after each burst transfer, but will resume the block transfer when the according DMA channel is granted access as the active channel again.

For each beat transfer, an optional output event can be generated. 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.

22.6.1.2 CRC

The internal CRC engine supports two commonly used CRC polynomials: CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3). It can be used on a selectable DMA channel, or on the I/O interface. Refer to [CRC Operation](#) for details.

### 22.6.2 Basic Operation

#### 22.6.2.1 Initialization

The following DMAC registers are enable-protected, meaning that they can only be written when the DMAC is disabled (CTRL.DMAENABLE=0):

- Descriptor Base Memory Address register (BASEADDR)
- Write-Back Memory Base Address register (WRBADDR)

The following DMAC bit is enable-protected, meaning that it can only be written when both the DMAC and CRC are disabled (CTRL.DMAENABLE=0 and CTRL.CRCENABLE=0):

- 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=0):

- Channel Control B (CHCTRLB) register, except the Command bit (CHCTRLB.COMD) and the Channel Arbitration Level bit (CHCTRLB.LVL)

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=0):

- CRC Control register (CRCCTRL)
- CRC Checksum register (CRCCHKSUM)

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 setting the Priority Level  $x$  Enable bit in the Control register (CTRL.LVLEN $x$ =1)

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 engine must be configured before it is enabled, as outlined by the following steps:

- The CRC input source must be selected by writing the CRC Input Source bit group in the CRC Control register (CRCCTRL.CRCSRC)
- The 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 selected as input source, the beat size must be selected by writing the CRC Beat Size bit group in the CRC Control register (CRCCTRL.CRCBEATSIZE)

#### 22.6.2.2 Enabling, Disabling, and Resetting

The DMAC is enabled by writing the DMA Enable bit in the Control register (CTRL.DMAENABLE) to '1'. The DMAC is disabled by writing a '0' to CTRL.DMAENABLE.

A DMA channel is enabled by writing the Enable bit in the Channel Control A register (CHCTRLA.ENABLE) to '1', 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 '0' to CHCTRLA.ENABLE.

The CRC is enabled by writing a '1' to the CRC Enable bit in the Control register (CTRL.CRCENABLE). The CRC is disabled by writing a '0' to CTRL.CRCENABLE.

The DMAC is reset by writing a '1' to the Software Reset bit in the Control register (CTRL.SWRST) while 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 '1' 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.

#### 22.6.2.3 Transfer Descriptors

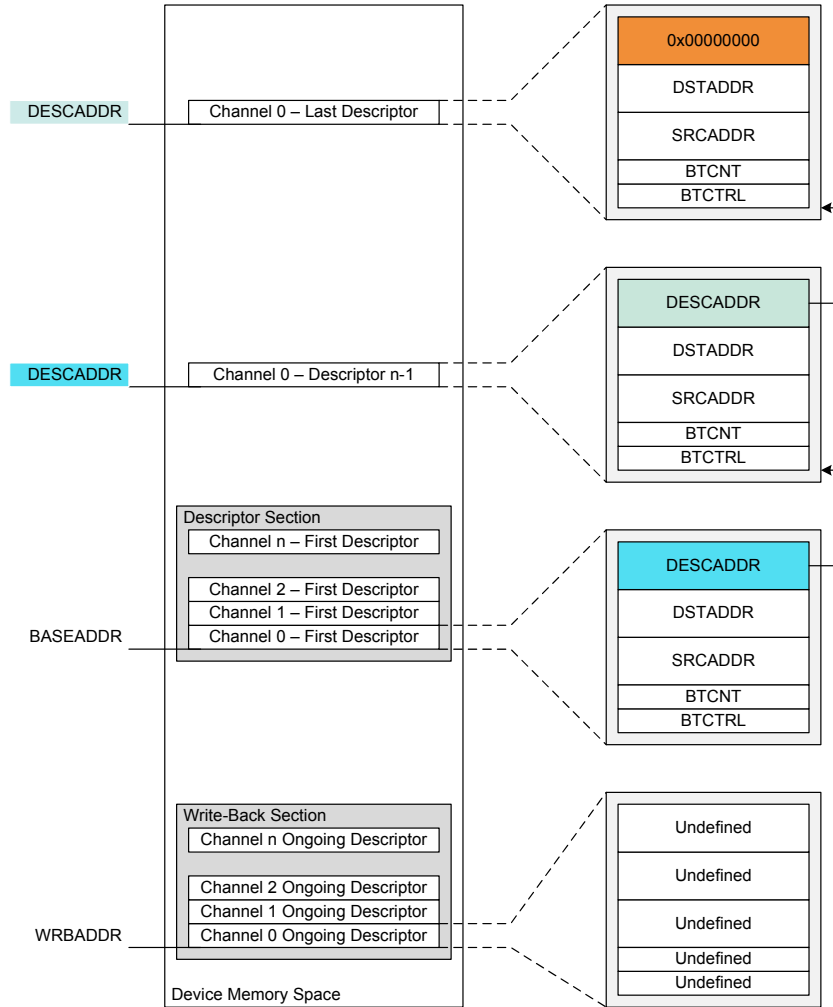
Together with the channel configurations the transfer descriptors decides how a block transfer should be executed. Before a DMA channel is enabled (CHCTRLA.ENABLE is written to one), and receives a transfer trigger, its first transfer descriptor has to be initialized and valid (BTCTRL.VALID). The first transfer descriptor describes the first block transfer of a transaction.

All transfer descriptors must reside in SRAM. The addresses stored in the Descriptor Memory Section Base Address (BASEADDR) and Write-Back Memory Section Base Address (WRBADDR) registers tell 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 (see figure below), all first transfer descriptors must be stored in a contiguous memory section, where the transfer descriptors must be ordered according to their channel number. For further details on linked descriptors, refer to [Linked Descriptors](#).

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. The figure below shows an example of linked descriptors on DMA channel 0. For further details on linked descriptors, refer to [Linked Descriptors](#).

Figure 22-3. Memory Sections



The size of the descriptor and write-back memory sections is dependent on the number of the most significant enabled DMA channel  $m$ , as shown below:

$$Size = 128\text{bits} \cdot (m + 1)$$

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.



22.6.2.4 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. Depending on the arbitration scheme, the arbiter will choose which DMA channel will be the next active channel. 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 bit PENDCH.PENDCHx will be cleared. See also the following figure.

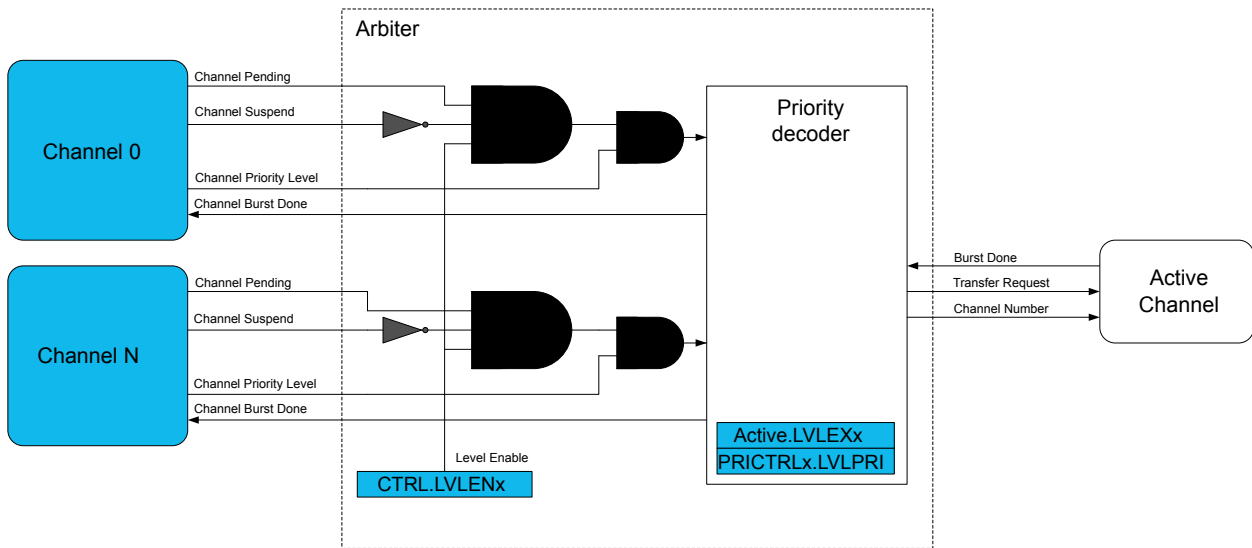
If the upcoming burst transfer is the first for the transfer request, the corresponding Busy Channel x bit in the Busy Channels register will be set (BUSYCH.BUSYCHx=1), and it will remain '1' for the subsequent granted burst transfers.

When the channel has performed its granted burst transfer(s) it will be either fed into the queue of channels with pending transfers, set to be waiting for a new transfer trigger, suspended, or 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 '1'. 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=0) 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 22-4. Arbiter Overview



Priority Levels

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 a higher priority level number will have priority over

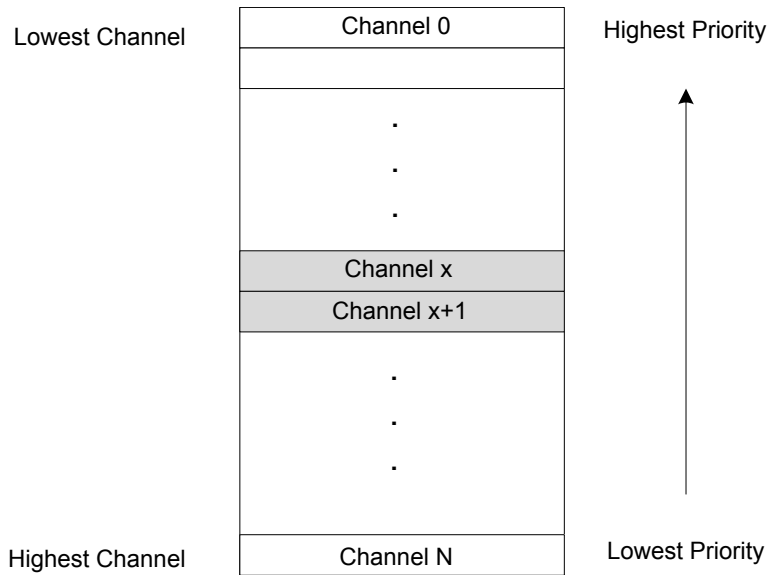
a channel with a lower priority level number. Each priority level x is enabled by setting the corresponding Priority Level x Enable bit in the Control register ([CTRL.LVLENx=1](#)).

Within each priority level the DMAC's arbiter can be configured to prioritize statically or dynamically:

*Static Arbitration* within a priority level is selected by writing a '0' to the Level x Round-Robin Scheduling Enable bit in the Priority Control 0 register ([PRICTRL0.RRLVLENx](#)).

When static arbitration is selected, the arbiter will prioritize a low channel number over a high channel number as shown in the figure below. When using the static arbitration 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.

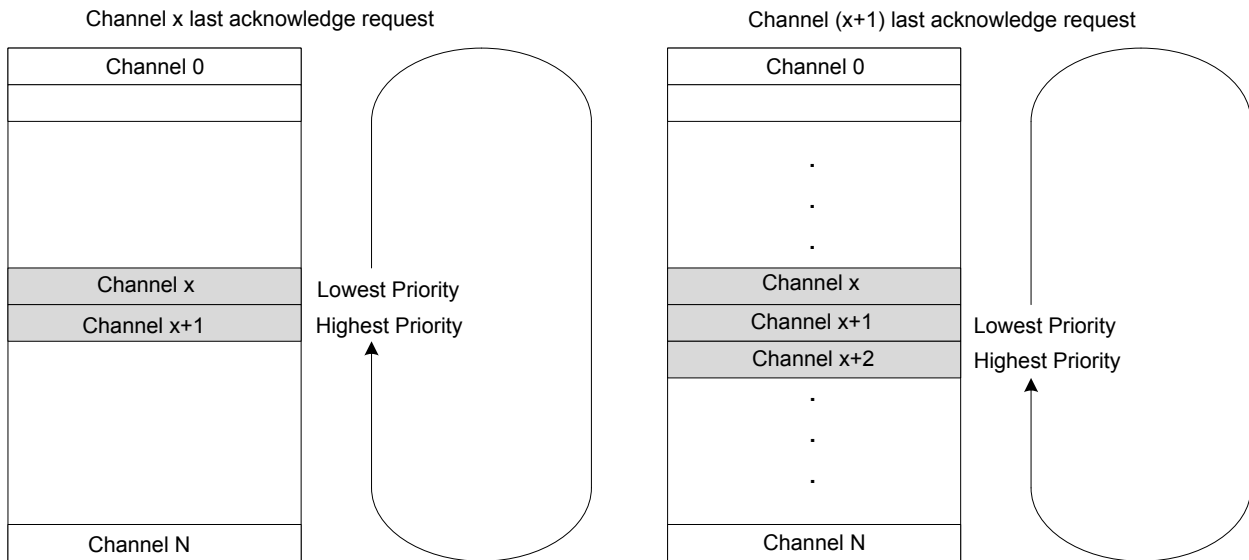
**Figure 22-5. Static Priority Scheduling**



*Dynamic Arbitration* within a priority level is selected by writing a '1' to [PRICTRL0.RRLVLENx](#).

The dynamic arbitration scheme in the DMAC is round-robin. 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 22-6](#). The channel number of the last channel being granted access as the active channel is stored in the Level x Channel Priority Number bit group in the Priority Control 0 register ([PRICTRL0.LVLPRIx](#)) for the corresponding priority level.

Figure 22-6. Dynamic (Round-Robin) Priority Scheduling



22.6.2.5 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 DMA Block Diagram section) 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. For a new block transfer, the transfer descriptor will be fetched from the descriptor memory section ([BASEADDR](#)); For an ongoing block transfer, the descriptor will be fetched from 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 the section on [Addressing](#).

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 by the number of beats in a burst transfer, the optional output event Beat will be generated if configured and enabled, and the active channel will perform a new burst transfer. If a different DMA channel than the current active channel is granted access, 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.

When a block transfer has come to its end ([BTCNT](#) is zero), the Valid bit in the Block Transfer Control register will be cleared ([BTCTRL.VALID=0](#)) 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. After the last block transfer in a transaction, the Next Descriptor Address register ([DESCADDR](#)) 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.

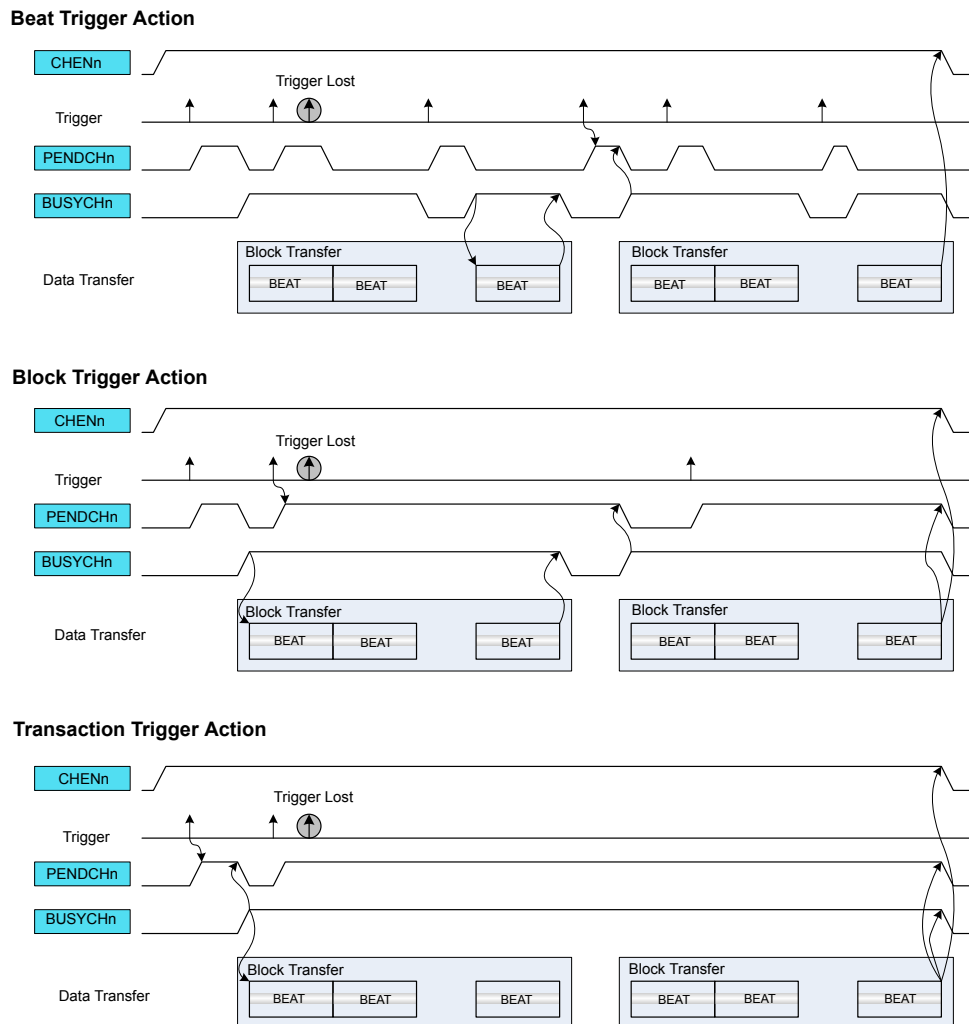
### 22.6.2.6 Transfer Triggers and Actions

A DMA transfer through a DMA channel can be started only when a DMA transfer request is detected, and the DMA channel has been granted access to the DMA. A transfer request can be triggered from software, from a 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 generates a request for a block transfer operation. If a single descriptor is defined for a channel, the channel is automatically disabled when a block transfer has been completed. If a list of linked descriptors is defined for a channel, the channel is automatically disabled when the last descriptor in the list is executed. If the list still has descriptors to execute, the channel will be waiting for the next block transfer trigger. When enabled again, the channel will wait for the next block transfer trigger. The trigger actions can also be configured to generate a request for a beat transfer (CHCTRLB.TRIGACT=0x2) or transaction transfer (CHCTRLB.TRIGACT=0x3) instead of a block transfer (CHCTRLB.TRIGACT=0x0).

Figure 22-7 shows an example where triggers are used with two linked block descriptors.

**Figure 22-7. Trigger Action and Transfers**



If the trigger source generates a transfer request for a channel during an ongoing transfer, the new transfer request will be kept pending (CHSTATUS.PEND=1), and the new transfer can start after the

ongoing one is done. Only one pending transfer can be kept per channel. If the trigger source generates more transfer requests while one is already pending, the additional ones 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 channel busy status flags are also available in the Busy Channels register (BUSYCH) in DMAC.

### 22.6.2.7 Addressing

Each block transfer needs to have both a source address and a destination address defined. The source address is set by writing the Transfer Source Address (SRCADDR) register, the destination address is set by writing the Transfer Destination Address (DSTADDR) 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=1). 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=1) and writing the desired step size in the Address Increment Step Size bit group in the Block Transfer Control register (BTCTRL.STEPSIZE). If BTCTRL.STEPSEL=0, the step size for the source incrementation will be the size of one beat.

When source address incrementation is configured (BTCTRL.SRCINC=1), SRCADDR is calculated as follows:

If BTCTRL.STEPSEL=1:

$$\text{SRCADDR} = \text{SRCADDR}_{\text{START}} + \text{BTCNT} \cdot (\text{BEATSIZE} + 1) \cdot 2^{\text{STEPSIZE}}$$

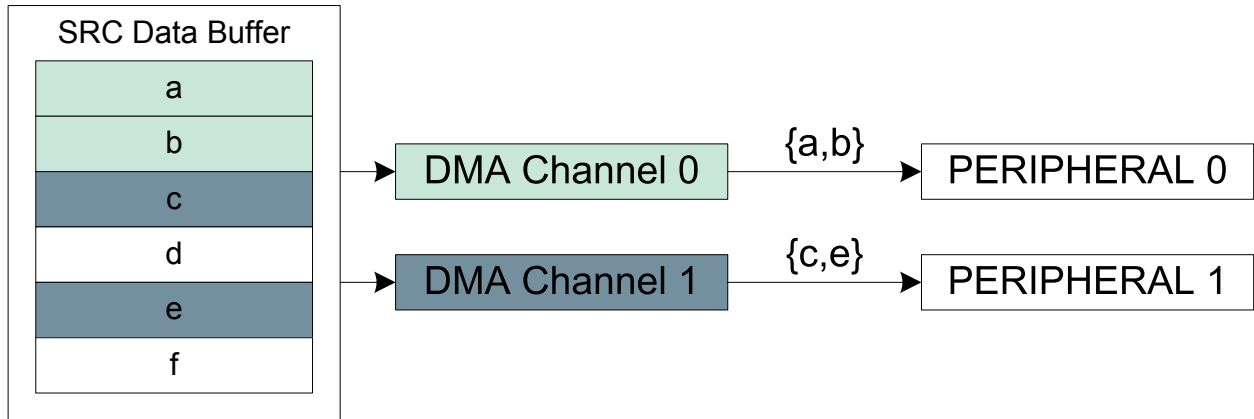
If BTCTRL.STEPSEL=0:

$$\text{SRCADDR} = \text{SRCADDR}_{\text{START}} + \text{BTCNT} \cdot (\text{BEATSIZE} + 1)$$

- SRCADDR<sub>START</sub> 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

The following figure shows an example where DMA channel 0 is configured to increment the source address by one beat after each beat transfer (BTCTRL.SRCINC=1), and DMA channel 1 is configured to increment the source address by two beats (BTCTRL.SRCINC=1, BTCTRL.STEPSEL=1, and BTCTRL.STEPSIZE=0x1). As the destination address for both channels are peripherals, destination incrementation is disabled (BTCTRL.DSTINC=0).

Figure 22-8. Source Address Increment



Incrementation for the destination address of a block transfer is enabled by setting the Destination Address Incrementation Enable bit in the Block Transfer Control register (`BTCTRL.DSTINC=1`). The step size of the incrementation is configurable by clearing `BTCTRL.STEPSEL=0` and writing `BTCTRL.STEPSIZE` to the desired step size. If `BTCTRL.STEPSEL=1`, the step size for the destination incrementation will be the size of one beat.

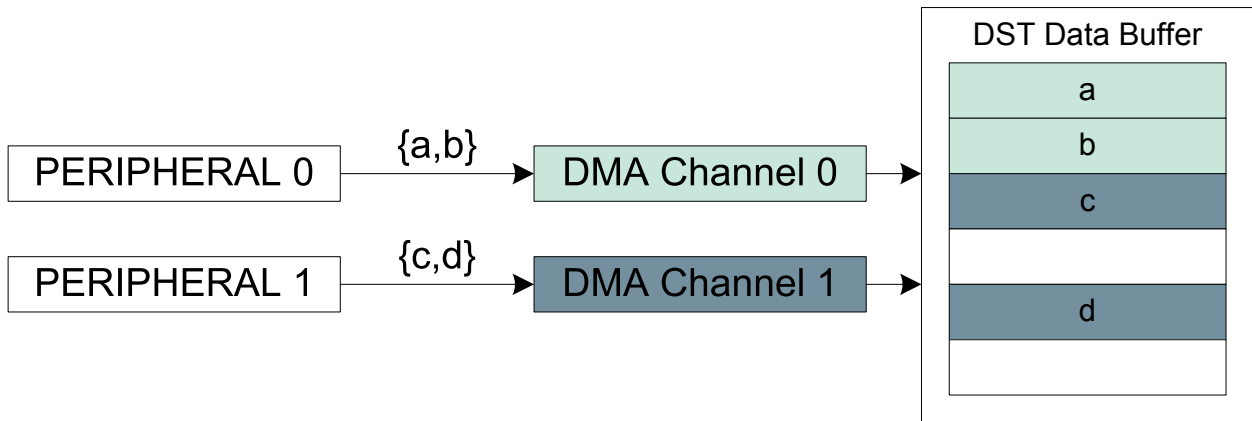
When the destination address incrementation is configured (`BTCTRL.DSTINC=1`), `DSTADDR` must be set and calculated as follows:

$DSTADDR = DSTADDR_{START} + BTCNT \cdot (BEATSIZE + 1) \cdot 2^{STEPSIZE}$	where <code>BTCTRL.STEPSEL</code> is zero
$DSTADDR = DSTADDR_{START} + BTCNT \cdot (BEATSIZE + 1)$	where <code>BTCTRL.STEPSEL</code> 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

The following figure shows an example where DMA channel 0 is configured to increment destination address by one beat (`BTCTRL.DSTINC=1`) and DMA channel 1 is configured to increment destination address by two beats (`BTCTRL.DSTINC=1`, `BTCTRL.STEPSEL=0`, and `BTCTRL.STEPSIZE=0x1`). As the source address for both channels are peripherals, source incrementation is disabled (`BTCTRL.SRCINC=0`).

Figure 22-9. Destination Address Increment



22.6.2.8 Error Handling

If a bus error is received from an 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 enabled, the optional transfer 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=0) or when the channel is resumed and the DMA fetches the next descriptor with null address (DESCADDR=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, the optional suspend interrupt is generated.

22.6.3 Additional Features

22.6.3.1 Linked Descriptors

A transaction can consist of either a single block transfer or of several block transfers. When a transaction consists of several block transfers it is done with the help of linked descriptors.

Figure 22-3 illustrates 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 of 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 section Data Transmission.

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 that 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 linked list, the following actions must be performed:

1. Enable the Suspend interrupt for the DMA channel.
2. Enable the DMA channel.
3. Reserve memory space in SRAM to configure a new descriptor.
4. 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
5. Clear the VALID bit for the existing list and for the descriptor which has to be updated.
  6. Read [DESCADDR](#) from the Write-Back memory.
    - If the DMA has not already fetched the descriptor which requires changes (i.e., [DESCADDR](#) is wrong):
      - Update the [DESCADDR](#) location of the descriptor from the List
      - Optionally clear the Suspend block action
      - Set the descriptor VALID bit to '1'
      - 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 next descriptor address ([DESCRADDR](#)) in the write-back memory
      - 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 '1'
  7. Go to step 4 if needed.

### Adding a Descriptor Between Existing Descriptors

To insert a new descriptor 'C' between two existing descriptors ('A' and '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:
  - 2.1. Set the descriptor A VALID bit to '0'.
  - 2.2. Set the [DESCADDR](#) value of descriptor A to point to descriptor C instead of descriptor B.
  - 2.3. Set the [DESCADDR](#) value of descriptor C to point to descriptor B.
  - 2.4. Set the descriptor A VALID bit to '1'.
3. If DMA is executing descriptor A:
  - 3.1. Apply the software suspend command to the channel and
  - 3.2. Perform steps 2.1 through 2.4.
  - 3.3. Apply the software resume command to the channel.

### 22.6.3.2 Channel Suspend

The channel operation can be suspended at any time by software by writing a '1' to the Suspend command in the Command bit field of Channel Control B register (CHCTRLB.CMD). After the ongoing burst transfer is completed, the channel operation is suspended and the suspend command is automatically cleared.

When suspended, the Channel Suspend Interrupt flag in the Channel Interrupt Status and Clear register is set (CHINTFLAG.SUSP=1) and the optional suspend interrupt is generated.



By configuring the block action to suspend by writing Block Action bit group in the Block Transfer Control register (BTCTRL.BLOCKACT is 0x2 or 0x3), the DMA channel will be suspended after it has completed a block transfer. The DMA channel will be kept enabled and will be able to receive transfer triggers, but it will be removed from the arbitration scheme.

If an invalid transfer descriptor (BTCTRL.VALID=0) is fetched from SRAM, the DMA channel will be suspended, and the Channel Fetch Error bit in the Channel Status register(CHASTATUS.FERR) will be set.

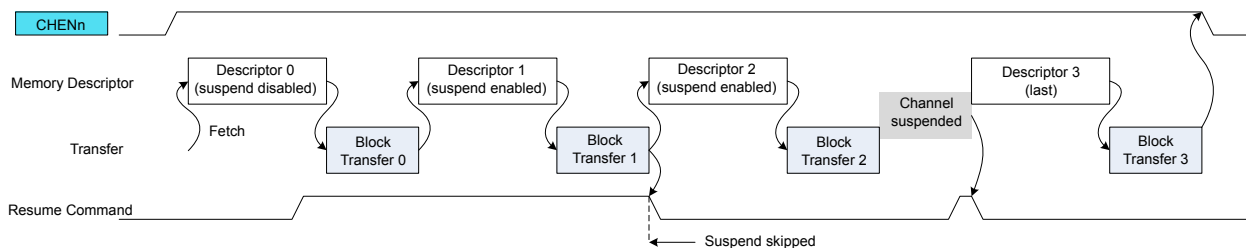
**Note:** Only enabled DMA channels can be suspended. If a channel is disabled when it is attempted to be suspended, the internal suspend command will be ignored.

For more details on transfer descriptors, refer to section [Transfer Descriptors](#).

**22.6.3.3 Channel Resume and Next Suspend Skip**

A channel operation can be resumed by software by setting the Resume command in the Command bit field of the 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 22-10. Channel Suspend/Resume Operation**



**22.6.3.4 Event Input Actions**

The event input actions are available only on the least significant DMA channels. For details on channels with event input support, refer to the in the Event system documentation.

Before using event input actions, the event controller must be configured first according to the following table, and the Channel Event Input Enable bit in the Channel Control B register (CHCTRLB.EVIE) must be written to '1'. Refer also to [Events](#).

**Table 22-1. Event Input Action**

Action	CHCTRLB.EVACT	CHCTRLB.TRGSRC
None	NOACT	-
Normal Transfer	TRIG	DISABLE
Conditional Transfer on Strobe	TRIG	any peripheral
Conditional Transfer	CTRIG	
Conditional Block Transfer	CBLOCK	
Channel Suspend	SUSPEND	
Channel Resume	RESUME	
Skip Next Block Suspend	SSKIP	

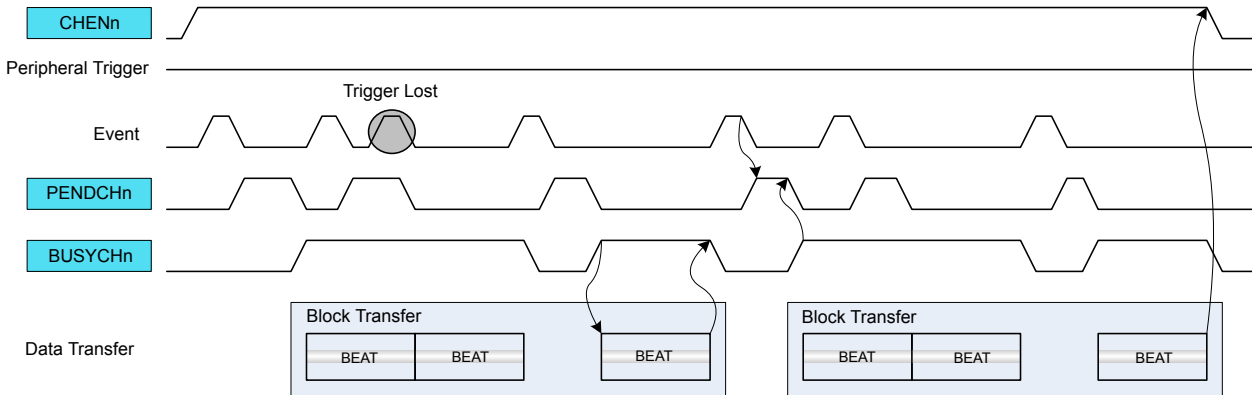
### Normal Transfer

The event input is used to trigger a beat or burst transfer on peripherals.

The event is acknowledged as soon as the event is received. When received, both the Channel Pending status bit in the Channel Status register ([CHSTATUS.PEND](#)) and the corresponding Channel n bit in the Pending Channels register ([PENDCH.PENDCHn](#)) are set. If the event is received while the channel is pending, the event trigger is lost.

The figure below shows an example where beat transfers are enabled by internal events.

**Figure 22-11. Beat Event Trigger Action**



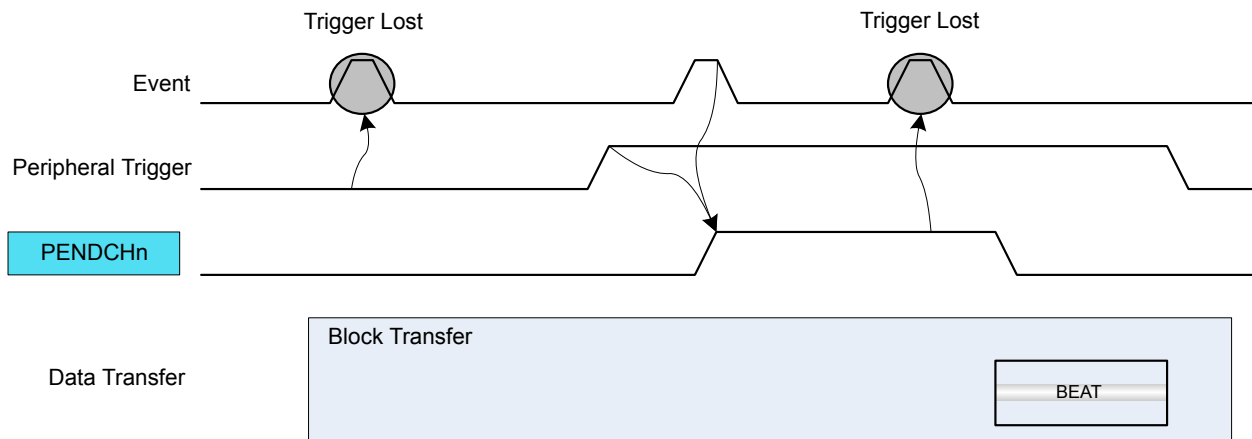
### Conditional Transfer on Strobe

The event input is used to trigger a transfer on peripherals with pending transfer requests. This event action is intended to be used with peripheral triggers, e.g. for timed communication protocols or periodic transfers between peripherals: only when the peripheral trigger coincides with the occurrence of a (possibly cyclic) event the transfer is issued.

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. the 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, both [CHSTATUS.PEND](#) and [PENDCH.PENDCHn](#) are set. A software trigger will now trigger a transfer.

The figure below shows an example where the peripheral beat transfer is started by a conditional strobe event action.

**Figure 22-12. Periodic Event with Beat Peripheral Triggers**



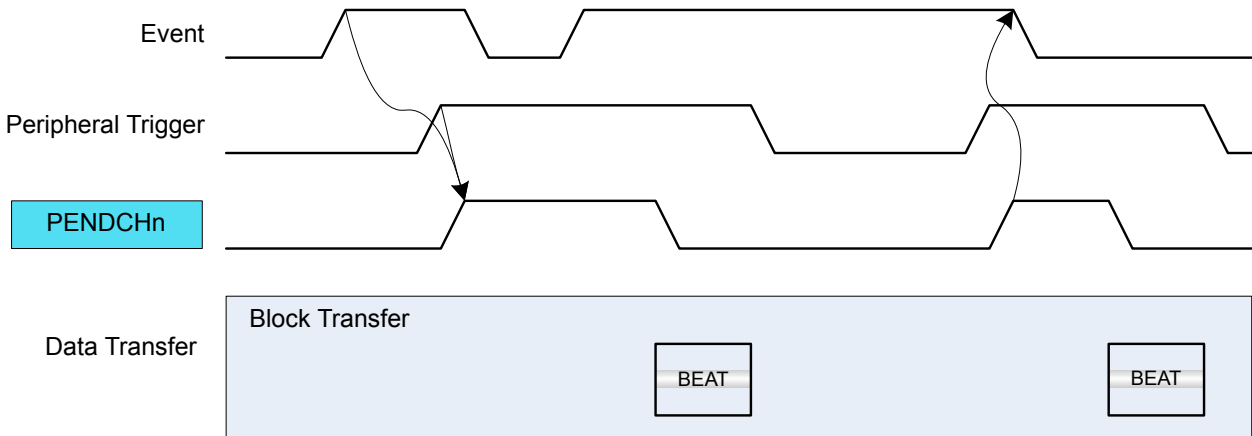
### Conditional Transfer

The event input is used to trigger a conditional transfer on peripherals with pending transfer requests. For example, this type of event can be used for peripheral-to-peripheral transfers, where one peripheral is the source of event and the second peripheral is the source of the trigger.

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](#)), the respective Pending Channel n Bit in the Pending Channels register is set ([PENDCH.PENDCHn](#)), and the event is acknowledged. A software trigger will now trigger a transfer.

The figure below shows an example where conditional event is enabled with peripheral beat trigger requests.

**Figure 22-13. Conditional Event with Beat Peripheral Triggers**



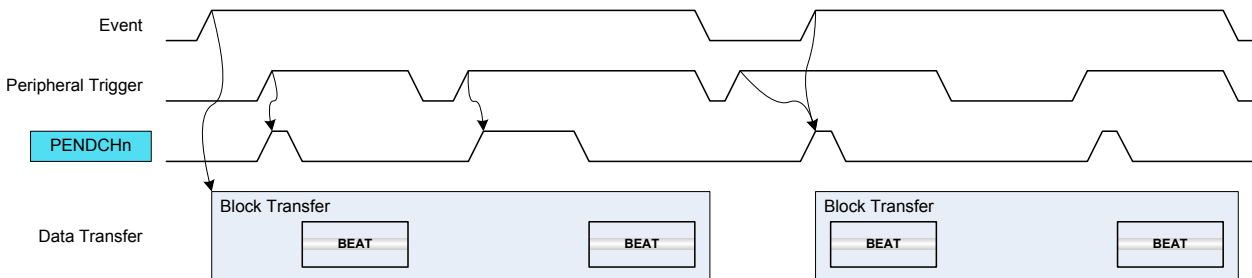
### Conditional Block Transfer

The event input is used to trigger a conditional block transfer on peripherals.

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.

The figure below shows an example where conditional event block transfer is started with peripheral beat trigger requests.

**Figure 22-14. Conditional Block Transfer with Beat Peripheral Triggers**



### Channel Suspend

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](#).

**Channel Resume**

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 refer to [Channel Suspend](#).

**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.

**Related Links**

[USER](#)

**22.6.3.5 Event Output Selection**

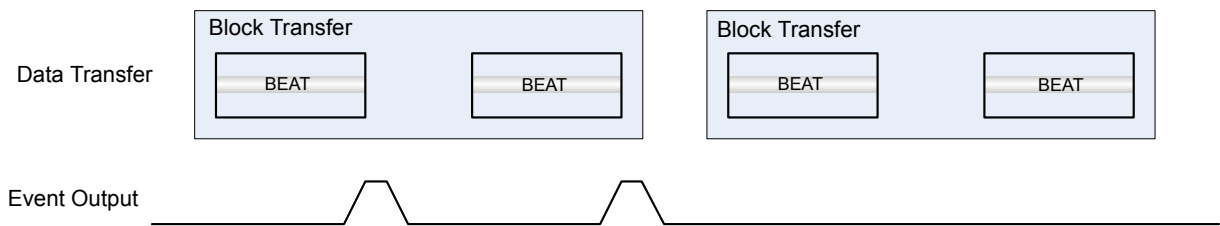
Event output selection is available only for the least significant DMA channels. The pulse width of an event output from a channel is one AHB clock cycle.

The output of channel events is enabled by writing a '1' to the Channel Event Output Enable bit in the Control B register ([CHCTRLB.EVOE](#)). The event output cause is selected by writing to the Event Output Selection bits in the Block Transfer Control register ([BTCTRL.EVOSEL](#)). It is possible to generate events after each block transfer ([BTCTRL.EVOSEL=0x1](#)) or beat transfer ([BTCTRL.EVOSEL=0x3](#)). To enable an event being generated when a transaction is complete, the block event selection must be set in the last transfer descriptor only.

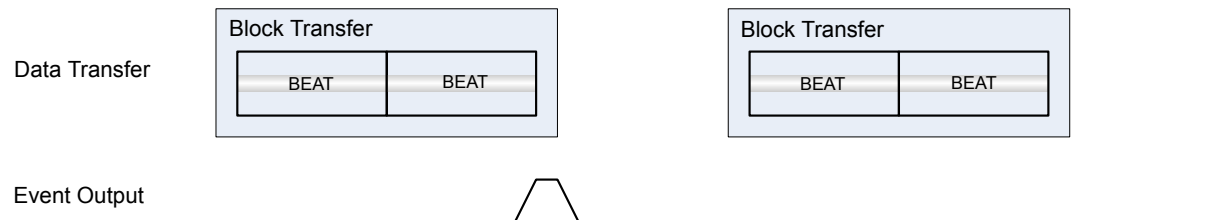
[Figure 22-15](#) shows an example where the event output generation is enabled in the first block transfer, and disabled in the second block.

**Figure 22-15. Event Output Generation**

**Beat Event Output**



**Block Event Output**



### 22.6.3.6 Aborting Transfers

Transfers on any channel can be aborted gracefully by software by disabling the corresponding DMA channel. It is also possible to abort all ongoing or pending transfers by disabling the DMAC.

When a DMA channel disable request or DMAC disable request is detected:

- Ongoing transfers of the active channel will be disabled when the ongoing beat transfer 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 is cleared (CHCTRLA.ENABLE=0) when the channel is disabled.

The corresponding DMAC Enable bit in the Control register is cleared (CTRL.DMAENABLE=0) when the entire DMAC module is disabled.

### 22.6.3.7 CRC Operation

A cyclic redundancy check (CRC) is an error detection technique used to find errors in data. It is commonly used to determine whether the data during a transmission, or data present in data and program 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 data is received, 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.

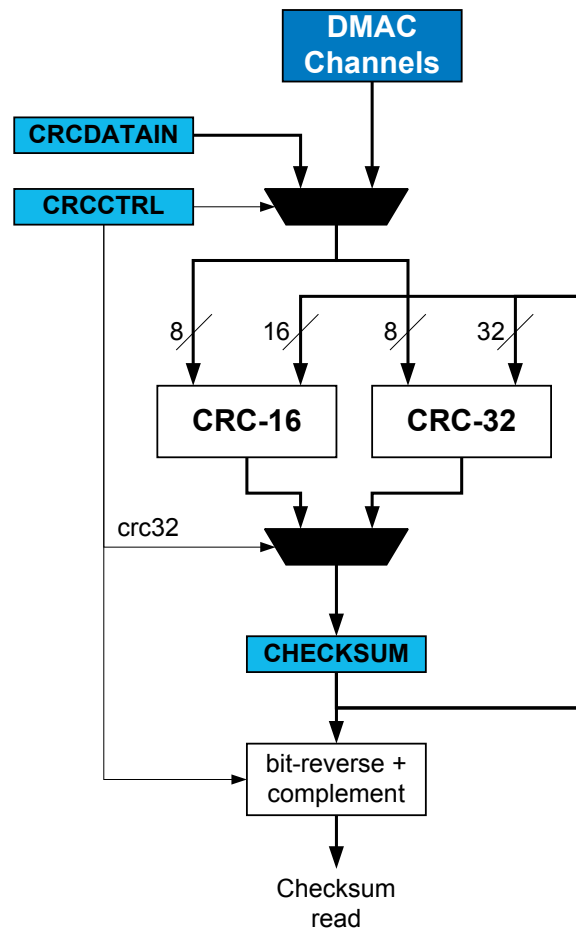
The CRC engine in DMAC supports two commonly used CRC polynomials: CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3). Typically, applying CRC-n (CRC-16 or CRC-32) to a data block of arbitrary length will detect any single alteration that is  $\leq n$  bits in length, and will detect the fraction  $1-2^{-n}$  of all longer error bursts.

- 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 engine can either be one of the DMA channels or the APB bus interface, and must be selected by writing to the CRC Input Source bits in the CRC Control register (CRCCTRL.CRCSRC). The CRC engine 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 22-16](#).

The CRC polynomial is selected by writing to the CRC Polynomial Type bit in the CRC Control register (CRCCTRL.CRCPOLY), the default is CRC-16. The CRC engine operates on byte only. When the DMA is used as data source for the CRC engine, the DMA channel beat size setting will be used. When used with APB bus interface, the application must select 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 engine will operate on the input data in a byte by byte manner.

Figure 22-16. 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 engine 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 engine.

**CRC using the I/O interface** Before using the CRC engine 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 engine 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 engine takes four cycles to calculate the CRC. The CRC complete is signaled by a set CRCBUSY bit in the CRCSTATUS register. New data can be written only when CRCBUSY flag is not set.

**22.6.4 DMA Operation**

Not applicable.

### 22.6.5 Interrupts

The DMAC channels have the following interrupt sources:

- Transfer Complete (TCMPL): Indicates that a block transfer is completed on the corresponding channel. Refer to [Data Transmission](#) 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](#) for details.
- Channel Suspend (SUSP): Indicates that the corresponding channel has been suspended. Refer to [Channel Suspend](#) and [Data Transmission](#) 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 setting the corresponding bit in the Channel Interrupt Enable Set register (CHINTENSET=1), and disabled by setting the corresponding bit in the Channel Interrupt Enable Clear register (CHINTENCLR=1).

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.

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:** Interrupts must be globally enabled for interrupt requests to be generated.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 22.6.6 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 Selection* for details.

Setting the Channel Control B Event Output Enable bit (CHCTRLB.EVOE=1) enables the corresponding output event configured in the Event Output Selection bit group in the Block Transfer Control register (BTCTRL.EVOSEL). Clearing CHCTRLB.EVOE=0 disables the corresponding output event.

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
- Increase Priority (INCPRI): increase channel priority

Setting the Channel Control B Event Input Enable bit (CHCTRLB.EVIE=1) enables the corresponding action on input event. Clearing 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*.

**Note:** Event input and outputs are not available for every channel. Refer to [Features](#) for more information.

#### Related Links

[EVSYS – Event System](#)

### 22.6.7 Sleep Mode Operation

Each DMA channel can be configured to operate in any sleep mode. To be able to run in standby, the RUNSTDBY bit in Channel Control A register (CHCTRLA.RUNSTDBY) must be written to '1'. The DMAC can wake up the device using interrupts from any sleep mode or perform actions through the Event System.

For channels with CHCTRLA.RUNSTDBY = 0, it is up to software to stop DMA transfers on these channels and wait for completion before going to standby mode using the following sequence:

1. Suspend the DMAC channels for which CHCTRLA.RUNSTDBY = 0.
2. Check the SYNCBUSY bits of registers accessed by the DMAC channels being suspended.
3. Go to sleep.
4. When the device wakes up, resume the suspended channels.

**Note:** In Stand-by Sleep mode, the DMAC can only access RAM when it is not back biased (PM.STDBYCFG.BBIASxx = 0x0)

### 22.6.8 Synchronization

Not applicable.



# ATSAMHAXEXXA

## DMAC – Direct Memory Access Controller

### 22.7 Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0						CRCENABLE	DMAENABLE	SWRST
		15:8					LVLENx3	LVLENx2	LVLENx1	LVLENx0
0x02	CRCCTRL	7:0						CRCPOLY[1:0]		CRCBEATSIZE[1:0]
		15:8						CRCSRC[5:0]		
0x04	CRCDATAIN	7:0						CRCDATAIN[7:0]		
		15:8						CRCDATAIN[15:8]		
		23:16						CRCDATAIN[23:16]		
		31:24						CRCDATAIN[31:24]		
0x08	CRCCHKSUM	7:0						CRCCHKSUM[7:0]		
		15:8						CRCCHKSUM[15:8]		
		23:16						CRCCHKSUM[23:16]		
		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						SWTRIGn[7:0]		
		15:8						SWTRIGn[11:8]		
		23:16								
		31:24								
0x14	PRICTRL0	7:0	RRLVLEN0					LVLPRIO[3:0]		
		15:8	RRLVLEN1					LVLPRIO[3:0]		
		23:16	RRLVLEN2					LVLPRIO[3:0]		
		31:24	RRLVLEN3					LVLPRIO[3:0]		
0x18 ... 0x1F	Reserved									
0x20	INTPEND	7:0						ID[3:0]		
		15:8	PEND	BUSY	FERR			SUSP	TCMPL	TERR
0x22 ... 0x23	Reserved									
0x24	INTSTATUS	7:0						CHINTn[7:0]		
		15:8						CHINTn[11:8]		
		23:16								
		31:24								
0x28	BUSYCH	7:0						BUSYCHn[7:0]		
		15:8						BUSYCHn[11:8]		
		23:16								
		31:24								
0x2C	PENDCH	7:0	PENDCH7	PENDCH6	PENDCH5	PENDCH4	PENDCH3	PENDCH2	PENDCH1	PENDCH0
		15:8					PENDCH11	PENDCH10	PENDCH9	PENDCH8
		23:16								

Offset	Name	Bit Pos.									
		31:24									
0x30	ACTIVE	7:0				LVLEXx	LVLEXx	LVLEXx	LVLEXx		
		15:8	ABUSY			ID[4:0]					
		23:16	BTCNT[7:0]								
		31:24	BTCNT[15:8]								
0x34	BASEADDR	7:0									
		15:8									
		23:16									
		31:24									
0x38	WRBADDR	7:0									
		15:8									
		23:16									
		31:24									
0x3C ... 0x3E	Reserved										
0x3F	CHID	7:0				ID[3:0]					
0x40	CHCTRLA	7:0					ENABLE	SWRST			
0x41 ... 0x43	Reserved										
0x44	CHCTRLB	7:0		LVL[1:0]	EVOE	EVIE	EVACT[2:0]				
		15:8	TRIGSRC[5:0]								
		23:16	TRIGACT[1:0]								
		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			

## 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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#).

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 22.8.1 Control

**Name:** CTRL  
**Offset:** 0x00  
**Reset:** 0x00X0  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
					LVLENx3	LVLENx2	LVLENx1	LVLENx0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
						CRCENABLE	DMAENABLE	SWRST
Access						R/W	R/W	R/W
Reset						0	0	0

#### Bits 8, 9, 10, 11 – LVLENx Priority Level x Enable

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 the [Arbitration](#) section.

These bits are not enable-protected.

Value	Description
0	Transfer requests for Priority level x will not be handled.
1	Transfer requests for Priority level x will be handled.

#### Bit 2 – CRCENABLE CRC Enable

Writing a '0' to this bit will disable the CRC calculation when the CRC Status Busy flag is cleared (CRCSTATUS.CRCBUSY). The bit is zero when the CRC is disabled.

Writing a '1' to this bit will enable the CRC calculation.

Value	Description
0	The CRC calculation is disabled.
1	The CRC calculation is enabled.

#### Bit 1 – DMAENABLE DMA Enable

Setting this bit will enable the DMA module.

Writing a '0' to this bit will disable the DMA module. When writing a '0' 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.

This bit is not enable-protected.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

**Bit 0 – SWRST** Software Reset

Writing a '0' to this bit has no effect.

Writing a '1' to this bit when both the DMAC and the CRC module are disabled (DMAENABLE and CRCENABLE are '0') 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.

Value	Description
0	There is no Reset operation ongoing.
1	A Reset operation is ongoing.

### 22.8.2 CRC Control

**Name:** CRCCTRL  
**Offset:** 0x02  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

	Bit	15	14	13	12	11	10	9	8
		CRCSRC[5:0]							
Access				R/W	R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0	0
	Bit	7	6	5	4	3	2	1	0
		CRCPOLY[1:0]				CRCBEATSIZE[1:0]			
Access						R/W	R/W	R/W	R/W
Reset						0	0	0	0

#### Bits 13:8 – CRCSRC[5:0] CRC Input Source

These bits select the input source for generating the CRC, as shown in the table below. 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.

Value	Name	Description
0x00	NOACT	No action
0x01	IO	I/O interface
0x02-0x1F	-	Reserved
0x20	CHN	DMA channel 0
0x21	CHN	DMA channel 1
0x22	CHN	DMA channel 2
0x23	CHN	DMA channel 3
0x24	CHN	DMA channel 4
0x25	CHN	DMA channel 5
0x26	CHN	DMA channel 6
0x27	CHN	DMA channel 7
0x28	CHN	DMA channel 8
0x29	CHN	DMA channel 9
0x2A	CHN	DMA channel 10
0x2B	CHN	DMA channel 11
0x2C	CHN	DMA channel 12
0x2D	CHN	DMA channel 13
0x2E	CHN	DMA channel 14
0x2F	CHN	DMA channel 15
0x30	CHN	DMA channel 16
0x31	CHN	DMA channel 17
0x32	CHN	DMA channel 18

Value	Name	Description
0x33	CHN	DMA channel 19
0x34	CHN	DMA channel 20
0x35	CHN	DMA channel 21
0x36	CHN	DMA channel 22
0x37	CHN	DMA channel 23
0x38	CHN	DMA channel 24
0x39	CHN	DMA channel 25
0x3A	CHN	DMA channel 26
0x3B	CHN	DMA channel 27
0x3C	CHN	DMA channel 28
0x3D	CHN	DMA channel 29
0x3E	CHN	DMA channel 30
0x3F	CHN	DMA channel 31

### Bits 3:2 – CRCPOLY[1:0] CRC Polynomial Type

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 the table below.

Value	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.

Value	Name	Description
0x0	BYTE	8-bit bus transfer
0x1	HWORDED	16-bit bus transfer
0x2	WORD	32-bit bus transfer
0x3		Reserved

### 22.8.3 CRC Data Input

**Name:** CRCDATAIN  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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. A new CRC Checksum is ready (CRCBEAT+ 1) clock cycles after the CRCDATAIN register is written.

### 22.8.4 CRC Checksum

**Name:** CRCCHKSUM  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

The CRCCHKSUM represents the 16- or 32-bit checksum value and the generated CRC. The register is reset to zero by default, but it is possible to reset all bits to one by writing the CRCCHKSUM register directly. It is possible to write this register only when the CRC module is disabled. If CRC-32 is selected and the CRC Status Busy flag is cleared (i.e., CRC generation is completed or aborted), the bit reversed (bit 31 is swapped with bit 0, bit 30 with bit 1, etc.) and complemented result will be read from CRCCHKSUM. If CRC-16 is selected or the CRC Status Busy flag is set (i.e., CRC generation is ongoing), CRCCHKSUM will contain the actual content.

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

#### 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.



### 22.8.5 CRC Status

**Name:** CRCSTATUS  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### 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.

When running CRC-32 and appending the checksum at the end of the packet (as little endian), the final checksum should be 0x2144df1c, and not zero. However, if the checksum is complemented before it is appended (as little endian) to the data, the final result in the checksum register will be zero. See the description of CRCCHKSUM to read out different versions of the checksum.

#### Bit 0 – CRCBUSY CRC Module Busy

This flag is cleared by writing a one to it when used with I/O interface. When used with a DMA channel, the bit is set when the corresponding DMA channel is enabled, and cleared when the corresponding DMA channel is disabled. This register bit cannot be cleared by the application when the CRC is used with a DMA channel.

This bit is set when a source configuration is selected and as long as the source is using the CRC module.

### 22.8.6 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x0D  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### 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.

Value	Description
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.

22.8.7 Quality of Service Control

**Name:** QOSCTRL  
**Offset:** 0x0E  
**Reset:** 0x2A  
**Property:** PAC Write-Protection

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

**Bits 5:4 – DQOS[1:0]** Data Transfer Quality of Service

These bits define the memory priority access during the data transfer operation.

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.

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.

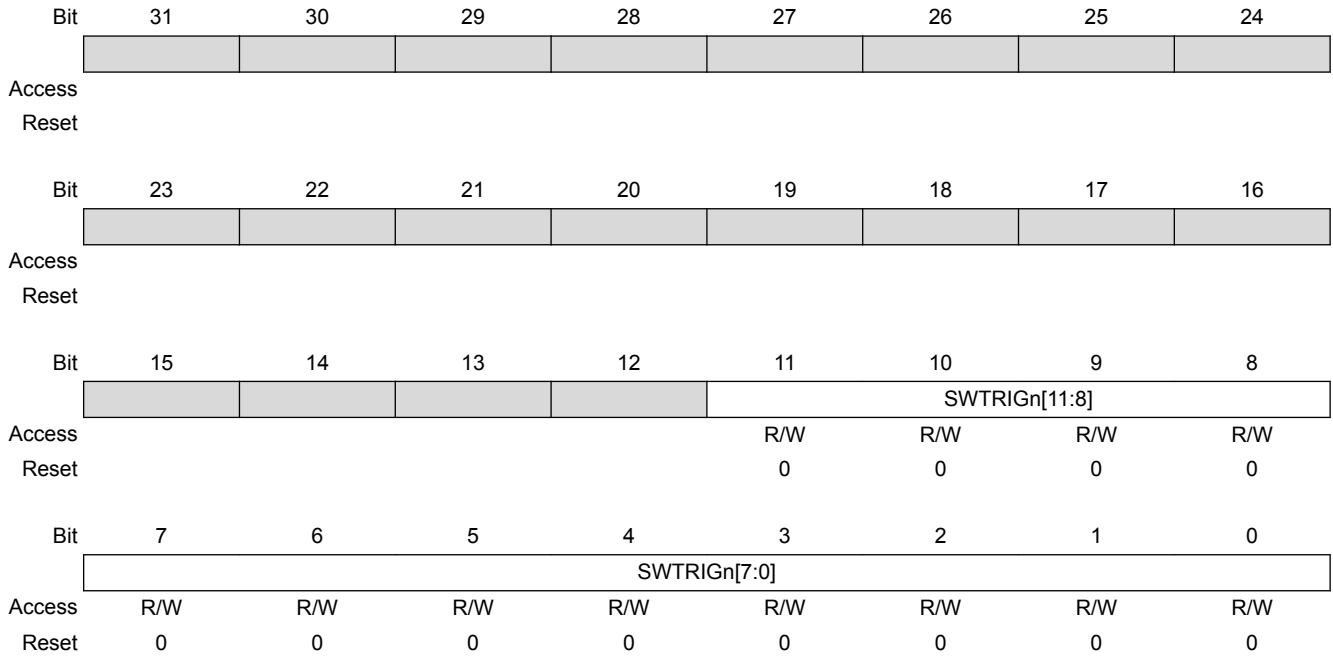
WRBQOS[1:0]	Name	Description
0x0	DISABLE	Background (no sensitive operation)
0x1	LOW	Sensitive Bandwidth
0x2	MEDIUM	Sensitive Latency
0x3	HIGH	Critical Latency

**Related Links**

[SRAM Quality of Service](#)

### 22.8.8 Software Trigger Control

**Name:** SWTRIGCTRL  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



**Bits 11:0 – SWTRIGn[11:0]** Channel n Software Trigger [n = 11..0]

This bit is cleared when the Channel Pending bit in the Channel Status register ([CHSTATUS.PEND](#)) for the corresponding channel is either set, or by writing a '1' to it.

This bit is set if [CHSTATUS.PEND](#) is already '1' when writing a '1' to that bit.

Writing a '0' to this bit will clear the bit.

Writing a '1' to this bit will generate a DMA software trigger on channel x, if [CHSTATUS.PEND](#)=0 for channel x. [CHSTATUS.PEND](#) will be set and [SWTRIGn](#) will remain cleared.

22.8.9 Priority Control 0

**Name:** PRICTRL0  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	RRLVLEN3			LVLPRI3[3:0]				
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0
Bit	23	22	21	20	19	18	17	16
	RRLVLEN2			LVLPRI2[3:0]				
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0
Bit	15	14	13	12	11	10	9	8
	RRLVLEN1			LVLPRI1[3:0]				
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0
Bit	7	6	5	4	3	2	1	0
	RRLVLEN0			LVLPRI0[3:0]				
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

**Bit 31 – RRLVLEN3** Level 3 Round-Robin Arbitration Enable

This bit controls which arbitration scheme is selected for DMA channels with priority level 3. For details on arbitration schemes, refer to [Arbitration](#).

Value	Description
0	Static arbitration scheme for channels with level 3 priority.
1	Round-robin arbitration scheme for channels with level 3 priority.

**Bits 27:24 – LVLPRI3[3:0]** Level 3 Channel Priority Number

When round-robin arbitration is enabled (PRICTRL0.RRLVLEN3=1) 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=0) for priority level 3, and the value of this bit group is non-zero, it will not affect the static priority scheme.

This bit group is not reset when round-robin arbitration gets disabled (PRICTRL0.RRLVLEN3 written to '0').

**Bit 23 – RRLVLEN2** Level 2 Round-Robin Arbitration Enable

This bit controls which arbitration scheme is selected for DMA channels with priority level 2. For details on arbitration schemes, refer to [Arbitration](#).

Value	Description
0	Static arbitration scheme for channels with level 2 priority.
1	Round-robin arbitration scheme for channels with level 2 priority.

### Bits 19:16 – LVLPR12[3:0] Level 2 Channel Priority Number

When round-robin arbitration is enabled (PRICTRL0.RRLVLEN2=1) 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=0) for priority level 2, and the value of this bit group is non-zero, it will not affect the static priority scheme.

This bit group is not reset when round-robin arbitration gets disabled (PRICTRL0.RRLVLEN2 written to '0').

### Bit 15 – RRLVLEN1 Level 1 Round-Robin Scheduling Enable

For details on arbitration schemes, refer to [Arbitration](#).

Value	Description
0	Static arbitration scheme for channels with level 1 priority.
1	Round-robin arbitration scheme for channels with level 1 priority.

### Bits 11:8 – LVLPR11[3:0] Level 1 Channel Priority Number

When round-robin arbitration is enabled (PRICTRL0.RRLVLEN1=1) 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=0) for priority level 1, and the value of this bit group is non-zero, it will not affect the static priority scheme.

This bit group is not reset when round-robin arbitration gets disabled (PRICTRL0.RRLVLEN1 written to '0').

### Bit 7 – RRLVLEN0 Level 0 Round-Robin Scheduling Enable

For details on arbitration schemes, refer to [Arbitration](#).

Value	Description
0	Static arbitration scheme for channels with level 0 priority.
1	Round-robin arbitration scheme for channels with level 0 priority.

### Bits 3:0 – LVLPR10[3:0] Level 0 Channel Priority Number

When round-robin arbitration is enabled (PRICTRL0.RRLVLEN0=1) 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=0) for priority level 0, and the value of this bit group is non-zero, it will not affect the static priority scheme.

This bit group is not reset when round-robin arbitration gets disabled (PRICTRL0.RRLVLEN0 written to '0').

### 22.8.10 Interrupt Pending

**Name:** INTPEND  
**Offset:** 0x20  
**Reset:** 0x0000  
**Property:** -

This register allows the user to identify the lowest DMA channel with pending interrupt.

Bit	15	14	13	12	11	10	9	8
	PEND	BUSY	FERR			SUSP	TCMPL	TERR
Access	R	R	R			R/W	R/W	R/W
Reset	0	0	0			0	0	0
Bit	7	6	5	4	3	2	1	0
					ID[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### Bit 15 – PEND Pending

This bit will read '1' when the channel selected by Channel ID field (ID) is pending.

#### Bit 14 – BUSY Busy

This bit will read '1' when the channel selected by Channel ID field (ID) is busy.

#### Bit 13 – FERR Fetch Error

This bit will read '1' when the channel selected by Channel ID field (ID) fetched an invalid descriptor.

#### Bit 10 – SUSP Channel Suspend

This bit will read '1' when the channel selected by Channel ID field (ID) has pending Suspend interrupt.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Channel ID (ID) Suspend interrupt flag.

#### Bit 9 – TCMPL Transfer Complete

This bit will read '1' when the channel selected by Channel ID field (ID) has pending Transfer Complete interrupt.

Writing a '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' to this bit will clear the Channel ID (ID) Transfer Error interrupt flag.

**Bits 3:0 – ID[3:0]** 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.



### 22.8.11 Interrupt Status

**Name:** INTSTATUS  
**Offset:** 0x24  
**Reset:** 0x00000000  
**Property:** -

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

**Bits 11:0 – CHINTn[11:0]** Channel n Pending Interrupt [n=11..0]

This bit is set when Channel n has a pending interrupt/the interrupt request is received.

This bit is cleared when the corresponding Channel n interrupts are disabled or the interrupts sources are cleared.

### 22.8.12 Busy Channels

**Name:** BUSYCH  
**Offset:** 0x28  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	[Greyed out]							
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	[Greyed out]							
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
	[Greyed out]				BUSYCHn[11:8]			
Access					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	BUSYCHn[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

**Bits 11:0 – BUSYCHn[11:0]** Busy Channel n [x=11..0]

This bit is cleared when the channel trigger action for DMA channel n is complete, when a bus error for DMA channel n is detected, or when DMA channel n is disabled.

This bit is set when DMA channel n starts a DMA transfer.

### 22.8.13 Pending Channels

**Name:** PENDING  
**Offset:** 0x2C  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
					PENDCH11	PENDCH10	PENDCH9	PENDCH8
Access					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	PENDCH7	PENDCH6	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 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – PENDCH** Pending Channel n [n=11..0]

This bit is cleared when trigger execution defined by channel trigger action settings for DMA channel n is started, when a bus error for DMA channel n is detected or when DMA channel n is disabled. For details on trigger action settings, refer to CHCTRLB.TRIGACT.

This bit is set when a transfer is pending on DMA channel n.

### 22.8.14 Active Channel and Levels

**Name:** ACTIVE  
**Offset:** 0x30  
**Reset:** 0x00000000  
**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	
Reset	0			0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
					LVLEXx	LVLEXx	LVLEXx	LVLEXx	
Access					R	R	R	R	
Reset					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 bit is set when the next descriptor transfer count is read from the write-back memory section.

#### 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.

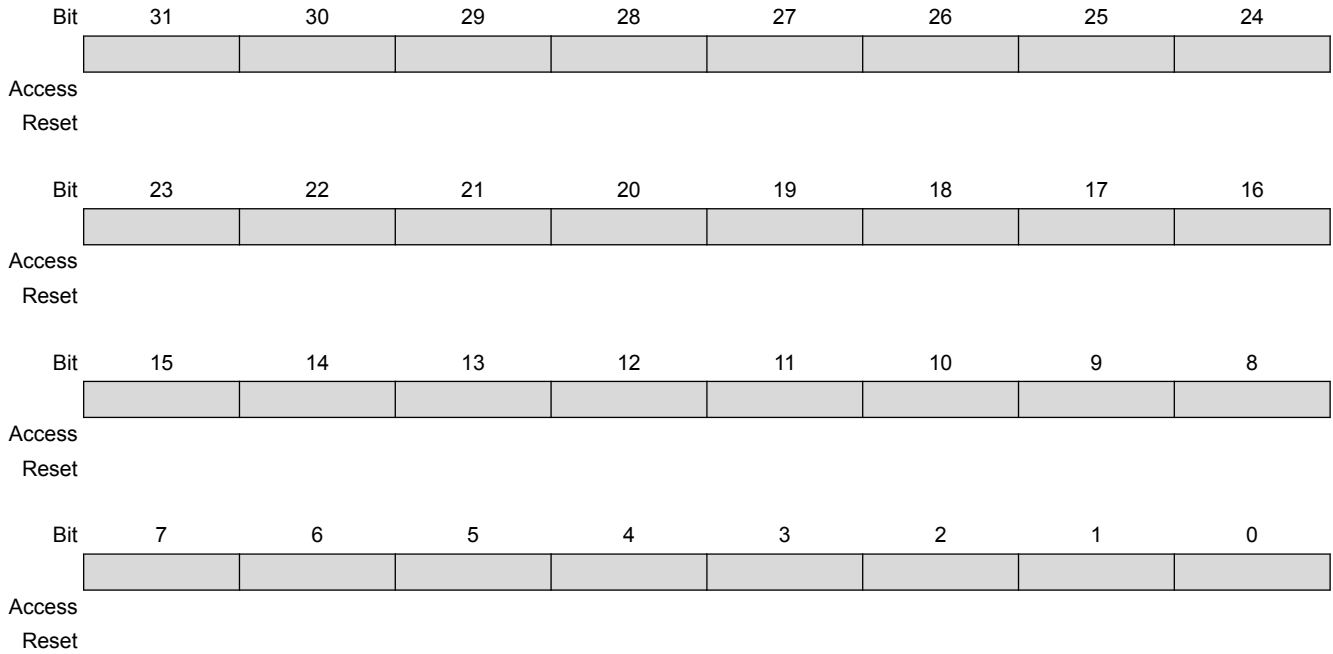
#### Bits 3,2,1,0 – LVLEXx Level x Channel Trigger Request Executing [x=3..0]

This bit is set when a level-x channel trigger request is executing or pending.

This bit is cleared when no request is pending or being executed.

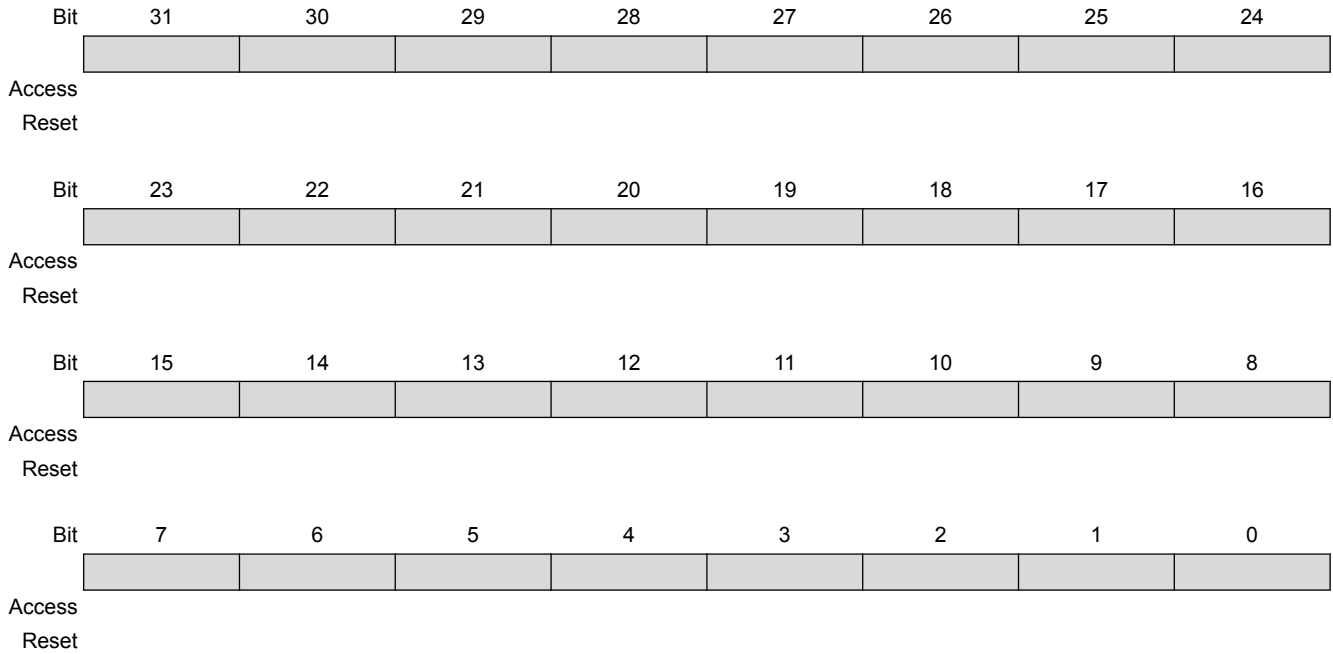
### 22.8.15 Descriptor Memory Section Base Address

**Name:** BASEADDR  
**Offset:** 0x34  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



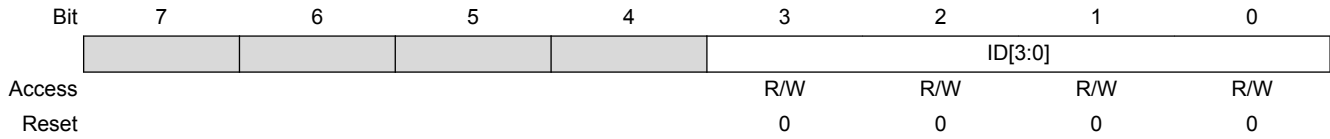
### 22.8.16 Write-Back Memory Section Base Address

**Name:** WRBADDR  
**Offset:** 0x38  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



### 22.8.17 Channel ID

**Name:** CHID  
**Offset:** 0x3F  
**Reset:** 0x00  
**Property:** -



#### Bits 3:0 – ID[3:0] Channel ID

These bits define the channel number that will be affected by the channel registers (CH\*). Before reading or writing a channel register, the channel ID bit group must be written first.

22.8.18 Channel Control A

**Name:** CHCTRLA  
**Offset:** 0x40  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

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

**Bit 1 – ENABLE** Channel Enable

Writing a '0' 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 '1' to this bit will enable the DMA channel.

This bit is not enable-protected.

Value	Description
0	DMA channel is disabled.
1	DMA channel is enabled.

**Bit 0 – SWRST** Channel Software Reset

Writing a '0' to this bit has no effect.

Writing a '1' 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 '1' to this bit will be ignored as long as ENABLE=1. This bit is automatically cleared when the reset is completed.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.



### 22.8.19 Channel Control B

**Name:** CHCTRLB  
**Offset:** 0x44  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

Bit	31	30	29	28	27	26	25	24
							CMD[1:0]	
Access							R/W	R/W
Reset							0	0
Bit	23	22	21	20	19	18	17	16
	TRIGACT[1:0]							
Access	R/W	R/W						
Reset	0	0						
Bit	15	14	13	12	11	10	9	8
			TRIGSRC[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			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/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

#### Bits 25:24 – CMD[1:0] Software Command

These bits define the software commands. Refer to [Channel Suspend](#) and [Channel Resume and Next Suspend Skip](#).

These bits are not enable-protected.

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.

TRIGACT[1:0]	Name	Description
0x0	BLOCK	One trigger required for each block transfer
0x1	-	Reserved

TRIGACT[1:0]	Name	Description
0x2	BEAT	One trigger required for each beat transfer
0x3	TRANSACTION	One trigger required for each transaction

### Bits 13:8 – TRIGSRC[5:0] 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](#) and [CHCTRLB.TRIGACT](#).

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	SERCOM2 RX	SERCOM2 RX Trigger
0x06	SERCOM2 TX	SERCOM2 TX Trigger
0x07	SERCOM3 RX	SERCOM3 RX Trigger
0x08	SERCOM3 TX	SERCOM3 TX Trigger
0x09	SERCOM4 RX	SERCOM4 RX Trigger
0x0A	SERCOM4 TX	SERCOM4 TX Trigger
0x0B	TCC0 OVF	TCC0 Overflow Trigger
0x0C	TCC0 MC0	TCC0 Match/Compare 0 Trigger
0x0D	TCC0 MC1	TCC0 Match/Compare 1 Trigger
0x0E	TCC0 MC2	TCC0 Match/Compare 2 Trigger
0x0F	TCC0 MC3	TCC0 Match/Compare 3 Trigger
0x10	TCC1 OVF	TCC1 Overflow Trigger
0x11	TCC1 MC0	TCC1 Match/Compare 0 Trigger
0x12	TCC1 MC1	TCC1 Match/Compare 1 Trigger
0x13	TCC2 OVF	TCC2 Overflow Trigger
0x14	TCC2 MC0	TCC2 Match/Compare 0 Trigger
0x15	TCC2 MC1	TCC2 Match/Compare 1 Trigger
0x16	TC0 OVF	TC0 Overflow Trigger
0x17	TC0 MC0	TC0 Match/Compare 0 Trigger
0x18	TC0 MC1	TC0 Match/Compare 1 Trigger
0x19	TC1 OVF	TC1 Overflow Trigger
0x1A	TC1 MC0	TC1 Match/Compare 0 Trigger
0x1B	TC1 MC1	TC1 Match/Compare 1 Trigger
0x1C	TC2 OVF	TC2 Overflow Trigger
0x1D	TC2 MC0	TC2 Match/Compare 0 Trigger
0x1E	TC2 MC1	TC2 Match/Compare 1 Trigger
0x1F	TC3 OVF	TC3 Overflow Trigger
0x20	TC3 MC0	TC3 Match/Compare 0 Trigger
0x21	TC3 MC1	TC3 Match/Compare 1 Trigger
0x22	TC4 OVF	TC4 Overflow Trigger
0x23	TC4 MC0	TC4 Match/Compare 0 Trigger
0x24	TC4 MC1	TC4 Match/Compare 1 Trigger
0x25	ADC RESRDY	ADC Result Ready Trigger

Value	Name	Description
0x26	DAC0 EMPTY	DAC0 Empty Trigger
0x27	DAC1 EMPTY	DAC1 Empty Trigger
0x28 - 0x2B	-	Reserved
0x2C	AES WR	AES Write Trigger
0x2D	AES RD	AES Read Trigger

### Bits 6:5 – LVL[1:0] Channel Arbitration Level

These bits define the arbitration level used for the DMA channel, where a high level has priority over a low level. For further details on arbitration schemes, refer to [Arbitration](#).

These bits are not enable-protected.

TRIGACT[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

### 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](#)).

This bit is available only for the least significant DMA channels. Refer to table: *User Multiplexer Selection* and *Event Generator Selection* of the Event System for details.

Value	Description
0	Channel event generation is disabled.
1	Channel event generation is enabled.

### Bit 3 – EVIE Channel Event Input Enable

This bit is available only for the least significant DMA channels. Refer to table: *User Multiplexer Selection* and *Event Generator Selection* of the Event System for details.

Value	Description
0	Channel event action will not be executed on any incoming event.
1	Channel event action will be executed on any incoming event.

### Bits 2:0 – EVACT[2:0] Event Input Action

These bits define the event input action, as shown below. The action is executed only if the corresponding EVIE bit in [CHCTRLB](#) register of the channel is set.

These bits are available only for the least significant DMA channels. Refer to table: *User Multiplexer Selection* and *Event Generator Selection* of the Event System for details.

<b>EVACT[2:0]</b>	<b>Name</b>	<b>Description</b>
0x0	NOACT	No action
0x1	TRIG	Normal Transfer and Conditional Transfer on Strobe 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

**Related Links**[USER](#)[CHANNEL](#)

**22.8.20 Channel Interrupt Enable Clear**

**Name:** CHINTENCLR  
**Offset:** 0x4C  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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. This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

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

**Bit 2 – SUSP** Channel Suspend Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Channel Suspend Interrupt Enable bit, which disables the Channel Suspend interrupt.

Value	Description
0	The Channel Suspend interrupt is disabled.
1	The Channel Suspend interrupt is enabled.

**Bit 1 – TCMPL** Channel Transfer Complete Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Channel Transfer Complete Interrupt Enable bit, which disables the Channel Transfer Complete interrupt.

Value	Description
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.

**Bit 0 – TERR** Channel Transfer Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Channel Transfer Error Interrupt Enable bit, which disables the Channel Transfer Error interrupt.

Value	Description
0	The Channel Transfer Error interrupt is disabled.
1	The Channel Transfer Error interrupt is enabled.

**22.8.21 Channel Interrupt Enable Set**

**Name:** CHINTENSET  
**Offset:** 0x4D  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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. This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

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

**Bit 2 – SUSP** Channel Suspend Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Channel Suspend Interrupt Enable bit, which enables the Channel Suspend interrupt.

Value	Description
0	The Channel Suspend interrupt is disabled.
1	The Channel Suspend interrupt is enabled.

**Bit 1 – TCMPL** Channel Transfer Complete Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Channel Transfer Complete Interrupt Enable bit, which enables the Channel Transfer Complete interrupt.

Value	Description
0	The Channel Transfer Complete interrupt is disabled.
1	The Channel Transfer Complete interrupt is enabled.

**Bit 0 – TERR** Channel Transfer Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Channel Transfer Error Interrupt Enable bit, which enables the Channel Transfer Error interrupt.

Value	Description
0	The Channel Transfer Error interrupt is disabled.
1	The Channel Transfer Error interrupt is enabled.

### 22.8.22 Channel Interrupt Flag Status and Clear

**Name:** CHINTFLAG  
**Offset:** 0x4E  
**Reset:** 0x00  
**Property:** -

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

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

#### Bit 2 – SUSP Channel Suspend

This flag is cleared by writing a '1' to it.

This flag 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 '0' to this bit has no effect.

Writing a '1' to this bit will clear the Channel Suspend interrupt flag for the corresponding channel.

For details on available software commands, refer to CHCTRLB.CMD.

For details on available event input actions, refer to CHCTRLB.EVACT.

For details on available block actions, refer to BTCTRL.BLOCKACT.

#### Bit 1 – TCMPL Channel Transfer Complete

This flag is cleared by writing a '1' to it.

This flag is set when a block transfer is completed and the corresponding interrupt block action is enabled.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Transfer Complete interrupt flag for the corresponding channel.

#### Bit 0 – TERR Channel Transfer Error

This flag is cleared by writing a '1' 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 '0' to this bit has no effect.

Writing a '1' to this bit will clear the Transfer Error interrupt flag for the corresponding channel.

### 22.8.23 Channel Status

**Name:** CHSTATUS  
**Offset:** 0x4F  
**Reset:** 0x00  
**Property:** -

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

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

#### Bit 2 – FERR Channel Fetch Error

This bit is cleared when a 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 completed, 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 the channel trigger action is started, when a bus error is detected or when the channel is disabled. For details on trigger action settings, refer to CHCTRLB.TRIGACT.

This bit is set when a transfer is pending on the DMA channel, as soon as the transfer request is received.



### 22.9 Register Summary - SRAM

Offset	Name	Bit Pos.							
0x00	BTCTRL	7:0				BLOCKACT[1:0]		EVOSEL[1:0]	VALID
		15:8	STEPSIZE[2:0]			STEPSEL	DSTINC	SRCINC	BEATSIZE[1:0]
0x02	BTCNT	7:0	BTCNT[7:0]						
		15:8	BTCNT[15:8]						
0x04	SRCADDR	7:0	SRCADDR[7:0]						
		15:8	SRCADDR[15:8]						
		23:16	SRCADDR[23:16]						
		31:24	SRCADDR[31:24]						
0x08	DSTADDR	7:0	DSTADDR[7:0]						
		15:8	DSTADDR[15:8]						
		23:16	DSTADDR[23:16]						
		31:24	DSTADDR[31:24]						
0x0C	DESCADDR	7:0	DESCADDR[7:0]						
		15:8	DESCADDR[15:8]						
		23:16	DESCADDR[23:16]						
		31:24	DESCADDR[31:24]						

### 22.10 Register Description - SRAM

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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#).

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 22.10.1 Block Transfer Control

**Name:** BTCTRL  
**Offset:** 0x00  
**Property:** -

The BTCTRL register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10

Bit	15	14	13	12	11	10	9	8
	STEPSIZE[2:0]			STEPSEL	DSTINC	SRCINC	BEATSIZE[1:0]	
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
				BLOCKACT[1:0]		EVOSEL[1:0]		VALID
Access								
Reset								

#### Bits 15:13 – STEPSIZE[2:0] Address Increment Step Size

These bits select the address increment step size. The setting apply to source or destination address, depending on STEPSEL setting.

Value	Name	Description
0x0	X1	Next ADDR = ADDR + (Beat size in byte) * 1
0x1	X2	Next ADDR = ADDR + (Beat size in byte) * 2
0x2	X4	Next ADDR = ADDR + (Beat size in byte) * 4
0x3	X8	Next ADDR = ADDR + (Beat size in byte) * 8
0x4	X16	Next ADDR = ADDR + (Beat size in byte) * 16
0x5	X32	Next ADDR = ADDR + (Beat size in byte) * 32
0x6	X64	Next ADDR = ADDR + (Beat size in byte) * 64
0x7	X128	Next ADDR = ADDR + (Beat size in byte) * 128

#### Bit 12 – STEPSEL Step Selection

This bit selects if source or destination addresses are using the step size settings.

Value	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

Writing a '0' to this bit will disable the destination address incrementation. The address will be kept fixed during the data transfer.

Writing a '1' 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.

Value	Description
0	The Destination Address Increment is disabled.
1	The Destination Address Increment is enabled.

### Bit 10 – SRCINC Source Address Increment Enable

Writing a '0' to this bit will disable the source address incrementation. The address will be kept fixed during the data transfer.

Writing a '1' 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.

Value	Description
0	The Source Address Increment is disabled.
1	The Source Address Increment is enabled.

### Bits 9:8 – BEATSIZE[1:0] Beat Size

These bits define the size of one beat. A beat is the size of one data transfer bus access, and the setting apply to both read and write accesses.

Value	Name	Description
0x0	BYTE	8-bit bus transfer
0x1	HWWORD	16-bit bus transfer
0x2	WORD	32-bit bus transfer
other		Reserved

### Bits 4:3 – BLOCKACT[1:0] Block Action

These bits define what actions the DMAC should take after a block transfer has completed.

BLOCKACT[1:0]	Name	Description
0x0	NOACT	Channel will be disabled if it is the last block transfer in the transaction
0x1	INT	Channel will be disabled if it is the last block transfer in the transaction 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.

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

Writing a '0' 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.

---

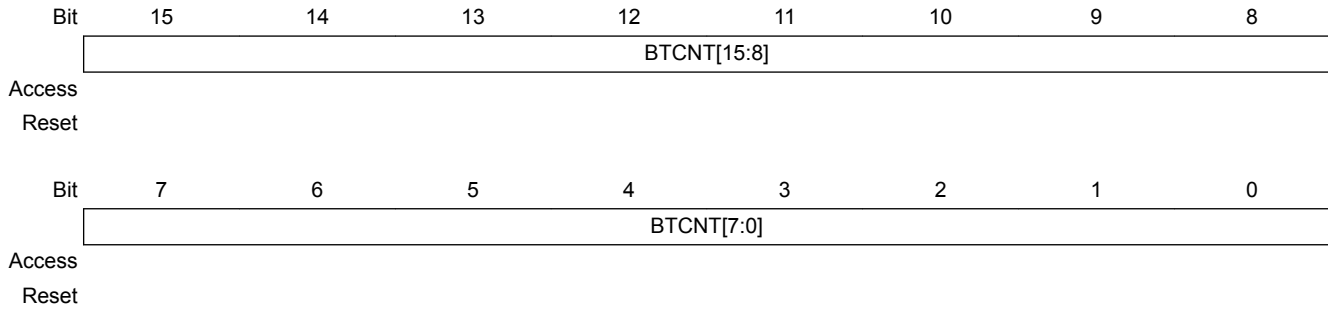
---

Value	Description
0	The descriptor is not valid.
1	The descriptor is valid.

### 22.10.2 Block Transfer Count

**Name:** BTCNT  
**Offset:** 0x02  
**Property:** -

The BTCNT register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10



**Bits 15:0 – BTCNT[15:0] Block Transfer Count**

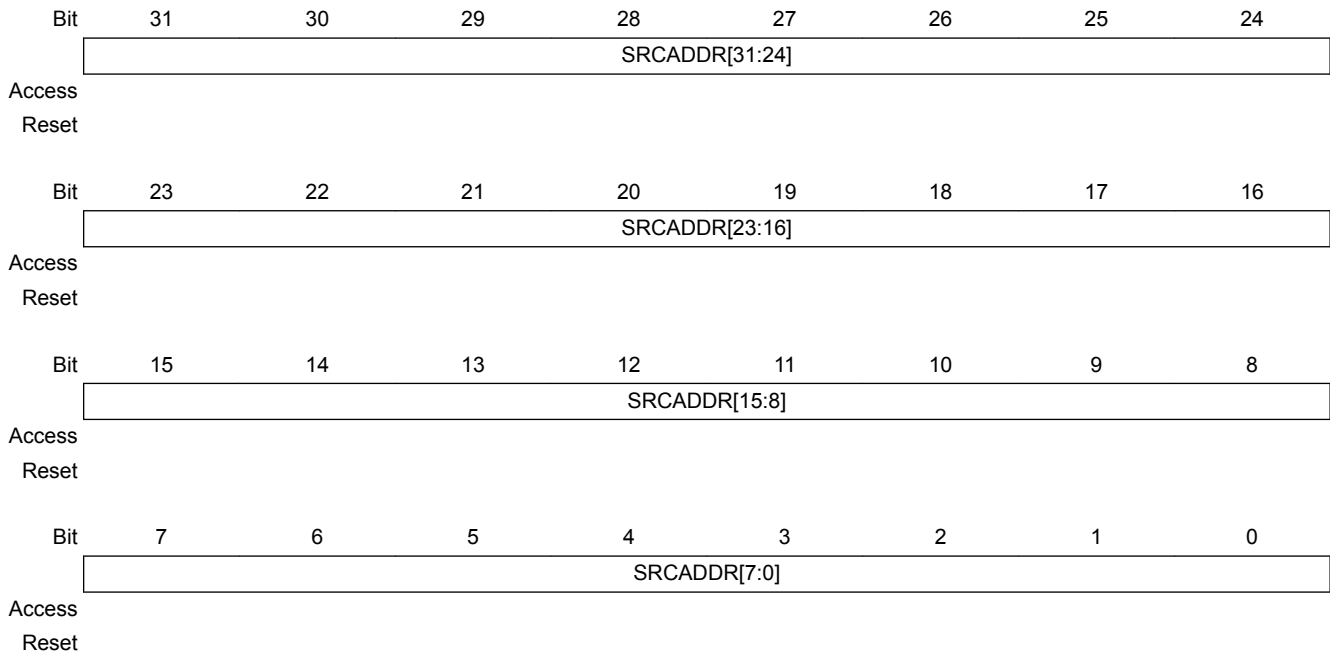
This bit group holds the 16-bit block transfer count.

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.

### 22.10.3 Block Transfer Source Address

**Name:** SRCADDR  
**Offset:** 0x04  
**Property:** -

The SRCADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10



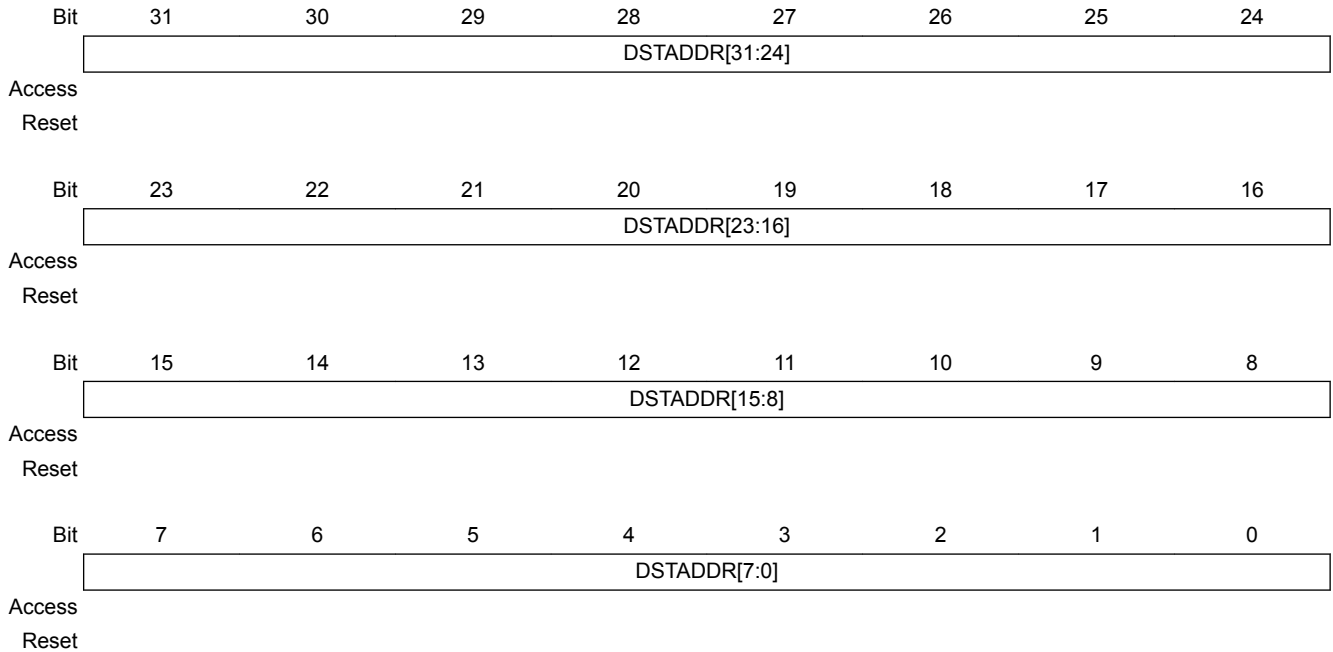
#### Bits 31:0 – SRCADDR[31:0] Transfer Source Address

This bit group holds the source address corresponding to the last beat transfer address in the block transfer.

### 22.10.4 Block Transfer Destination Address

**Name:** DSTADDR  
**Offset:** 0x08  
**Property:** -

The DSTADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10



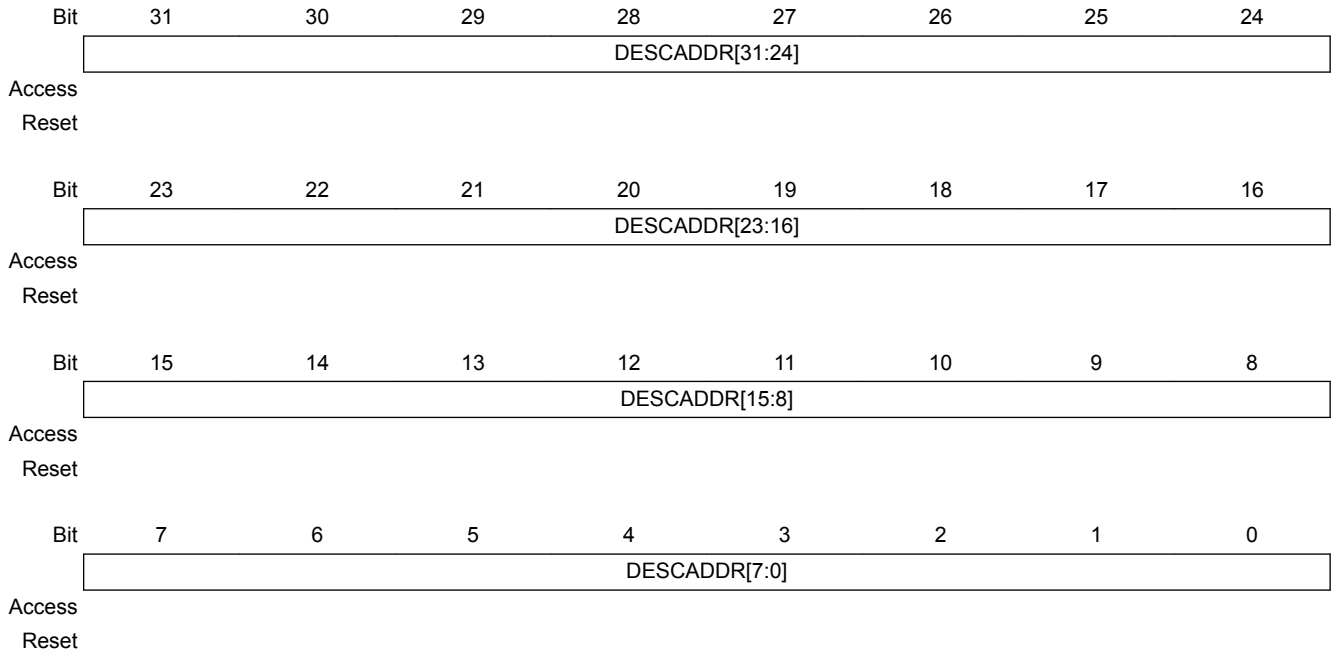
#### Bits 31:0 – DSTADDR[31:0] Transfer Destination Address

This bit group holds the destination address corresponding to the last beat transfer address in the block transfer.

### 22.10.5 Next Descriptor Address

**Name:** DESCADDR  
**Offset:** 0x0C  
**Property:** -

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.



## 23. EIC – External Interrupt Controller

### 23.1 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.

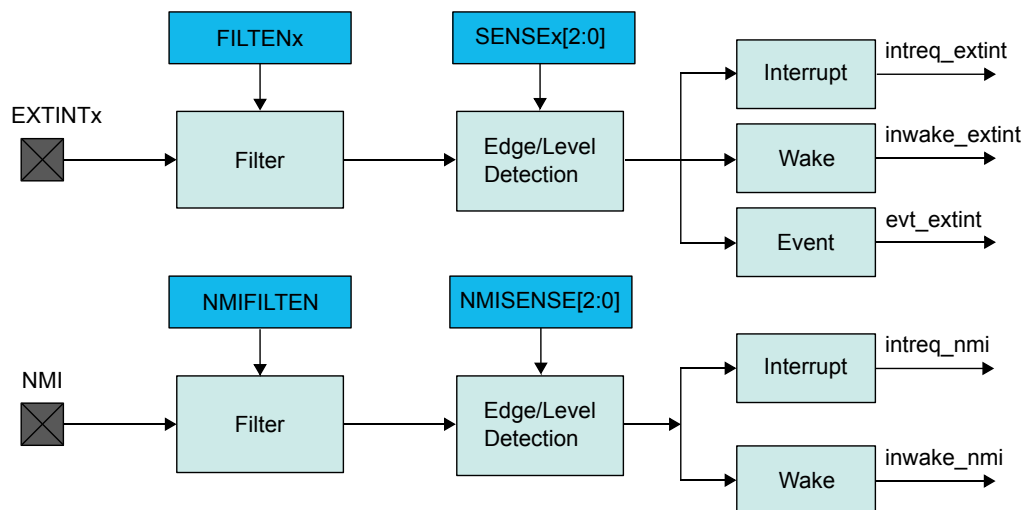
A separate non-maskable interrupt (NMI) is also supported. It has properties similar to the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other interrupt mode.

### 23.2 Features

- Up to 16 external pins (EXTINTx), plus one non-maskable pin (NMI)
- Dedicated, individually maskable interrupt for each pin
- 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 from EXTINTx

### 23.3 Block Diagram

Figure 23-1. EIC Block Diagram



### 23.4 Signal Description

Signal Name	Type	Description
EXTINT[15..0]	Digital Input	External interrupt pin
NMI	Digital Input	Non-maskable interrupt pin

One signal may be available on several pins.

#### Related Links

[I/O Multiplexing and Considerations](#)

### 23.5 Product Dependencies

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

#### 23.5.1 I/O Lines

Using the EIC's I/O lines requires the I/O pins to be configured.

#### Related Links

[PORT - I/O Pin Controller](#)

#### 23.5.2 Power Management

All interrupts are available down to STANDBY sleep mode, 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.

#### Related Links

[PM – Power Manager](#)

#### 23.5.3 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.

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*.

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](#) for further details.

#### Related Links

[Peripheral Clock Masking](#)

[GCLK - Generic Clock Controller](#)

#### 23.5.4 DMA

Not applicable.

### **23.5.5 Interrupts**

There are several interrupt request lines, at least 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.

The NMI interrupt request line is also connected to the interrupt controller, but does not require the interrupt to be configured.

#### **Related Links**

[Nested Vector Interrupt Controller](#)

### **23.5.6 Events**

The events are connected to the Event System. Using the events requires the Event System to be configured first.

#### **Related Links**

[EVSYS – Event System](#)

### **23.5.7 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.

### **23.5.8 Register Access Protection**

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

- Interrupt Flag Status and Clear register (INTFLAG)
- Non-Maskable Interrupt Flag Status and Clear register (NMIFLAG)

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

PAC write-protection does not apply to accesses through an external debugger.

#### **Related Links**

[PAC - Peripheral Access Controller](#)

### **23.5.9 Analog Connections**

Not applicable.

## **23.6 Functional Description**

### **23.6.1 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 GCLK\_EIC

**23.6.2 Basic Operation**

**23.6.2.1 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

To use NMI, GCLK\_EIC must be enabled after EIC configuration (NMICTRL).

**23.6.2.2 Enabling, Disabling and Resetting**

The EIC is enabled by writing a '1' the Enable bit in the Control register (CTRL.ENABLE). The EIC is disabled by writing CTRL.ENABLE to '0'.

The EIC is reset by setting 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 the CTRL register description for details.

**23.6.3 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 interrupt pins is configured by writing the Input Sense x bits in the Config n register (CONFIGn.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.

When the interrupt flag 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 n register (CONFIGn.FILTENx) is written to '1'. 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 23-1. Majority Vote Filter**

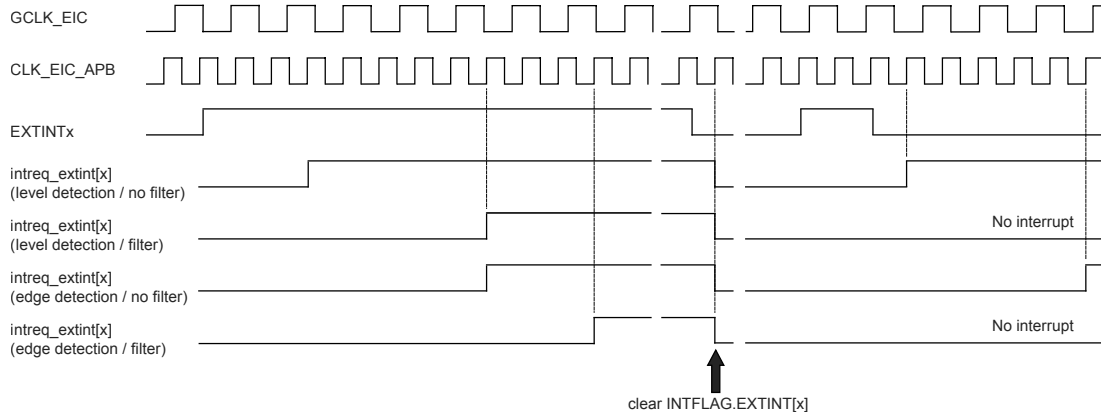
Samples [0, 1, 2]	Filter Output
[0,0,0]	0
[0,0,1]	0
[0,1,0]	0
[0,1,1]	1
[1,0,0]	0
[1,0,1]	1
[1,1,0]	1
[1,1,1]	1

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* for details). If level detection is enabled, GCLK\_EIC is not required, but interrupt and events can still be generated.

When an external interrupt is configured for level detection and when filtering is disabled, detection is done asynchronously. Asynchronous detection does not require GCLK\_EIC, but interrupt and events can still be generated. If filtering or edge detection is enabled, the EIC automatically requests GCLK\_EIC to operate. GCLK\_EIC must be enabled in the GCLK module.

**Figure 23-2. Interrupt Detections**



The detection delay depends on the detection mode.

**Table 23-2. Interrupt Latency**

Detection mode	Latency (worst case)
Level without filter	Three CLK_EIC_APB periods
Level with filter	Four GCLK_EIC periods + Three CLK_EIC_APB periods
Edge without filter	Four GCLK_EIC periods + Three CLK_EIC_APB periods
Edge with filter	Six GCLK_EIC periods + Three CLK_EIC_APB periods

### Related Links

[GCLK - Generic Clock Controller](#)

## 23.6.4 Additional Features

### 23.6.4.1 Non-Maskable Interrupt (NMI)

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). 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 '1' to the NMI Filter Enable bit (NMICTRL.NMIFILTEN).

If edge detection or filtering is required, enable GCLK\_EIC or CLK\_ULP32K.

NMI detection is enabled only by the NMICTRL.NMISENSE value, and the EIC is not required to be enabled.

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.

### 23.6.5 DMA Operation

Not applicable.

### 23.6.6 Interrupts

The EIC has the following interrupt sources:

- External interrupt pins (EXTINTx). See [Basic Operation](#).
- Non-maskable interrupt pin (NMI). See [Additional Features](#).

Each interrupt source has an associated interrupt flag. 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 setting the corresponding bit in the Interrupt Enable Set register (INTENSET=1), and disabled by setting the corresponding bit in the Interrupt Enable Clear register (INTENCLR=1).

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 interrupt request line for each external interrupt (EXTINTx) and one line for NMI. The user must read the INTFLAG (or NMIFLAG) register to determine which interrupt condition is present.

#### Note:

1. Interrupts must be globally enabled for interrupt requests to be generated.
2. If an external interrupts (EXTINT) is common on two or more I/O pins, only one will be active (the first one programmed).

#### Related Links

[Processor And Architecture](#)

### 23.6.7 Events

The EIC can generate the following output events:

- External event from pin (EXTINTx).

Setting an Event Output Control register (EVCTRL.EXTINTEO) enables the corresponding output event. Clearing this bit disables the corresponding output event. Refer to *Event System* for details on configuring the Event System.

When the condition on pin EXTINTx matches the configuration in the CONFIGn register, the corresponding event is generated, if enabled.

#### Related Links

[EVSYS – Event System](#)

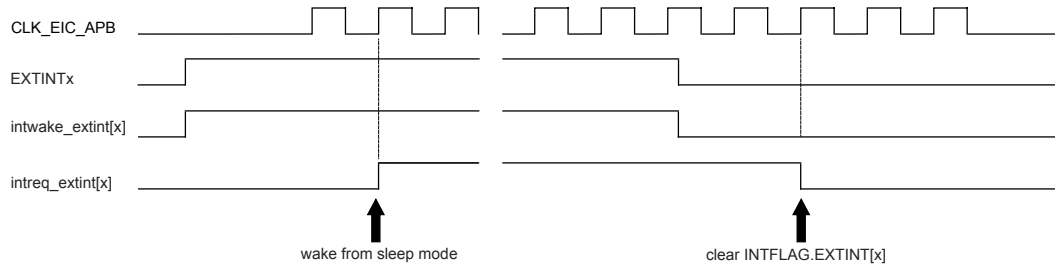
### 23.6.8 Sleep Mode Operation

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.

In sleep modes, an EXTINTx pin can wake up the device if the corresponding condition matches the configuration in CONFIGn register, and the corresponding bit in the Interrupt Enable Set register (INTENSET) is written to '1'. WAKEUP.WAKEUPEN[x]=1 can enable the wake-up from pin EXTINTx.

**Figure 23-3. Wake-Up Operation Example (High-Level Detection, No Filter, WAKEUPEN[x]=1)**



### 23.6.9 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

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 are synchronized when written:

- Software Reset bit in the Control register (CTRL.SWRST)
- Enable bit in the Control register (CTRL.ENABLE)

#### Related Links

[Register Synchronization](#)

### 23.7 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	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx
		15:8	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx
		23:16							EXTINTEOx	EXTINTEOx
		31:24								
0x08	INTENCLR	7:0	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx
		15:8	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx
		23:16							EXTINTx	EXTINTx
		31:24								
0x0C	INTENSET	7:0	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx
		15:8	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx
		23:16							EXTINTx	EXTINTx
		31:24								
0x10	INTFLAG	7:0	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx
		15:8	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx
		23:16							EXTINTx	EXTINTx
		31:24								
0x14	WAKEUP	7:0	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx
		15:8	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx
		23:16							WAKEUPENx	WAKEUPENx
		31:24								
0x18	CONFIG0	7:0	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		15:8	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		23:16	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		31:24	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
0x1C	CONFIG1	7:0	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		15:8	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		23:16	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		31:24	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
0x20	CONFIG2	7:0	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		15:8	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		23:16	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		
		31:24	FILTENx	SENSEx[2:0]			FILTENx	SENSEx[2:0]		

### 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.



# ATSAMHAXEXXA

## EIC – External Interrupt Controller

---

Some registers require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 23.8.1 Control

**Name:** CTRL  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** Write-Protected, Write-Synchronized

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

#### Bit 1 – ENABLE Enable

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.

Value	Description
0	The EIC is disabled.
1	The EIC is enabled.

#### 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 EIC to their initial state, and the EIC will be disabled.

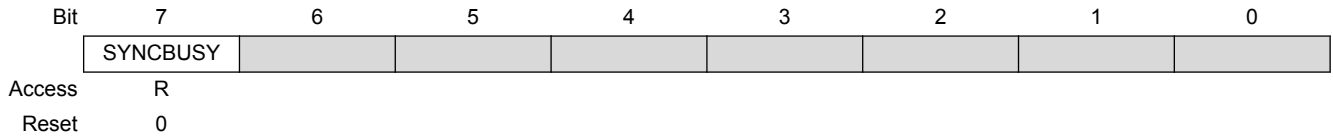
Writing a one to CTRL.SWRST will always take precedence, meaning that all other writes in the same write operation will be discarded.

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.

Value	Description
0	There is no ongoing reset operation.
1	The reset operation is ongoing.

### 23.8.2 Status

**Name:** STATUS  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** -



**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.

### 23.8.3 Non-Maskable Interrupt Control

**Name:** NMICTRL  
**Offset:** 0x02  
**Reset:** 0x00  
**Property:** Write-Protected

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

#### Bit 3 – NMIFILTEN Non-Maskable Interrupt Filter Enable

Value	Description
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.

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

**23.8.4 Non-Maskable Interrupt Flag Status and Clear**

**Name:** NMIFLAG  
**Offset:** 0x03  
**Reset:** 0x00  
**Property:** -

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

**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.

### 23.8.5 Event Control

**Name:** EVCTRL  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
							EXTINTEOx	EXTINTEOx
Access							R/W	R/W
Reset							0	0
Bit	15	14	13	12	11	10	9	8
	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx
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
	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx	EXTINTEOx
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 17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0 – EXTINTEOx** External Interrupt x Event Output Enable [x=17..0]

These bits indicate whether the event associated with the EXTINTx pin is enabled or not to generated for every detection.

Value	Description
0	Event from pin EXTINTx is disabled.
1	Event from pin EXTINTx is enabled.

### 23.8.6 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access							R/W	R/W
Reset							0	0
Bit	15	14	13	12	11	10	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
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 17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0 – EXTINTx** External Interrupt x Enable [x=17..0]

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.

Value	Description
0	The external interrupt x is disabled.
1	The external interrupt x is enabled.

### 23.8.7 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x0C  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
							EXTINTx	EXTINTx
Access							R/W	R/W
Reset							0	0
Bit	15	14	13	12	11	10	9	8
	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx
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
	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx	EXTINTx
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 17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0 – EXTINTx** External Interrupt x Enable [x=17..0]

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.

Value	Description
0	The external interrupt x is disabled.
1	The external interrupt x is enabled.



### 23.8.8 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access							R/W	R/W
Reset							0	0
Bit	15	14	13	12	11	10	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
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 17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0 – EXTINTx** External Interrupt x [x=17..0]

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.

### 23.8.9 Wake-Up Enable

**Name:** WAKEUP  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
	[Greyed out bits 31-24]							
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	[Greyed out bits 23-18]						WAKEUPENx	WAKEUPENx
Access							R/W	R/W
Reset							0	0
Bit	15	14	13	12	11	10	9	8
	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx
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
	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx	WAKEUPENx
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 17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0 – WAKEUPENx** External Interrupt x Wake-up Enable [x=17..0]

This bit enables or disables wake-up from sleep modes when the EXTINTx pin matches the external interrupt sense configuration.

Value	Description
0	Wake-up from the EXTINTx pin is disabled.
1	Wake-up from the EXTINTx pin is enabled.

### 23.8.10 Configuration n

**Name:** CONFIG  
**Offset:** 0x18 + n\*0x04 [n=0..2]  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24	
	FILTENx		SENSEx[2:0]			FILTENx		SENSEx[2:0]	
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	
	FILTENx		SENSEx[2:0]			FILTENx		SENSEx[2:0]	
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	
	FILTENx		SENSEx[2:0]			FILTENx		SENSEx[2:0]	
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	
	FILTENx		SENSEx[2:0]			FILTENx		SENSEx[2:0]	
Access	R	R	R	R	R	R	R	R	
Reset	0	0	0	0	0	0	0	0	

#### Bits 31, 27, 23, 19, 15, 11, 7,3 – FILTENx Filter 0 Enable [x=7..0]

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, 2:0 – SENSEx Input Sense 0 Configuration [x=7..0]

SENSE0[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

## 24. NVMCTRL – Nonvolatile Memory Controller

### 24.1 Overview

Nonvolatile Memory (NVM) is a reprogrammable Flash memory that retains program and data storage even with power off. It embeds a main array and a separate smaller array intended for EEPROM emulation (RWWEE) that can be programmed while reading the main array. 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.

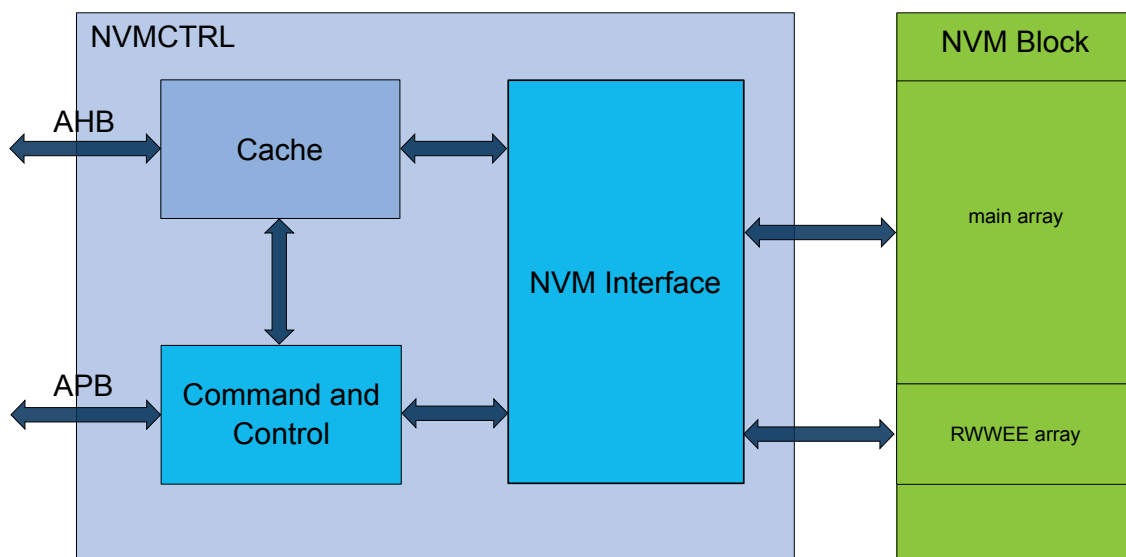
### 24.2 Features

- 32-bit AHB interface for reads and writes
- Read-While-Write DATA Flash
- 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 bootloader
- 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

**Note:** A register with property "Enable-Protected" may contain bits that are *not* enable-protected.

### 24.3 Block Diagram

Figure 24-1. Block Diagram



## 24.4 Signal Description

Not applicable.

## 24.5 Product Dependencies

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

### 24.5.1 Power Management

The NVMCTRL will continue to operate in any sleep mode where the selected source clock is running. The NVMCTRL interrupts can be used to wake up the device from 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) SLEEPFRM bit setting. Refer to the [CTRLB](#). SLEEPFRM register description for more details. The NVM block goes into low-power mode automatically when the device enters STANDBY mode regardless of SLEEPFRM. The NVM Page Buffer is lost when the NVM goes into low power mode therefore a write command must be issued prior entering the NVM low power mode. NVMCTRL SLEEPFRM can be disabled to avoid such loss when the CPU goes into sleep except if the device goes into STANDBY mode for which there is no way to retain the Page Buffer.

#### Related Links

[PM – Power Manager](#)

### 24.5.2 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 for the exact number of wait states to be used for a particular frequency range.

#### Related Links

[Electrical Characteristics](#)

### 24.5.3 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.

### 24.5.4 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 the section on the NVMCTRL [Security Bit](#) for details.

### 24.5.5 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)

- Status register (STATUS)

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

**Related Links**

[PAC - Peripheral Access Controller](#)

**24.5.6 Analog Connections**

Not applicable.

**24.6 Functional Description**

**24.6.1 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.

**24.6.1.1 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.

**24.6.2 Memory Organization**

Refer to the Physical Memory Map for memory sizes and addresses for each device.

The NVM is organized into rows, where each row contains four pages, as shown in the NVM Row Organization figure. 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 24-2. NVM Row Organization**

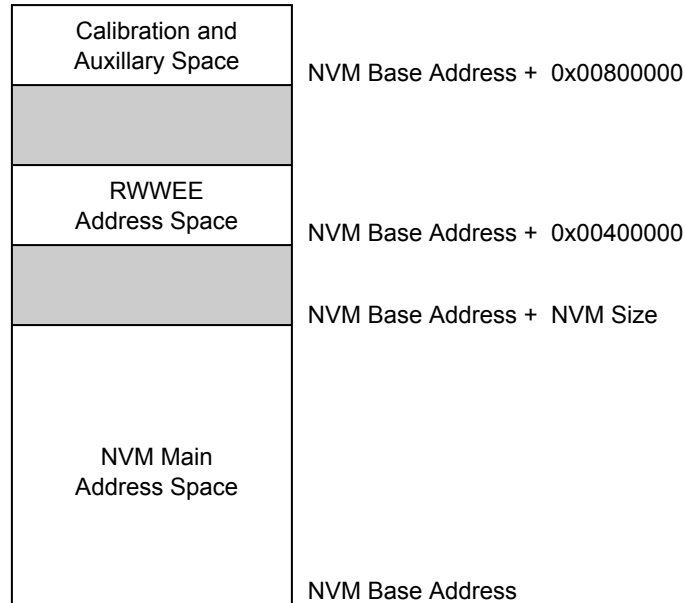
Row n	Page (n*4) + 3	Page (n*4) + 2	Page (n*4) + 1	Page (n*4) + 0
-------	----------------	----------------	----------------	----------------

The NVM block contains a calibration and auxiliary space plus a dedicated EEPROM emulation space that are memory mapped. Refer to the NVM Organization figure below for details.

The calibration and auxiliary space contains factory calibration and system configuration information. These spaces 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 section can be allocated at the end of the NVM main address space.

**Figure 24-3. NVM Memory Organization**

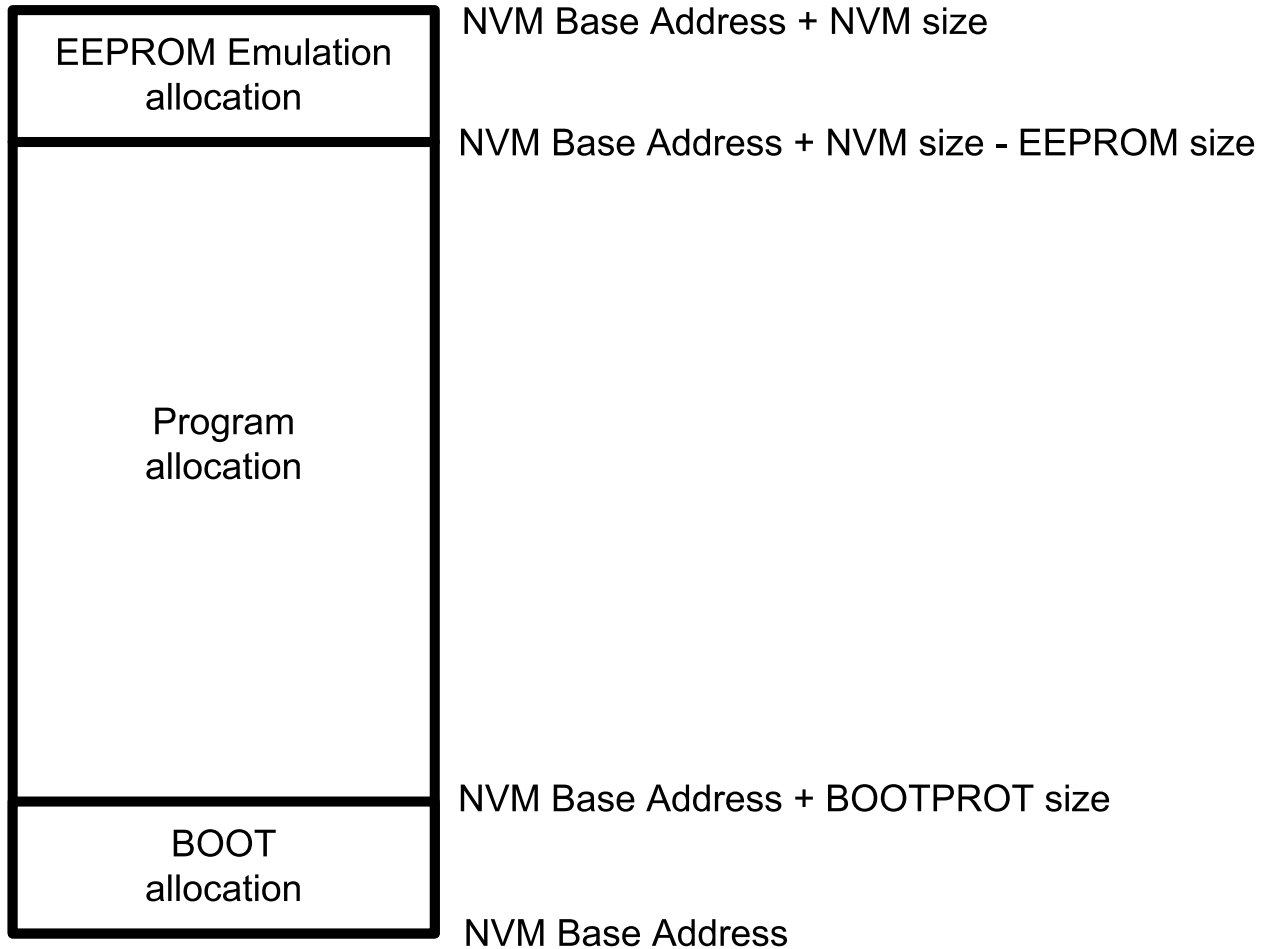


The lower rows in the NVM main address space can be allocated as a boot loader section by using the BOOTPROT fuses, and the upper rows can be allocated to EEPROM, as shown in the figure below.

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 is given in [Boot Loader Size](#), the number of rows allocated to the EEPROM are given in [EEPROM Size](#).

**Figure 24-4. EEPROM and Boot Loader Allocation**



**24.6.3 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 24-1. 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 until the setting is 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. Refer to the Physical Memory Map for calibration and auxiliary space address mapping.

#### 24.6.4 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 or the RWWEE address space directly, while other operations such as manual page writes and row erases 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.

The CTRLB register must be used to control the power reduction mode, read wait states, and the write mode.

##### 24.6.4.1 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.

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 Read Wait State Examples below.

Reading the NVM main address space while a programming or erase operation is ongoing on the NVM main array results in an AHB bus stall until the end of the operation. Reading the NVM main array does not stall the bus when the RWWEE array is being programmed or erased.

##### 24.6.4.2 RWWEE Read

Reading from the RWW EEPROM address space is performed via the AHB bus by addressing the RWWEE address space directly.

Read timings are similar to regular NVM read timings when access size is Byte or half-Word. The AHB data phase is twice as long in case of full-Word-size access.

It is not possible to read the RWWEE area while the NVM main array is being written or erased, whereas the RWWEE area can be written or erased while the main array is being read.

The RWWEE address space is not cached, therefore it is recommended to limit access to this area for performance and power consumption considerations.

##### 24.6.4.3 NVM Write

The NVM Controller requires that an erase must be done before programming. The entire NVM main address space and the RWWEE address space can be erased by a debugger Chip Erase command. Alternatively, rows can be individually erased by the Erase Row command or the RWWEE Erase Row command to erase the NVM main address space or the RWWEE address space, respectively.

After programming the NVM main array, 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 will lock all pages inside the region.

Data to be written to the NVM block are first written to 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 are not allowed and will cause a system exception.

Both the NVM main array and the RWWEE array share the same page buffer. 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 NVM main array or the RWWEE array by setting CTRLA.CMD to 'Write Page' or 'RWWEE Write Page', respectively, 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.

Automatic page writes are enabled by writing the manual write bit to zero (CTRLB.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 (CTRLB.MANW=1)

The row to be written to 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: CTRL.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 (CTRLB.MANW=0)

The row to be written to 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.

#### 24.6.4.4 Page Buffer Clear

The page buffer is automatically set to all '1' 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.

#### 24.6.4.5 Erase Row

Before a page can be written, the row containing that page must be erased. The Erase Row command can be used to erase the desired row in the NVM main address space. The RWWEE Erase Row can be used to erase the desired row in the RWWEE array. Erasing the row sets all bits to '1'. 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 to ADDR. Any address within the row can be used.
- Issue an Erase Row command.

**Note:** The NVM Address bit field in the Address register (ADDR.ADDR) uses 16-bit addressing.

### 24.6.4.6 Lock and Unlock Region

These commands are used to lock and unlock regions as detailed in section [Region Lock Bits](#).

### 24.6.4.7 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.

### 24.6.5 NVM User Configuration

The NVM user configuration resides in the auxiliary space. Refer to the Physical Memory Map of the device for calibration and auxiliary space address mapping.

The bootloader resides in the main array starting at offset zero. The allocated boot loader section is write-protected.

**Table 24-2. Boot Loader Size**

BOOTPROT [2:0]	Rows Protected by BOOTPROT	Boot Loader Size in Bytes
0x7 <sup>(1)</sup>	None	0
0x6	2	512
0x5	4	1024
0x4	8	2048
0x3	16	4096
0x2	32	8192
0x1	64	16384
0x0	128	32768

**Note:** 1) Default value is 0x7.

The EEPROM[2:0] bits indicate the EEPROM size, see the table below. The EEPROM resides in the upper rows of the NVM main address space and is writable, regardless of the region lock status.

**Table 24-3. EEPROM Size**

EEPROM[2:0]	Rows Allocated to EEPROM	EEPROM Size in Bytes
7	None	0
6	1	256
5	2	512
4	4	1024
3	8	2048
2	16	4096
1	32	8192
0	64	16384

### 24.6.6 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.

In order to increase the security level it is recommended to enable the internal BOD33 when the security bit is set.

#### Related Links

[DSU - Device Service Unit](#)

### 24.6.7 Cache

The NVM Controller cache reduces the device power consumption and improves system performance when wait states are required. Only the NVM main array address space is cached. 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 '0' to the Cache Disable bit in the Control B register ([CTRLB.CACHEDIS](#)).

The cache can be configured to three different modes using the Read Mode bit group in the Control B register ([CTRLB.READMODE](#)).

The INVALL command can be issued using the Command bits in the Control A register to invalidate all cache lines ([CTRLA.CMD=INVALL](#)). Commands affecting NVM content automatically invalidate cache lines.

### 24.7 Register Summary

Offset	Name	Bit Pos.										
0x00	CTRLA	7:0		CMD[6:0]								
		15:8		CMDEX[7:0]								
0x02 ... 0x03	Reserved											
0x04	CTRLB	7:0	MANW			RWS[3:0]						
		15:8						SLEPPRM[1:0]				
		23:16					CACHEDIS	READMODE[1:0]				
		31:24										
0x08	PARAM	7:0		NVMP[7:0]								
		15:8		NVMP[15:8]								
		23:16		RWWEEP[3:0]					PSZ[2:0]			
		31:24		RWWEEP[11:4]								
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			
		15:8							SB			
0x1A ... 0x1B	Reserved											
0x1C	ADDR	7:0		ADDR[7:0]								
		15:8		ADDR[15:8]								
		23:16			ADDR[21:16]							
		31:24										
0x20	LOCK	7:0		LOCK[7:0]								
		15:8		LOCK[15:8]								

### 24.8 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.

# ATSAMHAXEXXA

## NVMCTRL – Nonvolatile Memory Controller

---

Some registers require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 24.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x0000  
**Property:** PAC Write-Protection

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/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

**Bits 15:8 – CMDEX[7:0] Command Execution**

When this bit group is written to the key value 0xA5, the command written to CMD will be executed. If a value different from the key value is tried, the write will not be performed and the Programming Error bit in the Status register (STATUS.PROGE) will be set. PROGE is also set if a previously written command is not completed yet.

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.

INTFLAG.READY must be '1' when the command is issued.

Bit 0 of the CMDEX bit group will read back as '1' until the command is issued.

**Note:** The NVM Address bit field in the Address register (ADDR.ADDR) uses 16-bit addressing.

**Bits 6:0 – CMD[6:0] Command**

These bits define the command to be executed when the CMDEX key is written.

CMD[6:0]	Group Configuration	Description
0x00-0x01	-	Reserved
0x02	ER	Erase Row - Erases the row addressed by the ADDR register in the NVM main array.
0x03	-	Reserved
0x04	WP	Write Page - Writes the contents of the page buffer to the page addressed by the ADDR register.
0x05	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.

# ATSAMHAXEXXA

## NVMCTRL – Nonvolatile Memory Controller

CMD[6:0]	Group Configuration	Description
0x06	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.
0x07-0x0E	-	Reserved
0x0F	WL	Write Lockbits- write the LOCK register
0x1A-0x19	-	Reserved
0x1A	RWWEER	RWWE Erase Row - Erases the row addressed by the ADDR register in the RWWE array.
0x1B	-	Reserved
0x1C	RWWEWP	RWWE Write Page - Writes the contents of the page buffer to the page addressed by the ADDR register in the RWWE array.
0x1D-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	LDR	Lock Data Region - Locks the data region containing the address location in the ADDR register. When the Security Extension is enabled, only secure access can lock secure regions.
0x48	UDR	Unlock Data Region - Unlocks the data region containing the address location in the ADDR register. When the Security Extension is enabled, only secure access can unlock secure regions.
0x47-0x7F	-	Reserved



### 24.8.2 Control B

**Name:** CTRLB  
**Offset:** 0x04  
**Reset:** 0x00000080  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access						CACHEDIS	READMODE[1:0]	
Reset						0	0	0
Bit	15	14	13	12	11	10	9	8
Access							SLEEPPRM[1:0]	
Reset							0	0
Bit	7	6	5	4	3	2	1	0
Access	MANW			RWS[3:0]				
Reset	1			0	0	0	0	

**Bit 18 – CACHEDIS** Cache Disable  
 This bit is used to disable the cache.

Value	Description
0	The cache is enabled
1	The cache is disabled

**Bits 17:16 – READMODE[1:0]** NVMCTRL Read Mode

Value	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 increased 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 9:8 – SLEPPRM[1:0] Power Reduction Mode during Sleep

Indicates the Power Reduction Mode during sleep.

Value	Name	Description
0x0	WAKEUPACCESS	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

Note that reset value of this bit is '1'.

Value	Description
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 CTRLA.CMD register.

### Bits 4:1 – RWS[3:0] NVM Read Wait States

These bits control the number of wait states for a read operation. '0' indicates zero wait states, '1' 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.

### 24.8.3 NVM Parameter

**Name:** PARAM  
**Offset:** 0x08  
**Reset:** 0x000XXXXX  
**Property:** PAC Write-Protection

	Bit	31	30	29	28	27	26	25	24
		RWWEEP[11:4]							
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
		RWWEEP[3:0]				PSZ[2:0]			
Access		R	R	R	R		R	R	R
Reset		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 31:20 – RWWEEP[11:0]** Read While Write EEPROM emulation area Pages  
 Indicates the number of pages in the RWW EEPROM emulation address space.

**Bits 18:16 – PSZ[2:0]** Page Size  
 Indicates the page size. Not all devices of the device families will provide all the page sizes indicated in the table.

Value	Name	Description
0x0	8	8 bytes
0x1	16	16 bytes
0x2	32	32 bytes
0x3	64	64 bytes
0x4	128	128 bytes
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.

24.8.4 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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	6	5	4	3	2	1	0
							ERROR	READY
Access							R/W	R/W
Reset							0	0

**Bit 1 – ERROR** Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' to this bit clears the READY interrupt enable.

This bit will read as the current value of the READY interrupt enable.

### 24.8.5 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x10  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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).

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

**Bit 1 – ERROR** Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' 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 '0' to this bit has no effect.

Writing a '1' to this bit sets the READY interrupt enable.

This bit will read as the current value of the READY interrupt enable.

### 24.8.6 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** –

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

#### Bit 1 – ERROR Error

This flag is set on the occurrence of an NVME, LOCKE or PROGE error.

This bit can be cleared by writing a '1' to its bit location.

Value	Description
0	No errors have been received since the last clear.
1	At least one error has occurred since the last clear.

#### Bit 0 – READY NVM Ready

Value	Description
0	The NVM controller is busy programming or erasing.
1	The NVM controller is ready to accept a new command.

### 24.8.7 Status

**Name:** STATUS  
**Offset:** 0x18  
**Reset:** 0x0X00  
**Property:** –

Bit	15	14	13	12	11	10	9	8
								SB
Access								R
Reset								x
Bit	7	6	5	4	3	2	1	0
				NVME	LOCKE	PROGE	LOAD	PRM
Access				R/W	R/W	R/W	R/W	R
Reset				0	0	0	0	0

#### Bit 8 – SB Security Bit Status

Value	Description
0	The Security bit is inactive.
1	The Security bit is active.

#### Bit 4 – NVME NVM Error

This bit can be cleared by writing a '1' to its bit location.

Value	Description
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.

#### Bit 3 – LOCKE Lock Error Status

This bit can be cleared by writing a '1' to its bit location.

Value	Description
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.

#### Bit 2 – PROGE Programming Error Status

This bit can be cleared by writing a '1' to its bit location.

Value	Description
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.

**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. It remains set until a page write or a page buffer clear (PBCLR) command is given.

This bit can be cleared by writing a '1' 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 SLEEPFRM 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 SLEEPFRM set accordingly.

Value	Description
0	NVM is not in power reduction mode.
1	NVM is in power reduction mode.



### 24.8.8 Address

**Name:** ADDR  
**Offset:** 0x1C  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24	
Access									
Reset									
Bit	23	22	21	20	19	18	17	16	
			ADDR[21:16]						
Access			R/W	R/W	R/W	R/W	R/W	R/W	
Reset			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 21:0 – ADDR[21:0] NVM Address

ADDR drives the hardware (16-bit) address to the NVM when a command is executed using CMDEX. This register is also automatically updated when writing to the page buffer.

### 24.8.9 Lock Section

**Name:** LOCK  
**Offset:** 0x20  
**Reset:** 0xFFFF  
**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	x

#### Bits 15:0 – LOCK[15:0] Region Lock Bits

To set or clear these bits, the CMD register must be used.

Default state after erase will be unlocked (0x0000).

Value	Description
0	The corresponding lock region is locked.
1	The corresponding lock region is not locked.

## **25. PORT - I/O Pin Controller**

### **25.1 Overview**

The IO Pin Controller (PORT) controls the I/O pins of the device. The I/O pins are organized in a series of groups, collectively referred to as a PORT group. Each PORT group can have up to 32 pins that can be configured and controlled individually or as a group. The number of PORT groups on a device may depend on the package/number of pins. Each pin may either be used for general-purpose I/O under direct application control or be 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) explicitly without unintentionally changing the state of any other pins in the same port group by a single, atomic 8-, 16- or 32-bit write.

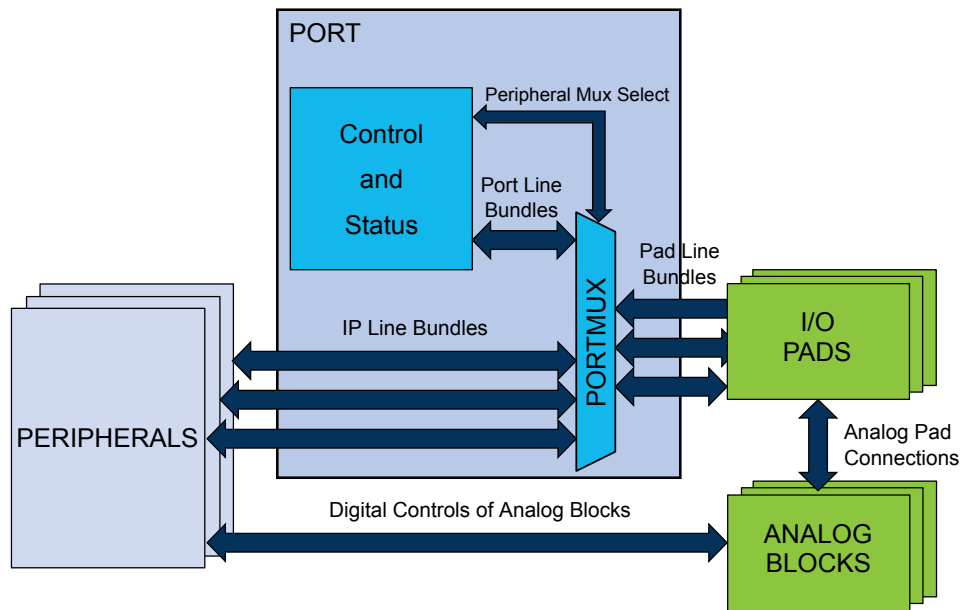
The PORT is connected to the high-speed bus matrix through an AHB/APB bridge. The Pin Direction, Data Output Value and Data Input Value registers may also be accessed using the low-latency CPU local bus (IOBUS; ARM® single-cycle I/O port) .

### **25.2 Features**

- Selectable input and output configuration for each individual 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

## 25.3 Block Diagram

Figure 25-1. PORT Block Diagram



## 25.4 Signal Description

Table 25-1. Signal description for PORT

Signal name	Type	Description
Pxy	Digital I/O	General-purpose I/O pin y in group x

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

### Related Links

[I/O Multiplexing and Considerations](#)

## 25.5 Product Dependencies

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

### 25.5.1 I/O Lines

The I/O lines of the PORT are mapped to pins of the physical device. The following naming scheme is used:

Each line bundle with up to 32 lines is assigned an identifier 'xy', with letter x=A, B, C... and two-digit number y=00, 01, ...31. Examples: A24, C03.

PORT pins are labeled 'Pxy' accordingly, for example PA24, PC03. This identifies each pin in the device uniquely.

Each pin may be controlled by one or more peripheral multiplexer settings, which allow the pad to be routed internally to a dedicated peripheral function. When the setting is enabled, the selected peripheral

has 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* for details.

Device-specific configurations may cause some lines (and the corresponding Pxy pin) not to be implemented.

### Related Links

[I/O Multiplexing and Considerations](#)

### 25.5.2 Power Management

During Reset, all PORT lines are configured as inputs with input buffers, output buffers and pull disabled.

The PORT peripheral will continue operating in any sleep mode where its source clock is running.

### 25.5.3 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 *PM – Power Manager*.

The PORT requires an APB clock, which may be divided from the CPU main clock and allows the CPU to access the registers of PORT through the high-speed matrix and the AHB/APB bridge.

The PORT also requires an AHB clock for CPU IOBUS accesses to the PORT. That AHB clock is the internal PORT clock.

The priority of IOBUS accesses is higher than APB accesses. One clock cycle latency can be observed on the APB access in case of concurrent PORT accesses.

### Related Links

[Peripheral Clock Masking](#)

### 25.5.4 DMA

Not applicable.

### 25.5.5 Interrupts

Not applicable.

### 25.5.6 Events

The events of this peripheral are connected to the Event System.

### Related Links

[EVSYS – Event System](#)

### 25.5.7 Debug Operation

When the CPU is halted in debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

### 25.5.8 Register Access Protection

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

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

Write-protection does not apply for accesses through an external debugger.

**Related Links**

[PAC - Peripheral Access Controller](#)

**25.5.9 TrustZone Specific Register Access Protection (only on devices with security attribution)**

**Note:** Refer to the *Mix-Secure Peripherals* section in the *Security Attribution Features* chapter for more information.

**25.5.10 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.

**25.5.11 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, which does not support wait states. It supports 8-bit, 16-bit and 32-bit sizes.

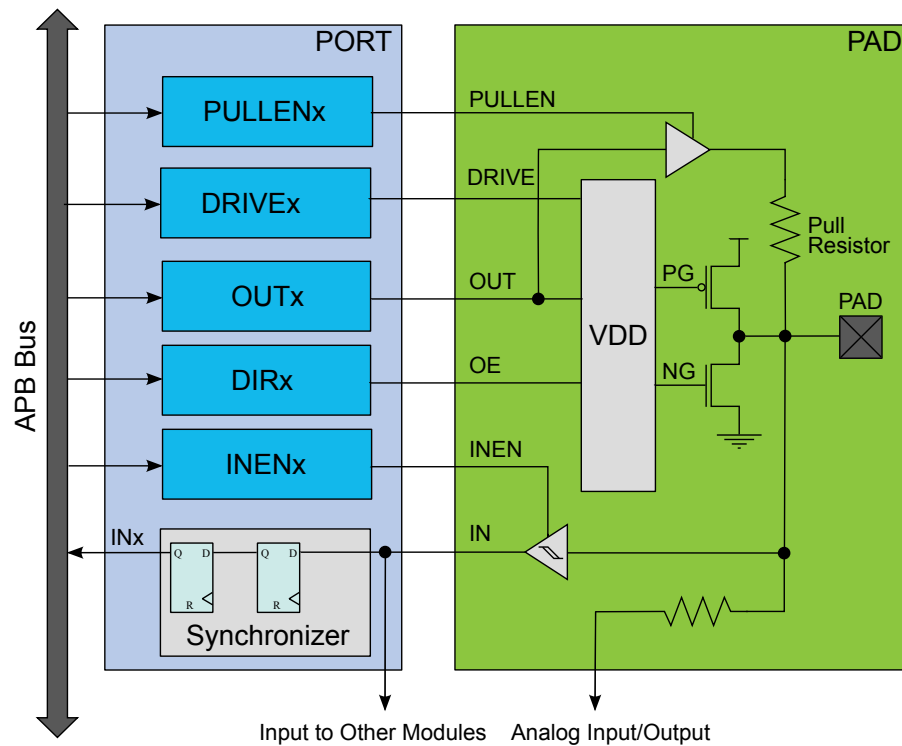
This bus is generally used for low latency operation. The Data Direction (DIR) and Data Output Value (OUT) registers can be read, written, set, cleared or be toggled using this bus, and the Data Input Value (IN) registers can be read.

Since the IOBUS cannot wait for IN register resynchronization, the Control register (CTRL) must be configured to continuous sampling of all pins that need to be read via the IOBUS in order to prevent stale data from being read.

**Note:** Refer to the *Product Mapping* chapter for the IOBUS address.

## 25.6 Functional Description

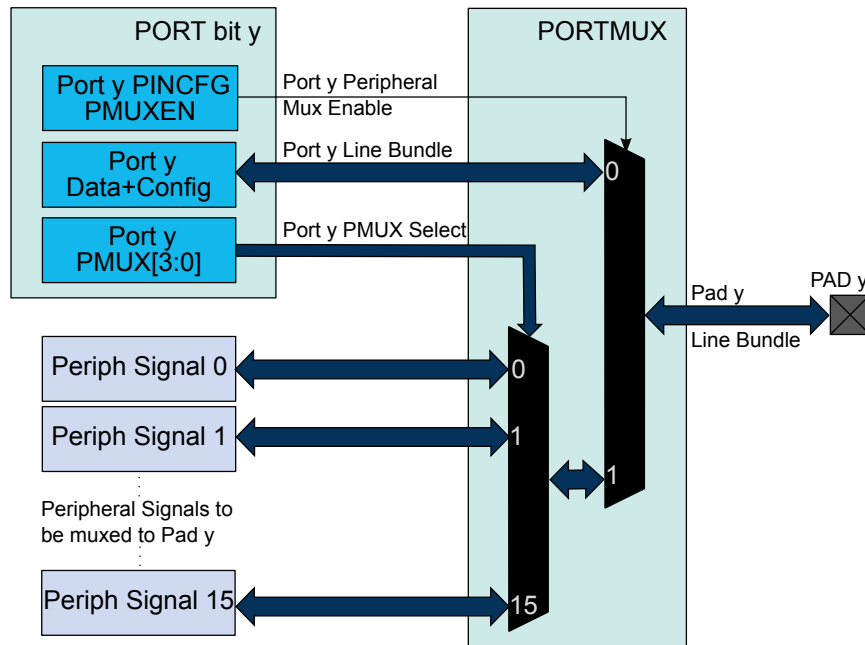
Figure 25-2. Overview of the PORT



### 25.6.1 Principle of Operation

Each PORT group of up to 32 pins is controlled by the registers in PORT, as described in the figure. These registers in PORT are duplicated for each PORT group, with increasing base addresses. The number of PORT groups may depend on the package/number of pins.

**Figure 25-3. Overview of the peripheral functions multiplexing**



The I/O pins of the device are controlled by PORT peripheral registers. Each port pin has a corresponding bit in the Data Direction (DIR) and Data Output Value (OUT) registers to enable that pin as an output and to define the output state.

The direction of each pin in a PORT group is configured by the DIR register. If a bit in DIR is set to '1', the corresponding pin is configured as an output pin. If a bit in DIR is set to '0', the corresponding pin is configured as an input pin.

When the direction is set as output, the corresponding bit in the OUT register will set the level of the pin. If bit y in OUT is written to '1', pin y is driven HIGH. If bit y in OUT is written to '0', pin y is driven LOW. Pin configuration can be set by Pin Configuration (PINCFGy) registers, with y=00, 01, ..31 representing the bit position.

The Data Input Value (IN) is set as the input value of a port pin with resynchronization to the PORT clock. To reduce power consumption, these input synchronizers can be clocked only when system requires reading the input value, as specified in the SAMPLING field of the Control register (CTRL). The value of the pin can always be read, whether the pin is configured as input or output. If the Input Enable bit in the Pin Configuration registers (PINCFGy.INEN) is '0', the input value will not be sampled.

In PORT, the Peripheral Multiplexer Enable bit in the PINCFGy register (PINCFGy.PMUXEN) can be written to '1' to enable the connection between peripheral functions and individual I/O pins. The Peripheral Multiplexing n (PMUXn) registers select the peripheral function for the corresponding pin. This will override the connection between the PORT and that I/O pin, and connect the selected peripheral signal to the particular I/O pin instead of the PORT line bundle.

## 25.6.2 Basic Operation

### 25.6.2.1 Initialization

After reset, all standard function device I/O pads are connected to the PORT with outputs tri-stated and input buffers disabled, even if there is no clock running.

However, specific pins, such as those used for connection to a debugger, may be configured differently, as required by their special function.



### 25.6.2.2 Operation

Each I/O pin Pxy can be controlled by the registers in PORT. Each PORT group x has its own set of PORT registers, with a base address at byte address (PORT + 0x80 \* group index) (A corresponds to group index 0, B to 1, etc...). Within that set of registers, the pin index is y, from 0 to 31.

Refer to *I/O Multiplexing and Considerations* for details on available pin configuration and PORT groups.

#### Configuring Pins as Output

To use pin Pxy as an *output*, write bit y of the DIR register to '1'. This can also be done by writing bit y in the DIRSET register to '1' - this will avoid disturbing the configuration of other pins in that group. The y bit in the OUT register must be written to the desired output value.

Similarly, writing an OUTSET bit to '1' will set the corresponding bit in the OUT register to '1'. Writing a bit in OUTCLR to '1' will set that bit in OUT to zero. Writing a bit in OUTTGL to '1' will toggle that bit in OUT.

#### Configuring Pins as Input

To use pin Pxy as an *input*, bit y in the DIR register must be written to '0'. This can also be done by writing bit y in the DIRCLR register to '1' - this will avoid disturbing the configuration of other pins in that group. The input value can be read from bit y in register IN as soon as the INEN bit in the Pin Configuration register (PINCFGy.INEN) is written to '1'.

By default, the input synchronizer is clocked only when an input read is requested. This will delay the read operation by two cycles of the PORT clock. To remove the delay, the input synchronizers for each PORT group of eight pins can be configured to be always active, but this will increase power consumption. This is enabled by writing '1' to the corresponding SAMPLINGn bit field of the CTRL register, see CTRL.SAMPLING for details.

#### Using Alternative Peripheral Functions

To use pin Pxy as one of the available peripheral functions, the corresponding PMUXEN bit of the PINCFGy register must be '1'. The PINCFGy register for pin Pxy is at byte offset (PINCFG0 + y).

The peripheral function can be selected by setting the PMUXO or PMUXE in the PMUXn register. The PMUXO/PMUXE is at byte offset PMUX0 + (y/2). The chosen peripheral must also be configured and enabled.

#### Related Links

[I/O Multiplexing and Considerations](#)

### 25.6.3 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 or pull configuration.

As 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 25-2](#).

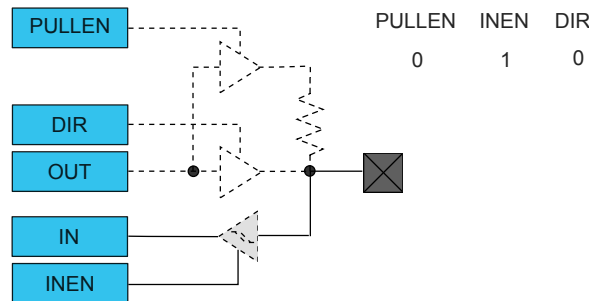
### 25.6.3.1 Pin Configurations Summary

**Table 25-2. Pin Configurations Summary**

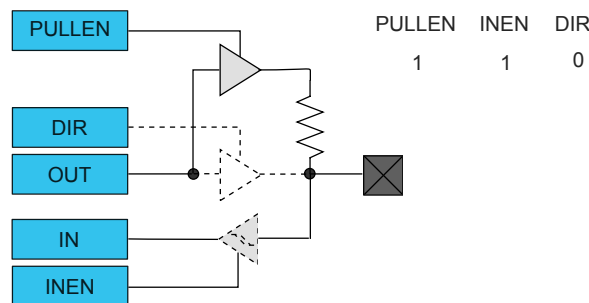
DIR	INEN	PULLEN	OUT	Configuration
0	0	0	X	Reset or analog I/O: all digital disabled
0	0	1	0	Pull-down; input disabled
0	0	1	1	Pull-up; input disabled
0	1	0	X	Input
0	1	1	0	Input with pull-down
0	1	1	1	Input with pull-up
1	0	X	X	Output; input disabled
1	1	X	X	Output; input enabled

### 25.6.3.2 Input Configuration

**Figure 25-4. I/O configuration - Standard Input**



**Figure 25-5. I/O Configuration - Input with Pull**



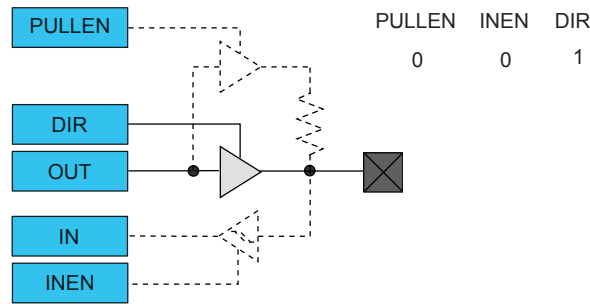
**Note:** When pull is enabled, the pull value is defined by the OUT value.

### 25.6.3.3 Totem-Pole Output

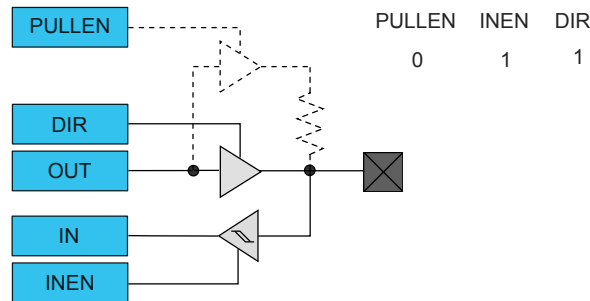
When configured for totem-pole (push-pull) output, the pin is driven low or high according to the corresponding bit setting in the OUT register. In this configuration there is no current limitation for sink or source other than what the pin is capable of. If the pin is configured for input, the pin will float if no external pull is connected.

**Note:** Enabling the output driver will automatically disable pull.

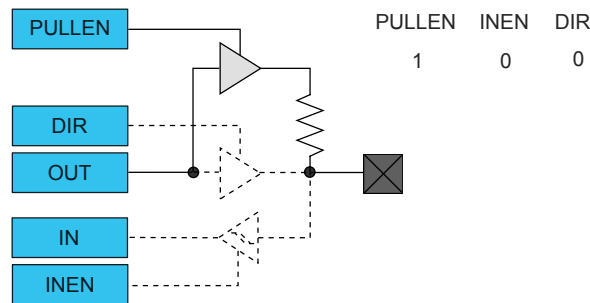
**Figure 25-6. I/O Configuration - Totem-Pole Output with Disabled Input**



**Figure 25-7. I/O Configuration - Totem-Pole Output with Enabled Input**



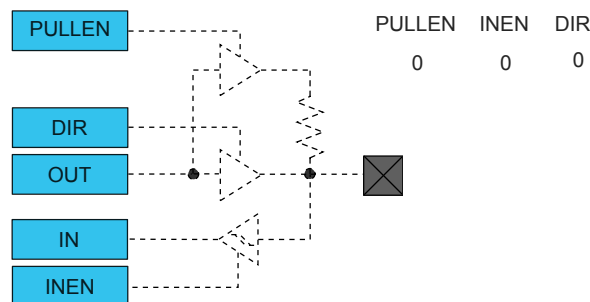
**Figure 25-8. I/O Configuration - Output with Pull**



#### 25.6.3.4 Digital Functionality Disabled

Neither Input nor Output functionality are enabled.

**Figure 25-9. I/O Configuration - Reset or Analog I/O: Digital Output, Input and Pull Disabled**



#### 25.6.4 PORT Access Priority

The PORT is accessed by different systems:

- The ARM® CPU through the ARM® single-cycle I/O port (IOBUS)
- The ARM® CPU through the high-speed matrix and the AHB/APB bridge (APB)

The following priority is adopted:

1. ARM® CPU IOBUS (No wait tolerated)
2. APB

## 25.7 Register Summary

Offset	Name	Bit Pos.							
0x00	DIR	7:0	DIR[7:0]						
		15:8	DIR[15:8]						
		23:16	DIR[23:16]						
		31:24	DIR[31:24]						
0x04	DIRCLR	7:0	DIRCLR[7:0]						
		15:8	DIRCLR[15:8]						
		23:16	DIRCLR[23:16]						
		31:24	DIRCLR[31:24]						
0x08	DIRSET	7:0	DIRSET[7:0]						
		15:8	DIRSET[15:8]						
		23:16	DIRSET[23:16]						
		31:24	DIRSET[31:24]						
0x0C	DIRTGL	7:0	DIRTGL[7:0]						
		15:8	DIRTGL[15:8]						
		23:16	DIRTGL[23:16]						
		31:24	DIRTGL[31:24]						
0x10	OUT	7:0	OUT[7:0]						
		15:8	OUT[15:8]						
		23:16	OUT[23:16]						
		31:24	OUT[31:24]						
0x14	OUTCLR	7:0	OUTCLR[7:0]						
		15:8	OUTCLR[15:8]						
		23:16	OUTCLR[23:16]						
		31:24	OUTCLR[31:24]						
0x18	OUTSET	7:0	OUTSET[7:0]						
		15:8	OUTSET[15:8]						
		23:16	OUTSET[23:16]						
		31:24	OUTSET[31:24]						
0x1C	OUTTGL	7:0	OUTTGL[7:0]						
		15:8	OUTTGL[15:8]						
		23:16	OUTTGL[23:16]						
		31:24	OUTTGL[31:24]						
0x20	IN	7:0	IN[7:0]						
		15:8	IN[15:8]						
		23:16	IN[23:16]						
		31:24	IN[31:24]						
0x24	CTRL	7:0	SAMPLING[7:0]						
		15:8	SAMPLING[15:8]						
		23:16	SAMPLING[23:16]						
		31:24	SAMPLING[31:24]						
0x28	WRCONFIG	7:0	PINMASK[7:0]						
		15:8	PINMASK[15:8]						
		23:16	DRVSTR				PULLEN	INEN	PMUXEN

# ATSAMHAXEXXA

## PORT - I/O Pin Controller

Offset	Name	Bit Pos.								
		31:24	HWSEL	WRPINCFG		WRPMUX				PMUX[3:0]
0x2C ... 0x2F	Reserved									
0x30	PMUX0	7:0								PMUXE[3:0]
0x31	PMUX1	7:0								PMUXE[3:0]
0x32	PMUX2	7:0								PMUXE[3:0]
0x33	PMUX3	7:0								PMUXE[3:0]
0x34	PMUX4	7:0								PMUXE[3:0]
0x35	PMUX5	7:0								PMUXE[3:0]
0x36	PMUX6	7:0								PMUXE[3:0]
0x37	PMUX7	7:0								PMUXE[3:0]
0x38	PMUX8	7:0								PMUXE[3:0]
0x39	PMUX9	7:0								PMUXE[3:0]
0x3A	PMUX10	7:0								PMUXE[3:0]
0x3B	PMUX11	7:0								PMUXE[3:0]
0x3C	PMUX12	7:0								PMUXE[3:0]
0x3D	PMUX13	7:0								PMUXE[3:0]
0x3E	PMUX14	7:0								PMUXE[3:0]
0x3F	PMUX15	7:0								PMUXE[3:0]
0x40	PINCFG0	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x41	PINCFG1	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x42	PINCFG2	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x43	PINCFG3	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x44	PINCFG4	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x45	PINCFG5	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x46	PINCFG6	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x47	PINCFG7	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x48	PINCFG8	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x49	PINCFG9	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x4A	PINCFG10	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x4B	PINCFG11	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x4C	PINCFG12	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x4D	PINCFG13	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x4E	PINCFG14	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x4F	PINCFG15	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x50	PINCFG16	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x51	PINCFG17	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x52	PINCFG18	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x53	PINCFG19	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x54	PINCFG20	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x55	PINCFG21	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x56	PINCFG22	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x57	PINCFG23	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x58	PINCFG24	7:0		DRVSTR					PULLEN	INEN PMUXEN
0x59	PINCFG25	7:0		DRVSTR					PULLEN	INEN PMUXEN

Offset	Name	Bit Pos.								
0x5A	<a href="#">PINCFG26</a>	7:0		DRVSTR				PULLEN	INEN	PMUXEN
0x5B	<a href="#">PINCFG27</a>	7:0		DRVSTR				PULLEN	INEN	PMUXEN
0x5C	<a href="#">PINCFG28</a>	7:0		DRVSTR				PULLEN	INEN	PMUXEN
0x5D	<a href="#">PINCFG29</a>	7:0		DRVSTR				PULLEN	INEN	PMUXEN
0x5E	<a href="#">PINCFG30</a>	7:0		DRVSTR				PULLEN	INEN	PMUXEN
0x5F	<a href="#">PINCFG31</a>	7:0		DRVSTR				PULLEN	INEN	PMUXEN

## 25.8 PORT Pin Groups and Register Repetition



**Tip:** The I/O pins are assembled in pin groups ("PORT groups") with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

## 25.9 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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#).

### 25.9.1 Data Direction

**Name:** DIR  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

This register allows the user to configure one or more I/O pins as an input or output. This register can be manipulated without doing a read-modify-write operation by using the Data Direction Toggle (DIRTGL), Data Direction Clear (DIRCLR) and Data Direction Set (DIRSET) registers.



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

Value	Description
0	The corresponding I/O pin in the PORT group is configured as an input.
1	The corresponding I/O pin in the PORT group is configured as an output.



### 25.9.2 Data Direction Clear

**Name:** DIRCLR  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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.



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

#### Bits 31:0 – DIRCLR[31:0] Port Data Direction Clear

Writing a '0' to a bit has no effect.

Writing a '1' to a bit will clear the corresponding bit in the DIR register, which configures the I/O pin as an input.

Value	Description
0	The corresponding I/O pin in the PORT group will keep its configuration.
1	The corresponding I/O pin in the PORT group is configured as input.

### 25.9.3 Data Direction Set

**Name:** DIRSET  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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.



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

#### Bits 31:0 – DIRSET[31:0] Port Data Direction Set

Writing '0' to a bit has no effect.

Writing '1' to a bit will set the corresponding bit in the DIR register, which configures the I/O pin as an output.

Value	Description
0	The corresponding I/O pin in the PORT group will keep its configuration.
1	The corresponding I/O pin in the PORT group is configured as an output.

### 25.9.4 Data Direction Toggle

**Name:** DIRTGL  
**Offset:** 0x0C  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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.



**Tip:** The I/O pins are assembled in pin groups ("PORT groups") with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

#### Bits 31:0 – DIRTGL[31:0] Port Data Direction Toggle

Writing '0' to a bit has no effect.

Writing '1' to a bit will toggle the corresponding bit in the DIR register, which reverses the direction of the I/O pin.

Value	Description
0	The corresponding I/O pin in the PORT group will keep its configuration.
1	The direction of the corresponding I/O pin is toggled.

### 25.9.5 Data Output Value

**Name:** OUT  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

This register sets the data output drive value for the individual I/O pins in the PORT.

This register can be manipulated without doing a read-modify-write operation by using the Data Output Value Clear (OUTCLR), Data Output Value Set (OUTSET), and Data Output Value Toggle (OUTTGL) registers.



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

#### Bits 31:0 – OUT[31:0] PORT Data Output Value

For pins configured as outputs via the Data Direction register (DIR), these bits set the logical output drive level.

For pins configured as inputs via the Data Direction register (DIR) and with pull enabled via the Pull Enable bit in the Pin Configuration register (PINCFG.PULLEN), these bits will set the input pull direction.

# ATSAMHAXEXXA

## PORT - I/O Pin Controller

---

---

Value	Description
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.

### 25.9.6 Data Output Value Clear

**Name:** OUTCLR  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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.



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

**Bits 31:0 – OUTCLR[31:0] PORT Data Output Value Clear**

Writing '0' to a bit has no effect.

Writing '1' to a bit will clear the corresponding bit in the OUT register. Pins configured as outputs via the Data Direction register (DIR) will be set to low output drive level. Pins configured as inputs via DIR and with pull enabled via the Pull Enable bit in the Pin Configuration register (PINCFG.PULLEN) will set the input pull direction to an internal pull-down.

# ATSAMHAXEXXA

## PORT - I/O Pin Controller

Value	Description
0	The corresponding I/O pin in the PORT group will keep its configuration.
1	The corresponding I/O pin output is driven low, or the input is connected to an internal pull-down.

### 25.9.7 Data Output Value Set

**Name:** OUTSET  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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.



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

#### Bits 31:0 – OUTSET[31:0] PORT Data Output Value Set

Writing '0' to a bit has no effect.

Writing '1' 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 Data Direction register (DIR) with pull enabled via the Pull Enable register (PULLEN), these bits will set the input pull direction to an internal pull-up.



# ATSAMHAXEXXA

## PORT - I/O Pin Controller

Value	Description
0	The corresponding I/O pin in the group will keep its configuration.
1	The corresponding I/O pin output is driven high, or the input is connected to an internal pull-up.

### 25.9.8 Data Output Value Toggle

**Name:** OUTTGL  
**Offset:** 0x1C  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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.



**Tip:** The I/O pins are assembled in pin groups ("PORT groups") with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

#### Bits 31:0 – OUTTGL[31:0] PORT Data Output Value Toggle

Writing '0' to a bit has no effect.

Writing '1' 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 Data Direction register (DIR) with pull enabled via the Pull Enable register (PULLEN), these bits will toggle the input pull direction.

Value	Description
0	The corresponding I/O pin in the PORT group will keep its configuration.
1	The corresponding OUT bit value is toggled.

### 25.9.9 Data Input Value

**Name:** IN  
**Offset:** 0x20  
**Reset:** 0x40000000  
**Property:** -



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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.

### 25.9.10 Control

**Name:** CTRL  
**Offset:** 0x24  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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

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.

Value	Description
0	On demand sampling of I/O pin is enabled.
1	Continuous sampling of I/O pin is enabled.

### 25.9.11 Write Configuration

**Name:** WRCONFIG  
**Offset:** 0x28  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

This write-only register is used to configure several pins simultaneously with the same configuration and/or peripheral multiplexing.

In order to avoid 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	30	29	28	27	26	25	24
	HWSEL	WRPINCFG		WRPMUX	PMUX[3:0]			
Access	W	W		W	W	W	W	W
Reset	0	0		0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
		DRVSTR				PULLEN	INEN	PMUXEN
Access		W				W	W	W
Reset		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

**Bit 31 – HWSEL** Half-Word Select

This bit selects the half-word field of a 32-PORT group to be reconfigured in the atomic write operation.

This bit will always read as zero.

Value	Description
0	The lower 16 pins of the PORT group will be configured.
1	The upper 16 pins of the PORT group will be configured.

### 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.

Writing '0' to this bit has no effect.

Writing '1' to this bit updates the configuration of the selected pins with the written WRCONFIG.DRVSTR, WRCONFIG.PULLEN, WRCONFIG.INEN, WRCONFIG.PMUXEN, and WRCONFIG.PINMASK values.

This bit will always read as zero.

Value	Description
0	The PINCFGy registers of the selected pins will not be updated.
1	The PINCFGy registers of the selected pins will be updated.

### 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.

Writing '0' to this bit has no effect.

Writing '1' 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.

Value	Description
0	The PMUXn registers of the selected pins will not be updated.
1	The PMUXn registers of the selected pins will be updated.

### 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 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, when the WRCONFIG.WRPINCFG bit is set.

This bit will always read as zero.

### 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.

These bits will always read as zero.

Value	Description
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 PORT group will be updated.

### 25.9.12 Peripheral Multiplexing n

**Name:** PMUX  
**Offset:** 0x30 + n\*0x01 [n=0..15]  
**Reset:** 0x00  
**Property:** PAC Write-Protection



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

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.

	Bit	7	6	5	4	3	2	1	0
		PMUXO[3:0]				PMUXE[3:0]			
Access		RW	RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0	0

#### Bits 7:4 – PMUXO[3:0] Peripheral Multiplexing for Odd-Numbered Pin

These bits select the peripheral function for odd-numbered pins ( $2*n + 1$ ) of a PORT group, if the corresponding PINCFGy.PMUXEN bit is '1'.

Not all possible values for this selection may be valid. For more details, refer to the *I/O Multiplexing and Considerations*.

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	I	Peripheral function I selected
0x9-0xF	-	Reserved



**Bits 3:0 – PMUXE[3:0]** Peripheral Multiplexing for Even-Numbered Pin

These bits select the peripheral function for even-numbered pins ( $2*n$ ) of a PORT group, if the corresponding PINCFGy.PMUXEN bit is '1'.

Not all possible values for this selection may be valid. For more details, refer to the *I/O Multiplexing and Considerations*.

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	I	Peripheral function I selected
0x9-0xF	-	Reserved

**Related Links**

[I/O Multiplexing and Considerations](#)

### 25.9.13 Pin Configuration

**Name:** PINCFG  
**Offset:** 0x40 + n\*0x01 [n=0..31]  
**Reset:** 0x00  
**Property:** PAC Write-Protection



**Tip:** The I/O pins are assembled in pin groups (“PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

There are up to 32 Pin Configuration registers in each PORT group, one for each I/O line.

	Bit	7	6	5	4	3	2	1	0
			DRVSTR				PULLEN	INEN	PMUXEN
Access			RW				RW	RW	RW
Reset			0				0	0	0

#### Bit 6 – DRVSTR Output Driver Strength Selection

This bit controls the output driver strength of an I/O pin configured as an output.

Value	Description
0	Pin drive strength is set to normal drive strength.
1	Pin drive strength is set to stronger drive strength.

#### 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.

Value	Description
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.

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.

Value	Description
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.

**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.

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 '1' 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 PINCFGn.INEN is set.

Value	Description
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 function controls the direction and output drive value.

## 26. EVSYS – Event System

### 26.1 Overview

The Event System (EVSYS) allows autonomous, low-latency and configurable communication between peripherals.

Several peripherals can be configured to generate 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 peripheral. Peripherals that respond to events are called event users. Peripherals that generate events are called event generators. A peripheral can have one or more event generators and can have one or more event users.

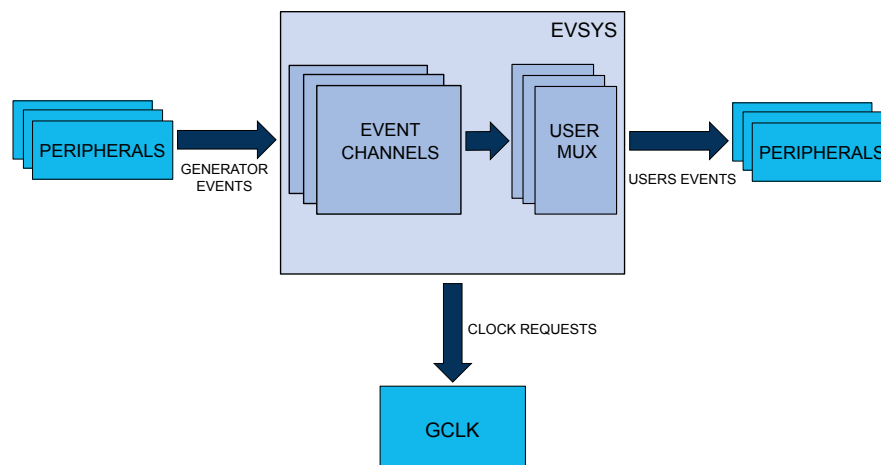
Communication is made without CPU intervention and without consuming system resources such as bus or RAM bandwidth. This reduces the load on the CPU and other system resources, compared to a traditional interrupt-based system.

### 26.2 Features

- 12 configurable event channels, where each channel can:
  - Be connected to any event generator.
  - Provide a pure asynchronous, resynchronized or synchronous path
- 74 event generators.
- 29 event users.
- Configurable edge detector.
- Peripherals can be event generators, event users, or both.
- SleepWalking and interrupt for operation in sleep modes.
- Software event generation.
- Each event user can choose which channel to respond to.

### 26.3 Block Diagram

Figure 26-1. Event System Block Diagram



## **26.4 Signal Description**

Not applicable.

## **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**

Not applicable.

### **26.5.2 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 the *PM – Power Manager* for details on the different sleep modes.

In all sleep modes, although the clock for the EVSYS is stopped, the device still can wake up the EVSYS clock. Some event generators can generate an event when their clocks are stopped.

#### **Related Links**

[PM – Power Manager](#)

### **26.5.3 Clocks**

The EVSYS bus clock (CLK\_EVSYS\_APB) can be enabled and disabled in the Main Clock module, and the default state of CLK\_EVSYS\_APB can be found in *Peripheral Clock Masking*.

Each EVSYS channel has a dedicated generic clock (GCLK\_EVSYS\_CHANNEL\_n). These are used for event detection and propagation for each channel. These clocks must be configured and enabled in the generic clock controller before using the EVSYS. Refer to *GCLK - Generic Clock Controller* for details.

#### **Related Links**

[Peripheral Clock Masking](#)

[GCLK - Generic Clock Controller](#)

### **26.5.4 DMA**

Not applicable.

### **26.5.5 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* for details.

#### **Related Links**

[Nested Vector Interrupt Controller](#)

### **26.5.6 Events**

Not applicable.

### **26.5.7 Debug Operation**

When the CPU is halted in Debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging.

### 26.5.8 Register Access Protection

Registers with write-access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag Status and Clear register (INTFLAG)

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

Write-protection does not apply for accesses through an external debugger.

### 26.5.9 Analog Connections

Not applicable.

## 26.6 Functional Description

### 26.6.1 Principle of Operation

The Event System consists of several channels which route the internal events from peripherals (generators) to other internal peripherals or IO pins (users). Each event generator can be selected as source for multiple channels, but a channel cannot be set to use multiple event generators at the same time.

### 26.6.2 Basic Operation

#### 26.6.2.1 Initialization

Before enabling events routing within the system, the User Multiplexer (USER) and Channel (CHANNEL) register must be configured. The User Multiplexer (USER) must be configured first.

Configure the User Multiplexer (USER) register:

1. The channel to be connected to a user is written to the Channel bit group (USER.CHANNEL)
2. The user to connect the channel is written to the User bit group (USER.USER)

Configure the Channel (CHANNEL) register:

1. The channel to be configured is written to the Channel Selection bit group (CHANNEL.CHANNEL)
2. The path to be used is written to the Path Selection bit group (CHANNEL.PATH)
3. The type of edge detection to use on the channel is written to the Edge Selection bit group (CHANNEL.EDGSEL)
4. The event generator to be used is written to the Event Generator bit group (CHANNEL.EVGEN)

#### 26.6.2.2 Enabling, Disabling and Resetting

The EVSYS is always enabled.

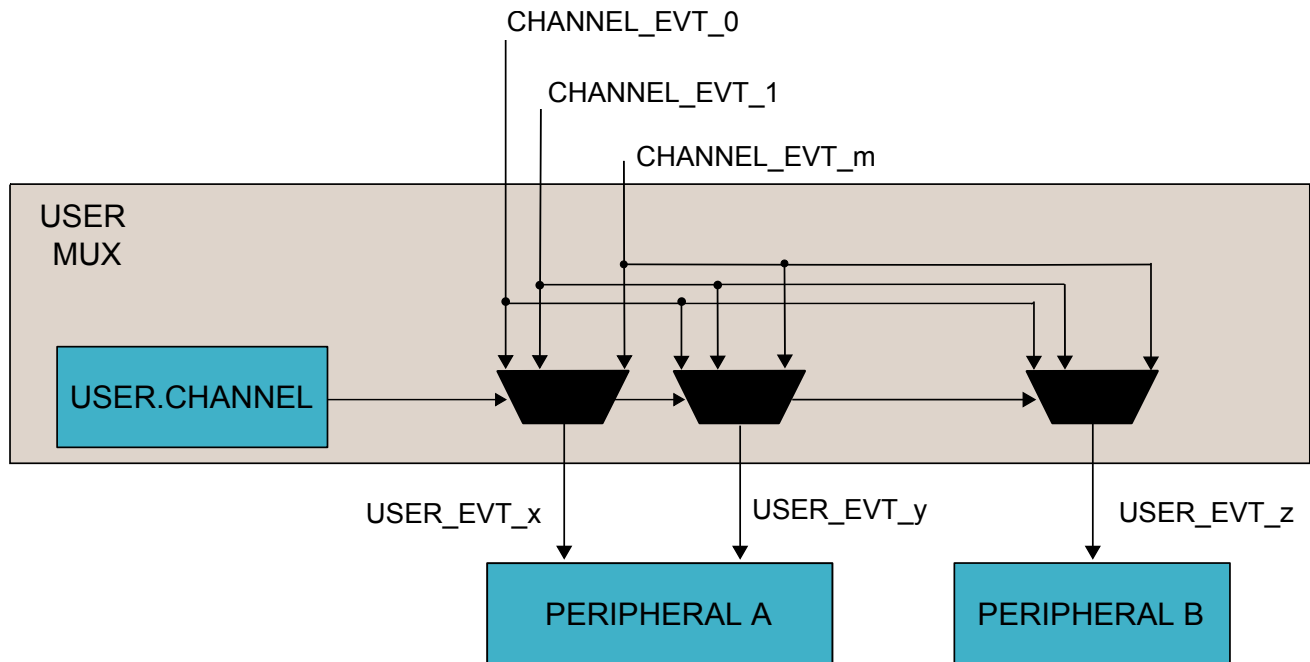
The EVSYS is reset by writing a '1' to the Software Reset bit in the Control register (CTRL.SWRST). All registers in the EVSYS will be reset to their initial state and all ongoing events will be canceled. Refer to CTRL.SWRST register for details.

#### 26.6.2.3 User Multiplexer Setup

The user multiplexer defines the channel to be connected to which event user. Each user multiplexer is dedicated to one event user. A user multiplexer receives all event channels output and must be configured to select one of these channels, as shown in the next figure. The channel is selected with the Channel bit group in the USER register (USER.CHANNEL). The user multiplexer must always be configured before the channel. A full list of selectable users can be found in the User Multiplexer register (USER) description. Refer to UserList 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 26-2. User MUX**



#### 26.6.2.4 Channel Setup

An event channel can select one event from a list of event generators. Depending on configuration, the selected event could be synchronized, resynchronized or asynchronously sent to the users. When synchronization or resynchronization is required, the channel includes an internal edge detector, allowing the Event System to generate internal events when rising, falling or both edges are detected on the selected event generator. An event channel is able to generate internal events for the specific software commands. All these configurations are available in the Channel register (CHANNEL).

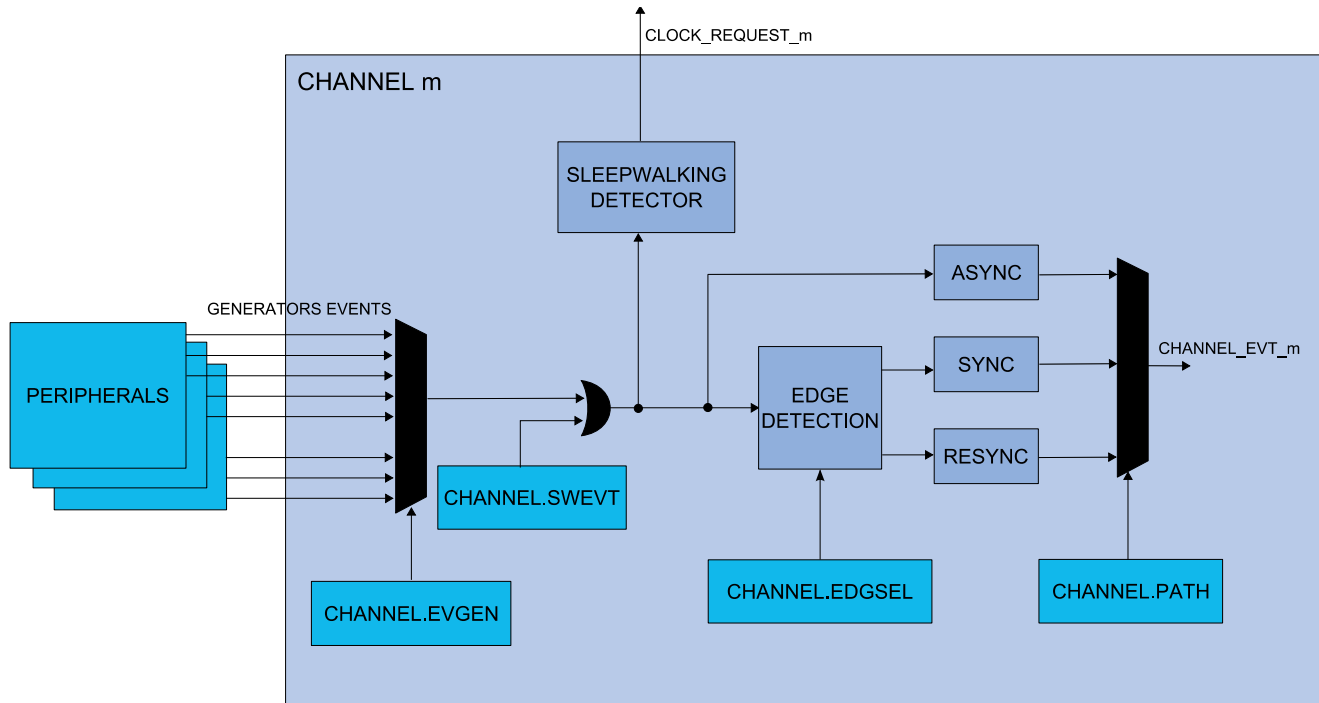
To configure a channel, the Channel register must be written in a single 32-bit write. It is possible to read out the configuration of a channel by first selecting the channel by writing to CHANNEL.CHANNEL using a, 8-bit write, and then performing a read of the CHANNEL register.

#### 26.6.2.5 Channel Path

There are three different ways to propagate the event provided by an event generator:

- Asynchronous path
- Synchronous path
- Resynchronized path

**Figure 26-3. 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 without intervention from the Event System. The GCLK for this channel (GCLK\_EVSYS\_CHANNEL\_n) is not mandatory, meaning that an event will be propagated to the user without any clock latency.

When the asynchronous path is selected, the channel cannot generate any interrupts, and the Channel Status register (CHSTATUS) is always zero. 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 generator for the generic clock and also if event user supports synchronous path. If event user doesn't support synchronous path, asynchronous path has to be selected. 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. For details on generic clock generators, refer to *GCLK - Generic Clock Controller*.

When using the synchronous path, the channel is able to generate 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.

**Related Links**

[GCLK - Generic Clock Controller](#)



### Resynchronized Path

The resynchronized path should be used when the event generator and the event channel do not share the same generic clock generator. When the resynchronized path is used, resynchronization of the event from the event generator is done in the channel. For details on generic clock generators, refer to *GCLK - Generic Clock Controller*.

When the resynchronized path is used, the channel is able to generate 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 is zero (CTRL.GCLKREQ=0), the channel operates in SleepWalking mode and requests the configured generic clock only when an event is to be propagated through the channel. If CTRL.GCLKREQ=1, the generic clock will always be on for the configured channel.

### Related Links

[GCLK - Generic Clock Controller](#)

#### 26.6.2.6 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.

#### 26.6.2.7 Event Generators

Each event channel can receive the events from all event generators. All event generators are listed in the statement of CHANNEL.EVGEN. For details on event generation, refer to the corresponding module chapter. The channel event generator is selected by the Event Generator bit group in the Channel register (CHANNEL.EVGEN). By default, the channels are not connected to any event generators (ie, CHANNEL.EVGEN = 0)

#### 26.6.2.8 Channel Status

The Channel Status register (CHSTATUS) shows the status of the channels when using a synchronous or resynchronized path. There are two different status bits in CHSTATUS for each of the available channels:

- The CHSTATUS.CHBUSYn bit will be set when an event on the corresponding channel n has not been handled by all event users connected to that channel.
- The CHSTATUS.USRRDYn bit will be set when all event users connected to the corresponding channel are ready to handle incoming events on that channel.

#### 26.6.2.9 Software Event

A software event can be initiated on a channel by setting the Software Event bit in the Channel register (CHANNEL.SWEVT) to '1' 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 '1' (CTRL.GCLKREQ=0x1).

### 26.6.3 Interrupts

The EVSYS has the following interrupt sources:

- Overrun Channel n (OVRn): for details, refer to *The Overrun Channel n Interrupt* section.
- Event Detected Channel n (EVDn): for details, refer to *The Event Detected Channel n Interrupt* section.

These interrupts events are asynchronous wake-up sources. See *Sleep Mode Controller*.

Each interrupt source has an interrupt flag which is in the Interrupt Flag Status and Clear (INTFLAG) register. The flag is set when the interrupt is issued. Each interrupt event can be individually enabled by setting a '1' to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by setting a '1' to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register. An interrupt event is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt event works until the interrupt flag is cleared, the interrupt is disabled, or the Event System is reset. See INTFLAG for details on how to clear interrupt flags.

All interrupt events from the peripheral are ORed together on system level to generate one combined interrupt request to the NVIC. Refer to the *Nested Vector Interrupt Controller* for details. The event user must read the INTFLAG register to determine what the interrupt condition is.

Note that interrupts must be globally enabled for interrupt requests to be generated. Refer to *Nested Vector Interrupt Controller* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

[Sleep Mode Controller](#)

#### 26.6.3.1 The Overrun Channel n Interrupt

The Overrun Channel n interrupt flag in the Interrupt Flag Status and Clear register (CHINTFLAGn.OVR) will be set, and the optional interrupt will be generated in the following cases:

- One or more event users on channel n is not ready when there is a new event.
- An event occurs when the previous event on channel m has not been handled by all event users connected to that channel.

The flag will only be set when using resynchronized paths. In the case of asynchronous path, the CHINTFLAGn.OVR is always read as zero.

#### Related Links

[Nested Vector Interrupt Controller](#)

#### 26.6.3.2 The Event Detected Channel n Interrupt

The Event Detected Channel n interrupt flag in the Interrupt Flag Status and Clear register (CHINTFLAGn.EVD) is set when an event coming from the event generator configured on channel n is detected.

The flag will only be set when using a resynchronized path. In the case of asynchronous path, the CHINTFLAGn.EVD is always zero.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 26.6.4 Sleep Mode Operation

The EVSYS can generate interrupts to wake up the device from any sleep mode.

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.

## 26.7 Register Summary

**Table 26-1. Event System Register Summary**

Offset	Name	Bit Pos.								
0x00	CTRL	7:0				GCLKREQ				SWRST
0x01	Reserved									
...										
0x03										
0x04	CHANNEL	7:0					CHANNEL[3:0]			
0x05		15:8								SWEVT
0x06		23:16					EVGEN[6:0]			
0x07		31:24					EDGSEL[1:0]		PATH[1:0]	
0x08	USER	7:0				USER[4:0]				
0x09		15:8				CHANNEL[4:0]				
0x0A	Reserved									
0x0B	Reserved									
0x0C	CHSTATUS	7:0	USRRDY7	USRRDY6	USRRDY5	USRRDY4	USRRDY3	USRRDY2	USRRDY1	USRRDY0
0x0D		15:8	CHBUSY7	CHBUSY6	CHBUSY5	CHBUSY4	CHBUSY3	CHBUSY2	CHBUSY1	CHBUSY0
0x0E		23:16					USRRDY11	USRRDY10	USRRDY9	USRRDY8
0x0F		31:24					CHBUSY11	CHBUSY10	CHBUSY9	CHBUSY8
0x10	INTENCLR	7:0	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
0x11		15:8	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
0x12		23:16					OVR11	OVR10	OVR9	OVR8
0x13		31:24					EVD11	EVD10	EVD9	EVD8
0x14	INTENSET	7:0	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
0x15		15:8	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
0x16		23:16					OVR11	OVR10	OVR9	OVR8
0x17		31:24					EVD11	EVD10	EVD9	EVD8
0x18	INTFLAG	7:0	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
0x19		15:8	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
0x1A		23:16					OVR11	OVR10	OVR9	OVR8
0x1B		31:24					EVD11	EVD10	EVD9	EVD8

## **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 enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Refer to [Register Access Protection](#).

**26.8.1 Control**

**Name:** CTRL  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** Write-Protected

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

**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.

Value	Description
0	Generic clock is requested and turned on only if an event is detected.
1	Generic clock for a channel is always on.

**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.

**Note:** Before applying a Software Reset it is recommended to disable the event generators.

### 26.8.2 Channel

**Name:** CHANNEL  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24	
					EDGSEL[1:0]		PATH[1:0]		
Access					R/W	R/W	R/W	R/W	
Reset					0	0	0	0	
Bit	23	22	21	20	19	18	17	16	
									EVGEN[6:0]
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset		0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	
								SWEVT	
Access								R/W	
Reset								0	
Bit	7	6	5	4	3	2	1	0	
					CHANNEL[3:0]				
Access					R/W	R/W	R/W	R/W	
Reset					0	0	0	0	

#### 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.

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.

PATH[1:0]	Name	Description
0x0	SYNCHRONOUS	Synchronous path
0x1	RESYNCHRONIZED	Resynchronized path
0x2	ASYNCHRONOUS	Asynchronous path
0x3		Reserved

### Bits 22:16 – EVGEN[6:0] Event Generator Selection

These bits are used to choose which event generator to connect to the selected channel.

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	
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	EIC EXTINT8	External Interrupt 8
0x15	EIC EXTINT9	External Interrupt 9
0x16	EIC EXTINT10	External Interrupt 10
0x17	EIC EXTINT11	External Interrupt 11

Value	Event Generator	Description
0x18	EIC EXTINT12	External Interrupt 12
0x19	EIC EXTINT13	External Interrupt 13
0x1A	EIC EXTINT14	External Interrupt 14
0x1B	EIC EXTINT15	External Interrupt 15
0x1C	Reserved	
0x1D	Reserved	
0x1E	DMAC CH0	Channel 0
0x1F	DMAC CH1	Channel 1
0x20	DMAC CH2	Channel 2
0x21	DMAC CH3	Channel 3
0x22	TCC0 OVF	Overflow
0x23	TCC0 TRG	Trig
0x24	TCC0 CNT	Counter
0x25	TCC0_MCX0	Match/Capture 0
0x26	TCC0_MCX1	Match/Capture 1
0x27	TCC0_MCX2	Match/Capture 2
0x28	TCC0_MCX3	Match/Capture 3
0x29	TCC1 OVF	Overflow
0x2A	TCC1 TRG	Trig
0x2B	TCC1 CNT	Counter
0x2C	TCC1_MCX0	Match/Capture 0
0x2D	TCC1_MCX1	Match/Capture 1
0x2E	TCC2 OVF	Overflow
0x2F	TCC2 TRG	Trig
0x30	TCC2 CNT	Counter
0x31	TCC2_MCX0	Match/Capture 0
0x32	TCC2_MCX1	Match/Capture 1
0x33	TC0 OVF	Overflow/Underflow
0x34	TC0 MC0	Match/Capture 0
0x35	TC0 MC1	Match/Capture 1
0x36	TC1 OVF	Overflow/Underflow
0x37	TC1 MC0	Match/Capture 0



Value	Event Generator	Description
0x38	TC1 MC1	Match/Capture 1
0x39	TC2 OVF	Overflow/Underflow
0x3A	TC2 MC0	Match/Capture 0
0x3B	TC2 MC1	Match/Capture 1
0x3C	TC3 OVF	Overflow/Underflow
0x3D	TC3 MC0	Match/Capture 0
0x3E	TC3 MC1	Match/Capture 1
0x3F	TC4 OVF	Overflow/Underflow
0x40	TC4 MC0	Match/Capture 0
0x41	TC4 MC1	Match/Capture 1
0x42	ADC RESRDY	Result Ready
0x43	ADC WINMON	Window Monitor
0x44	AC COMP0	Comparator 0
0x45	AC COMP1	Comparator 1
0x46	AC WIN0	Window 0
0x47	DAC EMPTY	
0x48	PTC EOC	End of Conversion
0x49	PTC WCOMP	Window Comparator
0x4A-0x7F	Reserved	

**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 has the same behavior similar to an event.

This bit must be written together with CHANNEL.CHANNEL using 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 3:0 – CHANNEL[3:0]** Channel Selection

These bits are used to select the channel to be set up or read from.

### 26.8.3 User Multiplexer

**Name:** USER  
**Offset:** 0x08  
**Reset:** 0x0000  
**Property:** Write-Protected

Bit	15	14	13	12	11	10	9	8
	CHANNEL[4:0]							
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	USER[4:0]							
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

**Bits 12:8 – CHANNEL[4:0]** 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.

CHANNEL[4:0]	Channel Number
0x0	No Channel Output Selected
0x1-0xC	Channel n-1 selected
0xD-0xFF	Reserved

**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 26-2. 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	TCC0 EV0		Asynchronous, synchronous and resynchronized paths
0x05	TCC0 EV1		Asynchronous, synchronous and resynchronized paths
0x06	TCC0 MC0	Match/Capture 0	Asynchronous, synchronous and resynchronized paths

USER[7:0]	User Multiplexer	Description	Path Type
0x07	TCC0 MC1	Match/Capture 1	Asynchronous, synchronous and resynchronized paths
0x08	TCC0 MC2	Match/Capture 2	Asynchronous, synchronous and resynchronized paths
0x09	TCC0 MC3	Match/Capture 3	Asynchronous, synchronous and resynchronized paths
0x0A	TCC1 EV0		Asynchronous, synchronous and resynchronized paths
0x0B	TCC1 EV1		Asynchronous, synchronous and resynchronized paths
0x0C	TCC1 MC0	Match/Capture 0	Asynchronous, synchronous and resynchronized paths
0x0D	TCC1 MC1	Match/Capture 1	Asynchronous, synchronous and resynchronized paths
0x0E	TCC2 EV0		Asynchronous, synchronous and resynchronized paths
0x0F	TCC2 EV1		Asynchronous, synchronous and resynchronized paths
0x10	TCC2 MC0	Match/Capture 0	Asynchronous, synchronous and resynchronized paths
0x11	TCC2 MC1	Match/Capture 1	Asynchronous, synchronous and resynchronized paths
0x12	TC0		Asynchronous, synchronous and resynchronized paths
0x13	TC1		Asynchronous, synchronous and resynchronized paths
0x14	TC2		Asynchronous, synchronous and resynchronized paths
0x15	TC3		Asynchronous, synchronous and resynchronized paths
0x16	TC4		Asynchronous, synchronous and resynchronized paths
0x17	ADC START	ADC start conversion	Asynchronous path only
0x18	ADC SYNC	Flush ADC	Asynchronous path only
0x19	AC COMP0	Start comparator 0	Asynchronous path only
0x1A	AC COMP1	Start comparator 1	Asynchronous path only
0x1B	DAC START	DAC start conversion	

# ATSAMHAXEXXA

## EVSYS – Event System

USER[7:0]	User Multiplexer	Description	Path Type
0x1C	PTC STCONV	PTC start conversion	Asynchronous path only
0x1D-0x1F	Reserved		Reserved

### 26.8.4 Channel Status

**Name:** CHSTATUS  
**Offset:** 0x0C  
**Reset:** 0x000F00FF  
**Property:** -

	Bit	31	30	29	28	27	26	25	24
						CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>
Access						R	R	R	R
Reset						0	0	0	0
	Bit	23	22	21	20	19	18	17	16
						USRRDY <sub>n</sub>	USRRDY <sub>n</sub>	USRRDY <sub>n</sub>	USRRDY <sub>n</sub>
Access						R	R	R	R
Reset						0	0	0	0
	Bit	15	14	13	12	11	10	9	8
		CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>	CHBUSY <sub>n</sub>
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
		USRRDY <sub>n</sub>	USRRDY <sub>n</sub>	USRRDY <sub>n</sub>	USRRDY <sub>n</sub>	USRRDY <sub>n</sub>	USRRDY <sub>n</sub>	USRRDY <sub>n</sub>	USRRDY <sub>n</sub>
Access		R	R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0	0

**Bits 27,26,25,24,15,14,13,12,11,10,9,8 – CHBUSY<sub>n</sub>** Channel n Busy [n=11..0]

This bit is cleared when channel n is idle

This bit is set if an event on channel n has not been handled by all event users connected to channel n.

**Bits 19,18,17,16,7,6,5,4,3,2,1,0 – USRRDY<sub>n</sub>** Channel n User Ready [n=11..0]

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 n are ready to handle incoming events on channel n.

### 26.8.5 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
					EVDn	EVDn	EVDn	EVDn
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					OVRn	OVRn	OVRn	OVRn
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn
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
	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn
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 27,26,25,24,15,14,13,12,11,10,9,8 – EVDn** Channel n Event Detection Interrupt Enable [n=11..0]  
 Writing a zero to this bit has no effect.

Writing a one to this bit will clear the Event Detected Channel n Interrupt Enable bit, which disables the Event Detected Channel n interrupt.

Value	Description
0	The Event Detected Channel n interrupt is disabled.
1	The Event Detected Channel n interrupt is enabled.

**Bits 19,18,17,16,7,6,5,4,3,2,1,0 – OVRn** Channel n Overrun Interrupt Enable [n=11..0]  
 Writing a zero to this bit has no effect.

Writing a one to this bit will clear the Overrun Channel n Interrupt Enable bit, which disables the Overrun Channel n interrupt.

Value	Description
0	The Overrun Channel n interrupt is disabled.
1	The Overrun Channel n interrupt is enabled.

### 26.8.6 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** Write-Protected

Bit	31	30	29	28	27	26	25	24
					EVDn	EVDn	EVDn	EVDn
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					OVRn	OVRn	OVRn	OVRn
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn
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
	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn
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 27,26,25,24,15,14,13,12,11,10,9,8 – EVDn** Channel n Event Detection Interrupt Enable [n=11..0]  
 Writing a zero to this bit has no effect.

Writing a one to this bit will set the Event Detected Channel n Interrupt Enable bit, which enables the Event Detected Channel n interrupt.

Value	Description
0	The Event Detected Channel n interrupt is disabled.
1	The Event Detected Channel n interrupt is enabled.

**Bits 19,18,17,16,7,6,5,4,3,2,1,0 – OVRn** Channel n Overrun Interrupt Enable [n=11..0]  
 Writing a zero to this bit has no effect.

Writing a one to this bit will set the Overrun Channel n Interrupt Enable bit, which enables the Overrun Channel n interrupt.

Value	Description
0	The Overrun Channel n interrupt is disabled.
1	The Overrun Channel n interrupt is enabled.

### 26.8.7 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
					EVDn	EVDn	EVDn	EVDn
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					OVRn	OVRn	OVRn	OVRn
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn	EVDn
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
	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn	OVRn
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 27,26,25,24,15,14,13,12,11,10,9,8 – EVDn** Channel n Event Detection [n=11..0]

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.EVDn is one.

When the event channel path is asynchronous, the EVDn 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 19,18,17,16,7,6,5,4,3,2,1,0 – OVRn** Channel n Overrun [n=11..0]

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.OVRn is one.

When the event channel path is asynchronous, the OVRn 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 n interrupt flag.



## 27. SERCOM – Serial Communication Interface

### 27.1 Overview

There are up to six instances of the serial communication interface (SERCOM) peripheral.

A SERCOM can be configured to support a number of modes: I<sup>2</sup>C, SPI, and USART. When an instance of SERCOM is configured and enabled, all of the resources of that SERCOM instance will be dedicated to the selected mode.

The SERCOM serial engine consists of a transmitter and receiver, baud-rate generator and address matching functionality. It can use the internal generic clock or an external clock. Using an external clock allows the SERCOM to be operated in all Sleep modes.

#### Related Links

[SERCOM USART](#)

[SERCOM SPI – SERCOM Serial Peripheral Interface](#)

[SERCOM I2C – SERCOM Inter-Integrated Circuit](#)

### 27.2 Features

- Interface for configuring into one of the following:
  - Inter-Integrated Circuit (I<sup>2</sup>C) Two-wire Serial Interface
  - System Management Bus (SMBus™) compatible
  - Serial Peripheral Interface (SPI)
  - Universal Synchronous/Asynchronous Receiver/Transmitter (USART)
- Single transmit buffer and double receive buffer
- Baud-rate generator
- Address match/mask logic
- Operational in all Sleep modes with an external clock source
- Can be used with DMA

See the Related Links for full feature lists of the interface configurations.

#### Related Links

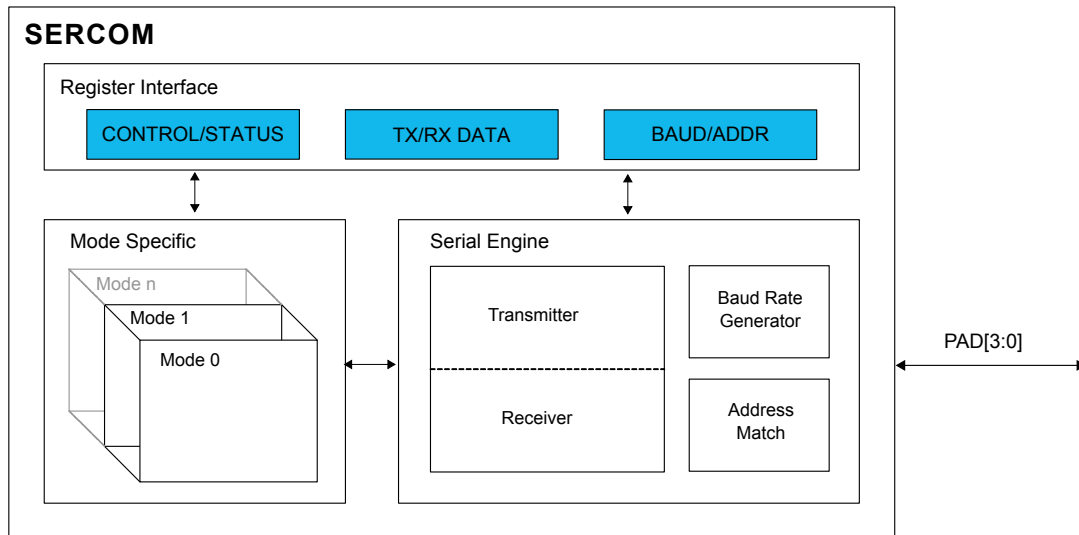
[SERCOM USART](#)

[SERCOM SPI – SERCOM Serial Peripheral Interface](#)

[SERCOM I2C – SERCOM Inter-Integrated Circuit](#)

## 27.3 Block Diagram

Figure 27-1. SERCOM Block Diagram



## 27.4 Signal Description

See the respective SERCOM mode chapters for details.

### Related Links

[SERCOM USART](#)

[SERCOM SPI – SERCOM Serial Peripheral Interface](#)

[SERCOM I2C – SERCOM Inter-Integrated Circuit](#)

## 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 I/O lines requires the I/O pins to be configured using port configuration (PORT).

The SERCOM has four internal pads, PAD[3:0], and the signals from I2C, 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.

### Related Links

[SERCOM USART](#)

[SERCOM SPI – SERCOM Serial Peripheral Interface](#)

[SERCOM I2C – SERCOM Inter-Integrated Circuit](#)

[PORT: IO Pin Controller](#)

[Block Diagram](#)

### 27.5.2 Power Management

The SERCOM can operate in any Sleep mode provided the selected clock source is running. SERCOM interrupts can be configured to wake the device from Sleep modes.

### Related Links

[PM – Power Manager](#)

### 27.5.3 Clocks

The SERCOM bus clock (CLK\_SERCOMx\_APB) can be enabled and disabled in the Power Manager. Refer to *Peripheral Clock Masking* for details and default status of this clock.

The SERCOM uses two generic clocks: GCLK\_SERCOMx\_CORE and GCLK\_SERCOMx\_SLOW. The core clock (GCLK\_SERCOMx\_CORE) is required to clock the SERCOM while working as a master. 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.

The generic clocks are asynchronous to the user interface clock (CLK\_SERCOMx\_APB). Due to this asynchronicity, writing to certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) for details.

### Related Links

[GCLK - Generic Clock Controller](#)

[Peripheral Clock Masking](#)

### 27.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). The DMAC must be configured before the SERCOM DMA requests are used.

### Related Links

[DMAC – Direct Memory Access Controller](#)

### 27.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller (NVIC). The NVIC must be configured before the SERCOM interrupts are used.

### Related Links

[Nested Vector Interrupt Controller](#)

### 27.5.6 Events

Not applicable.

### 27.5.7 Debug Operation

When the CPU is halted in debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

### 27.5.8 Register Access Protection

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

- Interrupt Flag Clear and Status register (INTFLAG)
- Status register (STATUS)

- Data register (DATA)
- Address register (ADDR)

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

PAC write-protection does not apply to accesses through an external debugger.

### Related Links

[PAC - Peripheral Access Controller](#)

### 27.5.9 Analog Connections

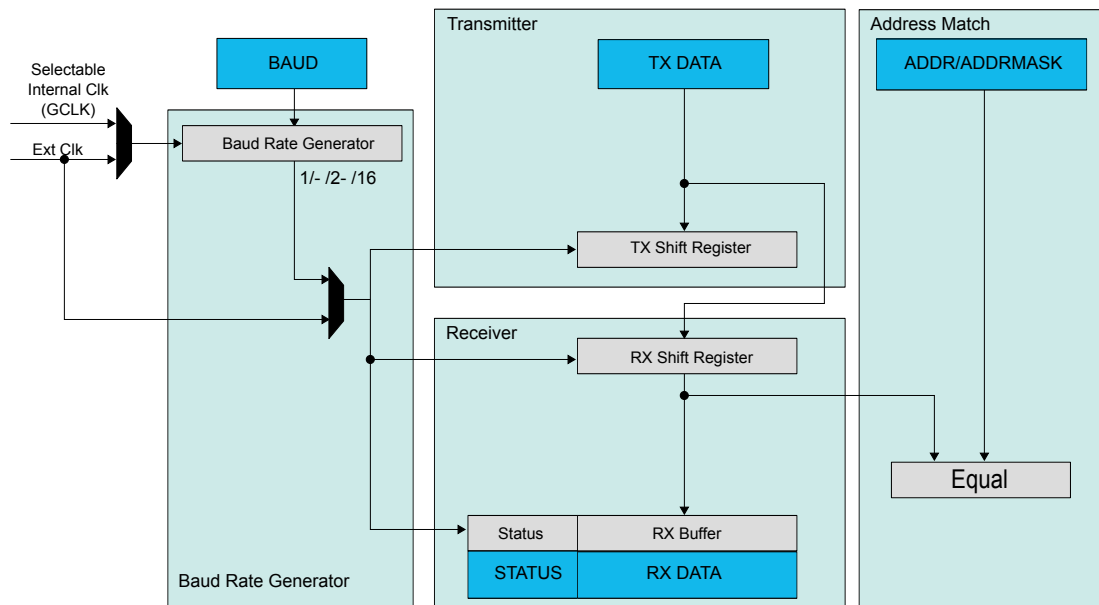
Not applicable.

## 27.6 Functional Description

### 27.6.1 Principle of Operation

The basic structure of the SERCOM serial engine is shown in [Figure 27-2](#). Labels in capital letters are synchronous to the system clock and accessible by the CPU; labels in lowercase letters can be configured to run on the GCLK\_SERCOMx\_CORE clock or an external clock.

**Figure 27-2. SERCOM Serial Engine**



The transmitter consists of a single write buffer and a shift register.

The receiver consists of a one-level (I<sup>2</sup>C), two-level (USART, SPI) 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 I<sup>2</sup>C operation.

**27.6.2 Basic Operation**

**27.6.2.1 Initialization**

The SERCOM must be configured to the desired mode by writing the Operating Mode bits in the Control A register (CTRLA.MODE). Refer to table SERCOM Modes for details.

**Table 27-1. 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 <sup>2</sup> C slave operation
0x5	I <sup>2</sup> C master operation
0x6-0x7	Reserved

For further initialization information, see the respective SERCOM mode chapters:

**Related Links**

[SERCOM USART](#)

[SERCOM SPI – SERCOM Serial Peripheral Interface](#)

[SERCOM I2C – SERCOM Inter-Integrated Circuit](#)

**27.6.2.2 Enabling, Disabling, and Resetting**

This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST) will reset all registers of this peripheral to their initial states, except the DBGCTRL register, and the peripheral is disabled.

Refer to the CTRLA register description for details.

**27.6.2.3 Clock Generation – Baud-Rate Generator**

The baud-rate generator, as shown in [Figure 27-3](#), generates internal clocks for asynchronous and synchronous communication. The 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 communication, the /16 (divide-by-16) output is used when transmitting, whereas the /1 (divide-by-1) output is used while receiving.

For synchronous communication, the /2 (divide-by-2) output is used.

This functionality is automatically configured, depending on the selected operating mode.

Figure 27-3. Baud Rate Generator

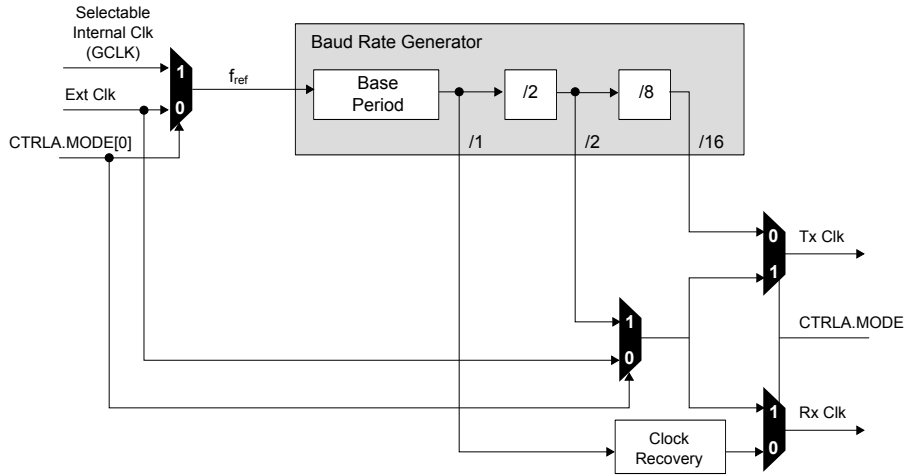


Table 27-2 contains equations for the baud rate (in bits per second) and the BAUD register value for each operating mode.

For asynchronous operation, there is one mode: *arithmetic mode*, the BAUD register value is 16 bits (0 to 65,535). *fractional mode*, the BAUD register value 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 operation, the BAUD register value is 8 bits (0 to 255).

Table 27-2. Baud Rate Equations

Operating Mode	Condition	Baud Rate (Bits Per Second)	BAUD Register Value Calculation
Asynchronous Arithmetic	$f_{BAUD} \leq \frac{f_{ref}}{16}$	$f_{BAUD} = \frac{f_{ref}}{16} \left(1 - \frac{BAUD}{65536}\right)$	$BAUD = 65536 \cdot \left(1 - 16 \cdot \frac{f_{BAUD}}{f_{ref}}\right)$
Asynchronous Fractional	$f_{BAUD} \leq \frac{f_{ref}}{S}$	$f_{BAUD} = \frac{f_{ref}}{S \cdot \left(BAUD + \frac{FP}{8}\right)}$	$BAUD = \frac{f_{ref}}{S \cdot f_{BAUD}} - \frac{FP}{8}$
Synchronous	$f_{BAUD} \leq \frac{f_{ref}}{2}$	$f_{BAUD} = \frac{f_{ref}}{2 \cdot (BAUD + 1)}$	$BAUD = \frac{f_{ref}}{2 \cdot f_{BAUD}} - 1$

S - Number of samples per bit, which 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:

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

**Asynchronous Arithmetic Mode BAUD Value Selection**

The formula given for  $f_{BAUD}$  calculates the average frequency over 65536  $f_{ref}$  cycles. Although the BAUD register can be set to any value between 0 and 65536, the actual average frequency of  $f_{BAUD}$  over a single frame is more granular. 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 27-3 shows the BAUD register value versus baud frequency  $f_{\text{BAUD}}$  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 27-3. BAUD Register Value vs. Baud Frequency**

BAUD Register Value	Serial Engine CPF	$f_{\text{BAUD}}$ at 48MHz Serial Engine Frequency ( $f_{\text{REF}}$ )
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

### 27.6.3 Additional Features

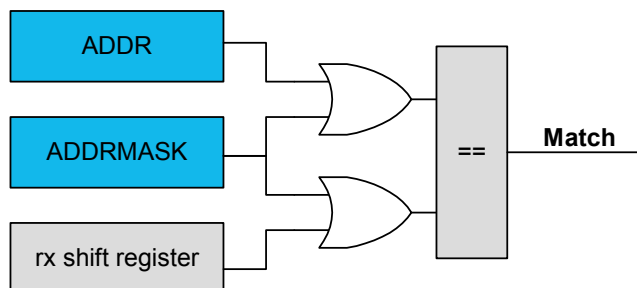
#### 27.6.3.1 Address Match and Mask

The SERCOM address match and mask feature is capable of matching either one address, two unique addresses, or a range of addresses with a mask, 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), and 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 writing the ADDR.ADDRMASK to 'all zeros' will match a single unique address, while writing ADDR.ADDRMASK to 'all ones' will result in all addresses being accepted.

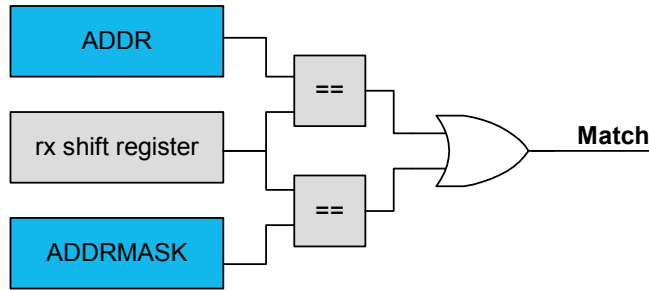
**Figure 27-4. Address With Mask**



##### Two Unique Addresses

The two addresses written to ADDR and ADDRMASK will cause a match.

Figure 27-5. 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 27-6. Address Range



**27.6.4 DMA Operation**

The available DMA interrupts and their depend on the operation mode of the SERCOM peripheral. Refer to the Functional Description sections of the respective SERCOM mode.

**Related Links**

- [SERCOM USART](#)
- [SERCOM SPI – SERCOM Serial Peripheral Interface](#)
- [SERCOM I2C – SERCOM Inter-Integrated Circuit](#)

**27.6.5 Interrupts**

Interrupt sources are mode-specific. See the respective SERCOM mode chapters for details.

Each interrupt source has its own interrupt flag.

The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the interrupt condition is met.

Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' 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 either the interrupt flag is cleared, the interrupt is disabled, or the SERCOM is reset. For details on clearing interrupt flags, refer to the INTFLAG register description.

The value of INTFLAG indicates which interrupt condition occurred. The user must read the INTFLAG register to determine which interrupt condition is present.

**Note:** Interrupts must be globally enabled for interrupt requests.

**Related Links**

- [Nested Vector Interrupt Controller](#)

**27.6.6 Events**

Not applicable.



### **27.6.7 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.

### **27.6.8 Synchronization**

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

Required read-synchronization is denoted by the "Read-Synchronized" property in the register description.

#### **Related Links**

[Register Synchronization](#)

## **28. SERCOM USART**

### **28.1 Overview**

The Universal Synchronous and Asynchronous Receiver and Transmitter (USART) is one of the available modes in the Serial Communication Interface (SERCOM).

The USART uses the SERCOM transmitter and receiver, see [Block Diagram](#). Labels in uppercase letters are synchronous to CLK\_SERCOMx\_APB and accessible for CPU. Labels in lowercase letters can be programmed 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 different frame formats. The write buffer support data transmission without any delay between frames. The receiver consists of a two-level receive buffer and a shift register. Status information of the received data is available for error checking. Data and clock recovery units ensure robust synchronization and noise filtering during asynchronous data reception.

#### **Related Links**

[SERCOM – Serial Communication Interface](#)

### **28.2 USART Features**

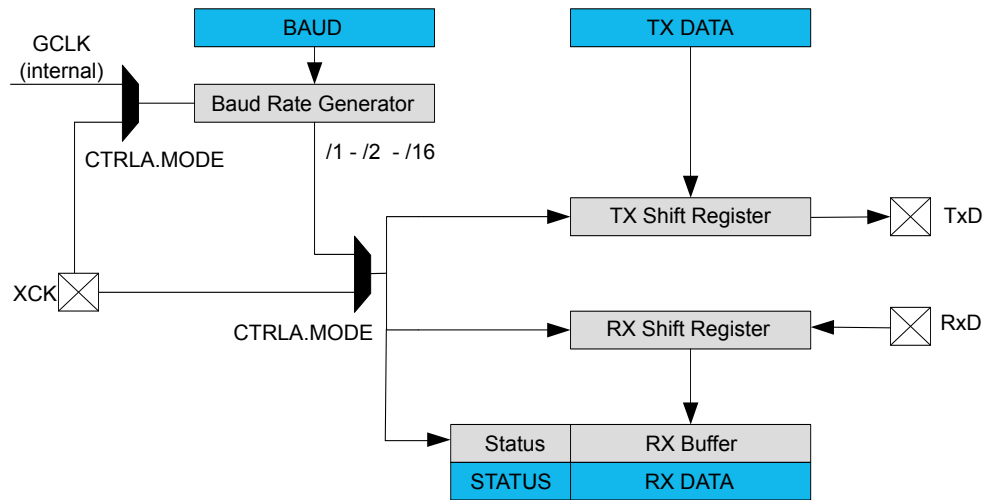
- Full-duplex operation
- Asynchronous (with clock reconstruction) or synchronous operation
- Internal or external clock source for asynchronous and synchronous operation
- Baud-rate generator
- Supports serial frames with 5, 6, 7, 8 or 9 data bits and 1 or 2 stop bits
- Odd or even parity generation 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 115.2kbps
- LIN slave support
  - Auto-baud and break character detection
- Start-of-frame detection
- Can work with DMA

#### **Related Links**

[Features](#)

## 28.3 Block Diagram

Figure 28-1. USART Block Diagram



## 28.4 Signal Description

Table 28-1. SERCOM USART Signals

Signal Name	Type	Description
PAD[3:0]	Digital I/O	General SERCOM pins

One signal can be mapped to one of several pins.

### Related Links

[I/O Multiplexing and Considerations](#)

## 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 USART's I/O lines requires the I/O pins to be configured using the I/O Pin Controller (PORT).

When the SERCOM is used in USART mode, the SERCOM controls the direction and value of the I/O pins according to the table below. Both PORT control bits PINCFGn.PULLEN and PINCFGn.DRVSTR are still effective. If the receiver or transmitter is disabled, these pins can be used for other purposes.

Table 28-2. 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 fields in the Control A register (CTRLA.TXPO and CTRLA.RXPO, respectively) will define the physical position of the USART signals in [Table 28-2](#).

### Related Links

[PORT: IO Pin Controller](#)

## 28.5.2 Power Management

This peripheral can continue to operate in any sleep mode where its source clock is running. The interrupts can wake up the device from sleep modes.

### Related Links

[PM – Power Manager](#)

## 28.5.3 Clocks

The SERCOM bus clock (CLK\_SERCOMx\_APB) can be enabled and disabled in the Power Manager. Refer to *Peripheral Clock Masking* for details and default status of this clock.

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* for details.

This generic clock is asynchronous to the bus clock (CLK\_SERCOMx\_APB). Therefore, writing to certain registers will require synchronization to the clock domains. Refer to *Synchronization* for further details.

### Related Links

[Synchronization](#)

[GCLK - Generic Clock Controller](#)

[Peripheral Clock Masking](#)

## 28.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). In order to use DMA requests with this peripheral the DMAC must be configured first. Refer to *DMAC – Direct Memory Access Controller* for details.

### Related Links

[DMAC – Direct Memory Access Controller](#)

## 28.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. Refer to *Nested Vector Interrupt Controller* for details.

### Related Links

[Nested Vector Interrupt Controller](#)

## 28.5.6 Events

Not applicable.

## 28.5.7 Debug Operation

When the CPU is halted in debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or

data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

**Related Links**

[DBGCTRL](#)

**28.5.8 Register Access Protection**

Registers with write-access can be write-protected optionally by the peripheral access controller (PAC).

PAC Write-Protection is not available for the following registers:

- Interrupt Flag Clear and Status register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)

Optional PAC Write-Protection is denoted by the "PAC Write-Protection" property in each individual register description.

Write-protection does not apply to accesses through an external debugger.

**Related Links**

[PAC - Peripheral Access Controller](#)

**28.5.9 Analog Connections**

Not applicable.

**28.6 Functional Description**

**28.6.1 Principle of Operation**

The USART uses the following lines for data transfer:

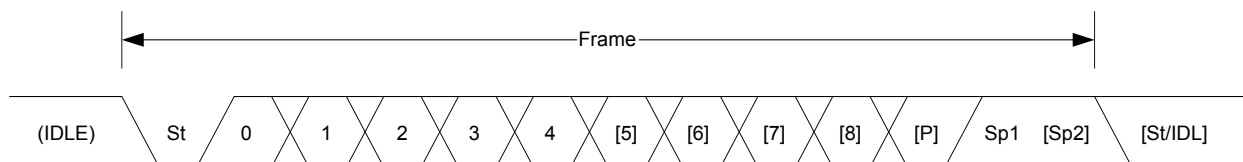
- RxD for receiving
- TxD for transmitting
- XCK for the transmission clock in synchronous operation

USART data transfer is frame based. A serial frame consists of:

- 1 start bit
- From 5 to 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. After the stop bit(s) of a frame, either the next frame can follow immediately, or the communication line can return to the idle (high) state. The figure below illustrates the possible frame formats. Brackets denote optional bits.

**Figure 28-2. Frame Formats**



**St** Start bit. Signal is always low.

**n, [n]** Data bits. 0 to [5..9]

**[P]** Parity bit. Either odd or even.

**Sp, [Sp]** Stop bit. Signal is always high.

**IDLE** No frame is transferred on the communication line. Signal is always high in this state.

## 28.6.2 Basic Operation

### 28.6.2.1 Initialization

The following registers are enable-protected, meaning they can only be written when the USART is disabled (CTRL.ENABLE=0):

- 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)

When the USART is enabled or is being enabled (CTRLA.ENABLE=1), any writing attempt to these registers will be discarded. If the peripheral is being disabled, writing to these registers will be executed after disabling is completed. Enable-protection is denoted by the "Enable-Protection" property in the register description.

Before the USART is enabled, it must be configured by these steps:

1. Select either external (0x0) or internal clock (0x1) by writing the Operating Mode value in the CTRLA register (CTRLA.MODE).
2. Select either asynchronous (0) or synchronous (1) communication mode by writing the Communication Mode bit in the CTRLA register (CTRLA.CMODE).
3. Select pin for receive data by writing the Receive Data Pinout value in the CTRLA register (CTRLA.RXPO).
4. Select pads for the transmitter and external clock by writing the Transmit Data Pinout bit in the CTRLA register (CTRLA.TXPO).
5. Configure the Character Size field in the CTRLB register (CTRLB.CHSIZE) for character size.
6. Set the Data Order bit in the CTRLA register (CTRLA.DORD) to determine MSB- or LSB-first data transmission.
7. To use parity mode:
  - 7.1. Enable parity mode by writing 0x1 to the Frame Format field in the CTRLA register (CTRLA.FORM).
  - 7.2. Configure the Parity Mode bit in the CTRLB register (CTRLB.PMODE) for even or odd parity.
8. Configure the number of stop bits in the Stop Bit Mode bit in the CTRLB register (CTRLB.SBMODE).
9. When using an internal clock, write the Baud register (BAUD) to generate the desired baud rate.
10. Enable the transmitter and receiver by writing '1' to the Receiver Enable and Transmitter Enable bits in the CTRLB register (CTRLB.RXEN and CTRLB.TXEN).

### 28.6.2.2 Enabling, Disabling, and Resetting

This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST) will reset all registers of this peripheral to their initial states, except the DBGCTRL register, and the peripheral is disabled.

Refer to the CTRLA register description for details.

### 28.6.2.3 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.

The synchronous mode is selected by writing a '1' to the Communication Mode bit in the Control A register (CTRLA.CMODE), the asynchronous mode is selected by writing a zero to CTRLA.CMODE.

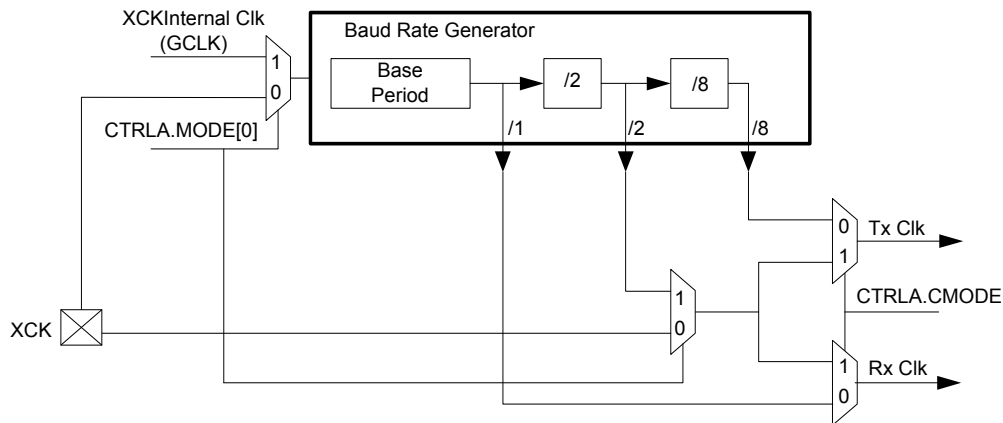
The internal clock source is selected by writing 0x1 to the Operation Mode bit field in the Control A register (CTRLA.MODE), the external clock source is selected by writing 0x0 to CTRLA.MODE.

The SERCOM baud-rate generator is configured as in the figure below.

In asynchronous mode (CTRLA.CMODE=0), the 16-bit Baud register value is used.

In synchronous mode (CTRLA.CMODE=1), the eight LSBs of the Baud register are used. Refer to *Clock Generation – Baud-Rate Generator* for details on configuring the baud rate.

**Figure 28-3. Clock Generation**



#### Related Links

[Clock Generation – Baud-Rate Generator](#)

[Asynchronous Arithmetic Mode BAUD Value Selection](#)

#### Synchronous Clock Operation

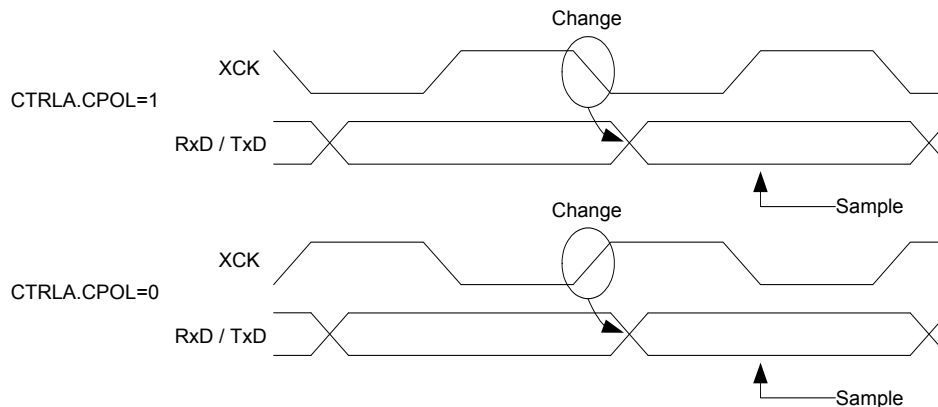
In synchronous mode, the CTRLA.MODE bit field determines whether the transmission clock line (XCK) serves either as input or output. The dependency between clock edges, data sampling, and data change is the same for internal and external clocks. Data input on the RxD pin is sampled at the opposite XCK clock edge when 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:

When CTRLA.CPOL is '0', the data will be changed on the rising edge of XCK, and sampled on the falling edge of XCK.

When CTRLA.CPOL is '1', the data will be changed on the falling edge of XCK, and sampled on the rising edge of XCK.

**Figure 28-4. Synchronous Mode XCK Timing**



When the clock is provided through XCK (CTRLA.MODE=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.

#### 28.6.2.4 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 TxDATA register. Reading the DATA register will return the contents of the RxDATA register.

#### 28.6.2.5 Data Transmission

Data transmission is initiated by writing the data to be sent into the DATA register. Then, the data in TxDATA will be moved to the shift register when the shift register is empty and ready to send a new frame. After the shift register is loaded with data, the data frame will be transmitted.

When the entire data frame including stop bit(s) has been transmitted and no new data was written to DATA, the Transmit Complete interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.TXC) will be set, and the optional interrupt will be generated.

The Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) indicates that the register is empty and ready for new data. The DATA register should only be written to when INTFLAG.DRE is set.

#### Disabling the Transmitter

The transmitter is disabled by writing '0' to the Transmitter Enable bit in the CTRLB register (CTRLB.TXEN).

Disabling the transmitter will complete only after any ongoing and pending transmissions are completed, i.e., there is no data in the transmit shift register and TxDATA to transmit.

#### 28.6.2.6 Data Reception

The receiver accepts data when a valid start bit is detected. Each bit following the start bit will be sampled according to the baud rate or XCK clock, and shifted into the receive shift register until the first stop bit of a frame is received. The second stop bit will be ignored by the receiver.

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. Then, the Receive Complete interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) will be set, and the optional interrupt will be generated.



The received data can be read from the DATA register when the Receive Complete interrupt flag is set.

### Disabling the Receiver

Writing '0' to the Receiver Enable bit in the CTRLB register (CTRLB.RXEN) will disable the receiver, flush the two-level receive buffer, and data from ongoing receptions will be lost.

### Error Bits

The USART receiver has three error bits in the Status (STATUS) register: Frame Error (FERR), Buffer Overflow (BUFOVF), and Parity Error (PERR). Once an error happens, the corresponding error bit will be set until it is cleared by writing '1' to it. These bits are also cleared automatically when the receiver is disabled.

There are two methods for buffer overflow notification, selected by the Immediate Buffer Overflow Notification bit in the Control A register (CTRLA.IBON):

When CTRLA.IBON=1, STATUS.BUFOVF is raised immediately upon buffer overflow. Software can then empty the receive FIFO by reading RxDATA, until the receiver complete interrupt flag (INTFLAG.RXC) is cleared.

When CTRLA.IBON=0, the buffer overflow condition is attending 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 can 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.

### Asynchronous Operational Range

The operational range of the asynchronous reception depends on the accuracy of the internal baud-rate clock, the rate of the incoming frames, and the frame size (in number of bits). In addition, the operational range of the receiver is depending 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: First, the reference clock will always have some minor instability. Second, the baud-rate generator cannot 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 set to give the lowest possible error. Refer to *Clock Generation – Baud-Rate Generator* for details.

Recommended maximum receiver baud-rate errors for various character sizes are shown in the table below.

**Table 28-3. Asynchronous Receiver Error for 16-fold Oversampling**

D (Data bits+Parity)	R <sub>SLOW</sub> [%]	R <sub>FAST</sub> [%]	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

D (Data bits+Parity)	R <sub>SLOW</sub> [%]	R <sub>FAST</sub> [%]	Max. total error [%]	Recommended max. Rx error [%]
9	96.39	104.76	+3.61/-4.76	±1.5
10	96.70	104.35	+3.30/-4.35	±1.5

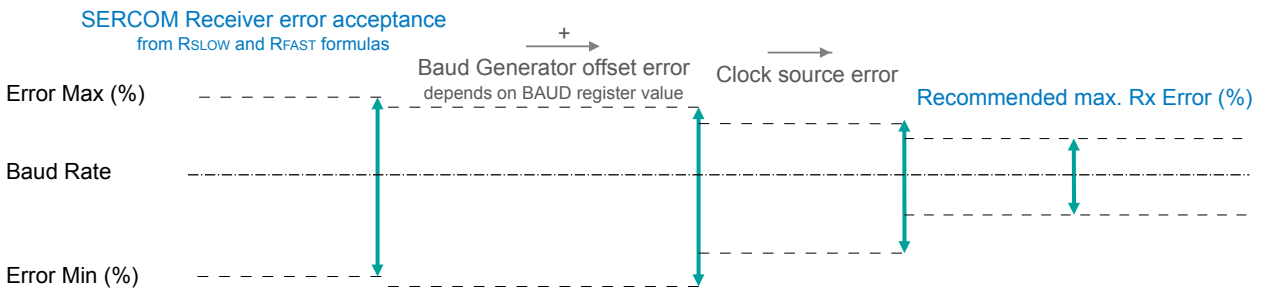
The following equations calculate the ratio of the incoming data rate and internal receiver baud rate:

$$R_{SLOW} = \frac{(D + 1)S}{S - 1 + D \cdot S + S_F} \quad , \quad R_{FAST} = \frac{(D + 2)S}{(D + 1)S + S_M}$$

- $R_{SLOW}$  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
- $D$  is the sum of character size and parity size ( $D = 5$  to  $10$  bits)
- $S$  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.

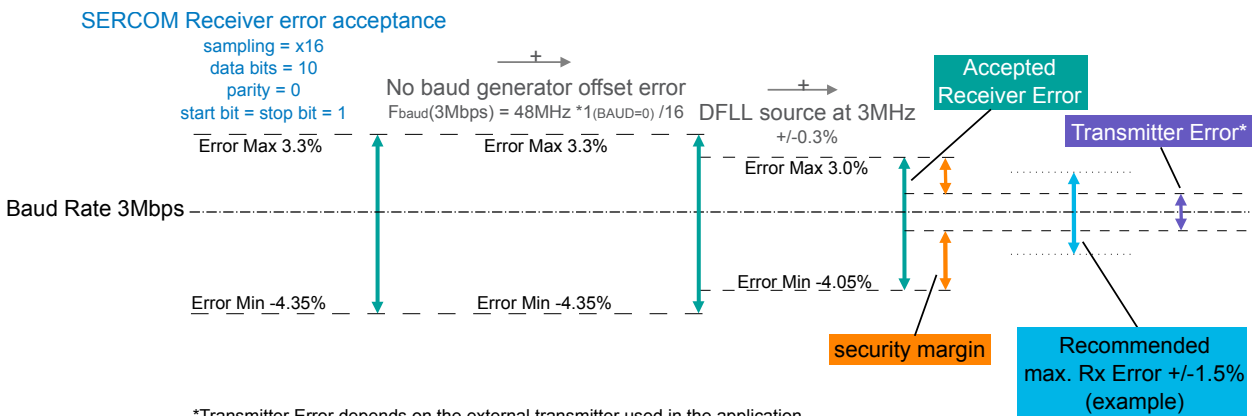
The recommended maximum Rx Error assumes that the receiver and transmitter equally divide the maximum total error. Its connection to the SERCOM Receiver error acceptance is depicted in this figure:

**Figure 28-5. USART Rx Error Calculation**



The recommendation values in the table above accommodate errors of the clock source and the baud generator. The following figure gives an example for a baud rate of 3Mbps:

**Figure 28-6. USART Rx Error Calculation Example**



\*Transmitter Error depends on the external transmitter used in the application. It is advised that it is within the Recommended max. Rx Error (+/-1.5% in this example). Larger Transmitter Errors are acceptable but must lie within the Accepted Receiver Error.

### Related Links

Clock Generation – Baud-Rate Generator  
Asynchronous Arithmetic Mode BAUD Value Selection

**28.6.3 Additional Features**

**28.6.3.1 Parity**

Even or odd parity can be selected for error checking by writing 0x1 to the Frame Format bit field in the Control A register (CTRLA.FORM).

If *even parity* is selected (CTRLB.PMODE=0), the parity bit of an outgoing frame is '1' if the data contains an odd number of bits that are '1', making the total number of '1' even.

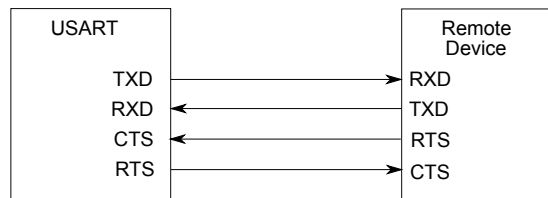
If *odd parity* is selected (CTRLB.PMODE=1), the parity bit of an outgoing frame is '1' if the data contains an even number of bits that are '0', making the total number of '1' odd.

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.

**28.6.3.2 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 the figure below.

**Figure 28-7. Connection with a Remote Device for Hardware Handshaking**

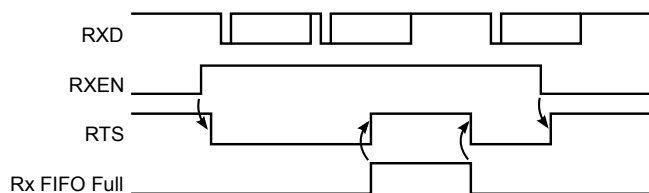


Hardware handshaking is only available in the following configuration:

- USART with internal clock (CTRLA.MODE=1),
- Asynchronous mode (CTRLA.CMODE=0),
- and Flow control pinout (CTRLA.TXPO=2).

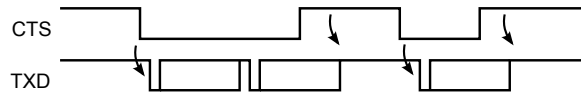
When the receiver is disabled or the receive FIFO is full, the receiver will drive the RTS pin high. This notifies the remote device to stop transfer after the ongoing transmission. Enabling and disabling the receiver by writing to CTRLB.RXEN will set/clear the RTS pin after a synchronization delay. When the receive FIFO goes full, RTS will be set immediately and the frame being received will be stored in the shift register until the receive FIFO is no longer full.

**Figure 28-8. Receiver Behavior when Operating with Hardware Handshaking**



The current CTS Status is in the STATUS register (STATUS.CTS). Character transmission will start only if STATUS.CTS=0. When CTS is set, the transmitter will complete the ongoing transmission and stop transmitting.

**Figure 28-9. Transmitter Behavior when Operating with Hardware Handshaking**



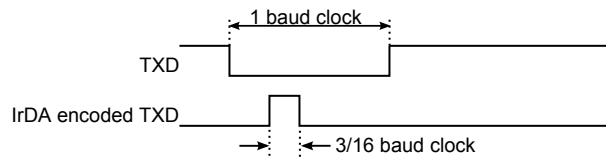
### 28.6.3.3 IrDA Modulation and Demodulation

Transmission and reception can be encoded IrDA compliant up to 115.2 kb/s. IrDA modulation and demodulation work in the following configuration:

- IrDA encoding enabled (CTRLB.ENC=1),
- Asynchronous mode (CTRLA.CMODE=0),
- and 16x sample rate (CTRLA.SAMPR[0]=0).

During transmission, each low bit is transmitted as a high pulse. The pulse width is 3/16 of the baud rate period, as illustrated in the figure below.

**Figure 28-10. IrDA Transmit Encoding**



The reception decoder has two main functions.

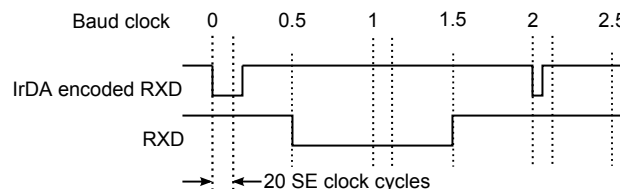
The first is to synchronize the incoming data to the IrDA baud rate counter. Synchronization is performed at the start of each zero pulse.

The second main 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.

**Note:** Note that the polarity of the transmitter and receiver are opposite: During transmission, a '0' bit is transmitted as a '1' pulse. During reception, an accepted '0' pulse is received as a '0' bit.

**Example:** The figure below illustrates reception where RXPL.RXPL is set to 19. This indicates that the pulse width should be at least 20 SE clock cycles. When using 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 '0', the second bit is a '1', and the third bit is also a '1'. A low pulse is rejected since it does not meet the minimum requirement of 2/16 baud clock.

**Figure 28-11. IrDA Receive Decoding**



### 28.6.3.4 Break Character Detection and Auto-Baud/LIN Slave

Break character detection and auto-baud are available in this configuration:

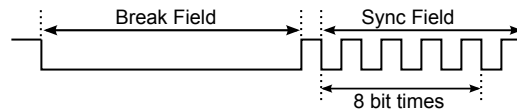
- Auto-baud frame format (CTRLA.FORM = 0x04 or 0x05),
- Asynchronous mode (CTRLA.CMODE = 0),

- and 16x sample rate using fractional baud rate generation (CTRLA.SAMPR = 1).

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.

The auto-baud follows the LIN format. All LIN Frames start with a Break Field followed by a Sync Field.

**Figure 28-12. 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) give the new clock divider (BAUD.BAUD), and the 3 least significant bits of this value (the remainder) give the new Fractional Part (BAUD.FP).

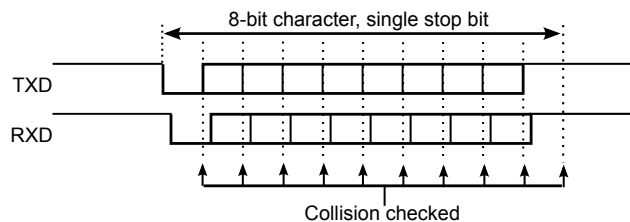
When the Sync Field has been received, the clock divider (BAUD.BAUD) and the Fractional Part (BAUD.FP) are updated after a synchronization delay. After the Break and Sync Fields are received, multiple characters of data can be received.

### 28.6.3.5 Collision Detection

When the receiver and transmitter are connected either through pin configuration or externally, transmit collision can be detected after selecting the Collision Detection Enable bit in the CTRLB register (CTRLB.COLDEN=1). To detect collision, the receiver and transmitter must be enabled (CTRLB.RXEN=1 and CTRLB.TXEN=1).

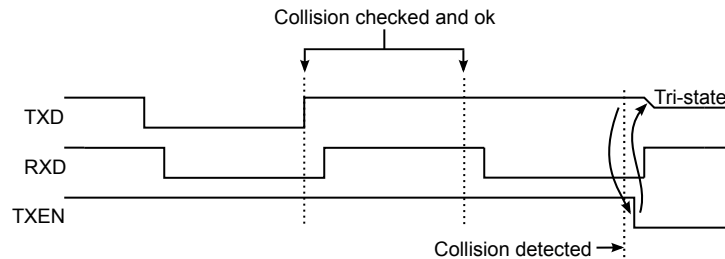
Collision detection is performed for each bit transmitted by comparing the received value with the transmit value, as shown in the figure below. While the transmitter is idle (no transmission in progress), characters can be received on RxD without triggering a collision.

**Figure 28-13. Collision Checking**



The next figure 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 28-14. Collision Detected**



When a collision is detected, the USART follows this sequence:

1. Abort the current transfer.
2. Flush the transmit buffer.
3. Disable transmitter (CTRLB.TXEN=0)
  - This is done after a synchronization delay. The CTRLB Synchronization Busy bit (SYNCBUSY.CTRLB) will be set until this is complete.
  - After disabling, the TxD pin will be tri-stated.
4. Set the Collision Detected bit (STATUS.COLL) along with the Error interrupt flag (INTFLAG.ERROR).
5. Set the Transmit Complete interrupt flag (INTFLAG.TXC), since the transmit buffer no longer contains data.

After a collision, software must manually enable the transmitter again before continuing, after assuring that the CTRLB Synchronization Busy bit (SYNCBUSY.CTRLB) is not set.

#### 28.6.3.6 Loop-Back Mode

For loop-back mode, configure the Receive Data Pinout (CTRLA.RXPO) and Transmit Data Pinout (CTRLA.TXPO) to use the same data pins for transmit and receive. The loop-back is through the pad, so the signal is also available externally.

#### 28.6.3.7 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* 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 '1' 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. In this case, the CPU will not wake up until the Receive Complete interrupt is generated.

#### Related Links

[Electrical Characteristics](#)

### 28.6.3.8 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=0, samples 7-8-9 are used for 16x oversampling, and samples 3-4-5 are used for 8x oversampling.

### 28.6.4 DMA, Interrupts and Events

#### 28.6.4.1 DMA Operation

The USART generates the following DMA requests:

#### 28.6.4.2 Interrupts

The USART has the following interrupt sources. These are asynchronous interrupts, and can wake up the device from any sleep mode:

- Data Register Empty (DRE)
- Receive Complete (RXC)
- Transmit Complete (TXC)
- Receive Start (RXS)
- Clear to Send Input Change (CTSIC)
- Received Break (RXBRK)
- Error (ERROR)

Each interrupt source has its own interrupt flag. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the interrupt condition is met. Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' to the corresponding bit in the Interrupt Enable Clear register (INTENCLR).

An interrupt request is generated when the interrupt flag is set and if the corresponding interrupt is enabled. The interrupt request remains active until either the interrupt flag is cleared, the interrupt is disabled, or the USART is reset. For details on clearing interrupt flags, refer to the INTFLAG register description.

The value of INTFLAG indicates which interrupt is executed. Note that interrupts must be globally enabled for interrupt requests. Refer to *Nested Vector Interrupt Controller* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

#### 28.6.4.3 Events

Not applicable.

#### 28.6.5 Sleep Mode Operation

The behavior in sleep mode is depending on the clock source and the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY):

- Internal clocking, CTRLA.RUNSTDBY=1: GCLK\_SERCOMx\_CORE can be enabled in all sleep modes. Any interrupt can wake up the device.
- External clocking, CTRLA.RUNSTDBY=1: The Receive Start and the Receive Complete interrupt(s) can wake up the device.
- Internal clocking, CTRLA.RUNSTDBY=0: Internal clock will be disabled, after any ongoing transfer was completed. The Receive Start and the Receive Complete interrupt(s) can wake up the device.

- External clocking, CTRLA.RUNSTDBY=0: External clock will be disconnected, after any ongoing transfer was completed. All reception will be dropped.

### 28.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in the CTRLA register (CTRLA.SWRST)
- Enable bit in the CTRLA register (CTRLA.ENABLE)
- Receiver Enable bit in the CTRLB register (CTRLB.RXEN)
- Transmitter Enable bit in the Control B register (CTRLB.TXEN)

**Note:** CTRLB.RXEN is write-synchronized somewhat differently. See also [CTRLB](#) for details.

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

#### Related Links

[Register Synchronization](#)



## 28.7 Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
		15:8	SAMPR[2:0]							IBON
		23:16	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
		31:24	DORD	CPOL	CMODE	FORM[3:0]				
0x04	CTRLB	7:0		SBMODE				CHSIZE[2:0]		
		15:8			PMODE			ENC	SFDE	COLDEN
		23:16							RXEN	TXEN
		31:24								
0x08	CTRLC	7:0								
		15:8								
		23:16								
		31:24								
0x0C	BAUD	7:0	BAUD[7:0]							
		15:8	BAUD[15:8]							
0x0E	RXPL	7:0	RXPL[7:0]							
0x0F ... 0x13	Reserved									
0x14	INTENCLR	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x15	Reserved									
0x16	INTENSET	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x19	Reserved									
0x1A	STATUS	7:0		TXE	COLL	ISF	CTS	BUFOVF	FERR	PERR
		15:8								
0x1C	SYNCBUSY	7:0						CTRLB	ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x20 ... 0x27	Reserved									
0x28	DATA	7:0	DATA[7:0]							
		15:8								DATA[8:8]
0x2A ... 0x2F	Reserved									
0x30	DBGCTRL	7:0								DBGSTOP

## **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 require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module 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:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
		DORD	CPOL	CMODE	FORM[3:0]			
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		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/W	R/W
Reset	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
Reset	0	0	0					0
Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

#### Bit 30 – DORD Data Order

This bit selects the data order when a character is shifted out from the Data register.

This bit is not synchronized.

Value	Description
0	MSB is transmitted first.
1	LSB is transmitted first.

#### Bit 29 – CPOL Clock Polarity

This bit selects the relationship between data output change and data input sampling in synchronous mode.

This bit is not synchronized.

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 selects asynchronous or synchronous communication.

This bit is not synchronized.

Value	Description
0	Asynchronous communication.
1	Synchronous communication.

**Bits 27:24 – FORM[3:0]** Frame Format

These bits define the frame format.

These bits are not synchronized.

FORM[3:0]	Description
0x0	USART frame
0x1	USART frame with parity
0x2-0x3	Reserved
0x4	Auto-baud (LIN Slave) - 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.

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.

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 17:16 – TXPO[1:0]** Transmit Data Pinout

These bits define the transmit data (TxD) and XCK pin configurations.

This bit is not synchronized.

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 select the sample rate.

These bits are not synchronized.

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

**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.

Value	Description
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.

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 4:2 – MODE[2:0]** Operating Mode

These bits select the USART serial communication interface of the SERCOM.

These bits are not synchronized.

Value	Description
0x0	USART with external clock
0x1	USART with internal clock

**Bit 1 – ENABLE** Enable

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.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled or being enabled.

**Bit 0 – SWRST** Software Reset

Writing '0' to this bit has no effect.

Writing '1' to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.

Writing '1' 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.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

**28.8.2 Control B**

**Name:** CTRLB  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	[Greyed out bits 31-24]							
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	[Greyed out bits 23-18]						RXEN	TXEN
Access							R/W	R/W
Reset							0	0
Bit	15	14	13	12	11	10	9	8
	[Greyed out bits 15-14]		PMODE	[Greyed out bits 12-11]		ENC	SFDE	COLDEN
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0
Bit	7	6	5	4	3	2	1	0
	[Greyed out bits 7-6]		SBMODE	[Greyed out bits 4-3]		CHSIZE[2:0]		
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

**Bit 17 – RXEN** Receiver Enable

Writing '0' 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 '1' 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 '1'.

Writing '1' 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 '1'.

This bit is not enable-protected.

Value	Description
0	The receiver is disabled or being enabled.
1	The receiver is enabled or will be enabled when the USART is enabled.

**Bit 16 – TXEN** Transmitter Enable

Writing '0' to this bit will disable the USART transmitter. Disabling the transmitter will not become effective until ongoing and pending transmissions are completed.

Writing '1' 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 '1'.

Writing '1' to CTRLB.TXEN when the USART is enabled will set SYNCBUSY.CTRLB, which will remain set until the transmitter is enabled, and CTRLB.TXEN will read back as '1'.

This bit is not enable-protected.

Value	Description
0	The transmitter is disabled or being enabled.
1	The transmitter is enabled or will be enabled when the USART is enabled.

**Bit 13 – PMODE** Parity Mode

This bit selects the type of parity used when parity is enabled (CTRLA.FORM is '1'). 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.

This bit is not synchronized.

Value	Description
0	Even parity.
1	Odd parity.

**Bit 10 – ENC** Encoding Format

This bit selects the data encoding format.

This bit is not synchronized.

Value	Description
0	Data is not encoded.
1	Data is IrDA encoded.

**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.

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 8 – COLDEN** Collision Detection Enable

This bit enables collision detection.

This bit is not synchronized.



Value	Description
0	Collision detection is not enabled.
1	Collision detection is enabled.

**Bit 6 – SBMODE** Stop Bit Mode

This bit selects the number of stop bits transmitted.

This bit is not synchronized.

Value	Description
0	One stop bit.
1	Two stop bits.

**Bits 2:0 – CHSIZE[2:0]** Character Size

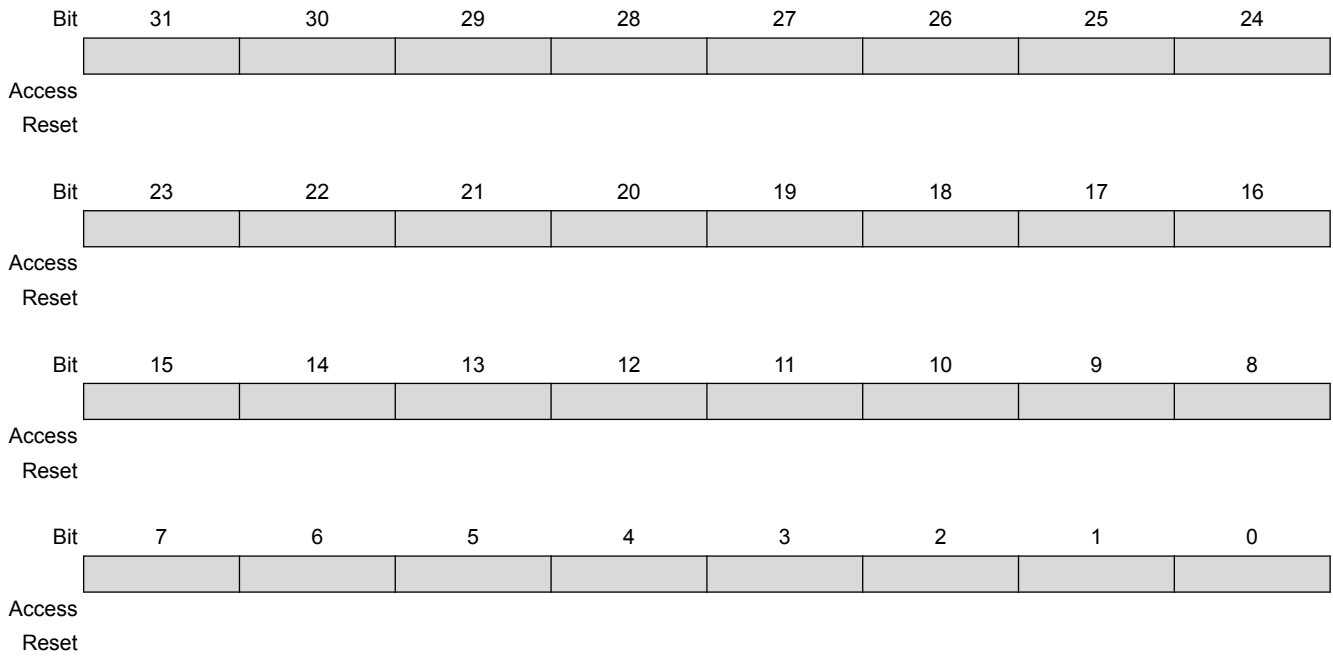
These bits select the number of bits in a character.

These bits are not synchronized.

CHSIZE[2:0]	Description
0x0	8 bits
0x1	9 bits
0x2-0x4	Reserved
0x5	5 bits
0x6	6 bits
0x7	7 bits

28.8.3 Control C

**Name:** CTRLC  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



### 28.8.4 Baud

**Name:** BAUD  
**Offset:** 0x0C  
**Reset:** 0x0000  
**Property:** Enable-Protected, PAC Write-Protection

Bit	15	14	13	12	11	10	9	8
BAUD[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
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 15:0 – BAUD[15:0] Baud Value**  
 Arithmetic Baud Rate Generation ( $CTRLA.SAMPR[0]=0$ ):

These bits control the clock generation, as described in the SERCOM Baud Rate section.

If Fractional Baud Rate Generation ( $CTRLA.SAMPR[0]=1$ ) bit positions 15 to 13 are replaced by FP[2:0] Fractional Part:

- **Bits 15:13 - FP[2:0]: Fractional Part**

These bits control the clock generation, as described in the *SERCOM Clock Generation – Baud-Rate Generator* section.

- **Bits 12:0 - BAUD[12:0]: Baud Value**

These bits control the clock generation, as described in the *SERCOM Clock Generation – Baud-Rate Generator* section.

**Related Links**

- [Clock Generation – Baud-Rate Generator](#)
- [Asynchronous Arithmetic Mode BAUD Value Selection](#)

### 28.8.5 Receive Pulse Length Register

**Name:** RXPL  
**Offset:** 0x0E  
**Reset:** 0x00  
**Property:** Enable-Protected, PAC Write-Protection

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  $SE_{per}$ .

$$PULSE \geq (RXPL + 2) \cdot SE_{per}$$

### 28.8.6 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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	6	5	4	3	2	1	0
	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

#### Bit 7 – ERROR Error Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

#### Bit 5 – RXBRK Receive Break Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Receive Break Interrupt Enable bit, which disables the Receive Break interrupt.

Value	Description
0	Receive Break interrupt is disabled.
1	Receive Break interrupt is enabled.

#### Bit 4 – CTSIC Clear to Send Input Change Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Clear To Send Input Change Interrupt Enable bit, which disables the Clear To Send Input Change interrupt.

Value	Description
0	Clear To Send Input Change interrupt is disabled.
1	Clear To Send Input Change interrupt is enabled.

#### Bit 3 – RXS Receive Start Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Receive Start Interrupt Enable bit, which disables the Receive Start interrupt.

Value	Description
0	Receive Start interrupt is disabled.
1	Receive Start interrupt is enabled.

---

**Bit 2 – RXC** Receive Complete Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Receive Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

Value	Description
0	Receive Complete interrupt is disabled.
1	Receive Complete interrupt is enabled.

**Bit 1 – TXC** Transmit Complete Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Transmit Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

Value	Description
0	Transmit Complete interrupt is disabled.
1	Transmit Complete interrupt is enabled.

**Bit 0 – DRE** Data Register Empty Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Data Register Empty Interrupt Enable bit, which disables the Data Register Empty interrupt.

Value	Description
0	Data Register Empty interrupt is disabled.
1	Data Register Empty interrupt is enabled.

### 28.8.7 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x16  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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).

Bit	7	6	5	4	3	2	1	0
	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

#### Bit 7 – ERROR Error Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

#### Bit 5 – RXBRK Receive Break Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Receive Break Interrupt Enable bit, which enables the Receive Break interrupt.

Value	Description
0	Receive Break interrupt is disabled.
1	Receive Break interrupt is enabled.

#### Bit 4 – CTSIC Clear to Send Input Change Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Clear To Send Input Change Interrupt Enable bit, which enables the Clear To Send Input Change interrupt.

Value	Description
0	Clear To Send Input Change interrupt is disabled.
1	Clear To Send Input Change interrupt is enabled.

#### Bit 3 – RXS Receive Start Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Receive Start Interrupt Enable bit, which enables the Receive Start interrupt.

Value	Description
0	Receive Start interrupt is disabled.
1	Receive Start interrupt is enabled.

---

---

**Bit 2 – RXC** Receive Complete Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Receive Complete Interrupt Enable bit, which enables the Receive Complete interrupt.

Value	Description
0	Receive Complete interrupt is disabled.
1	Receive Complete interrupt is enabled.

**Bit 1 – TXC** Transmit Complete Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Transmit Complete Interrupt Enable bit, which enables the Transmit Complete interrupt.

Value	Description
0	Transmit Complete interrupt is disabled.
1	Transmit Complete interrupt is enabled.

**Bit 0 – DRE** Data Register Empty Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Data Register Empty Interrupt Enable bit, which enables the Data Register Empty interrupt.

Value	Description
0	Data Register Empty interrupt is disabled.
1	Data Register Empty interrupt is enabled.



### 28.8.8 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
Access	R/W		R/W	R/W	R/W	R	R/W	R
Reset	0		0	0	0	0	0	0

**Bit 7 – ERROR** Error

This flag is cleared by writing '1' 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 '0' to this bit has no effect.

Writing '1' to this bit will clear the flag.

**Bit 5 – RXBRK** Receive Break

This flag is cleared by writing '1' to it.

This flag is set when auto-baud is enabled (CTRLA.FORM) and a break character is received.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the flag.

**Bit 4 – CTSIC** Clear to Send Input Change

This flag is cleared by writing a '1' to it.

This flag is set when a change is detected on the CTS pin.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the flag.

**Bit 3 – RXS** Receive Start

This flag is cleared by writing '1' 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 '1').

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Receive Start interrupt flag.

**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 '0' to this bit has no effect.

Writing '1' to this bit has no effect.

**Bit 1 – TXC** Transmit Complete

This flag is cleared by writing '1' 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 '0' to this bit has no effect.

Writing '1' 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 '0' to this bit has no effect.

Writing '1' to this bit has no effect.

**28.8.9 Status**

**Name:** STATUS  
**Offset:** 0x1A  
**Reset:** 0x0000  
**Property:** -

Bit	15	14	13	12	11	10	9	8
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
		TXE	COLL	ISF	CTS	BUFOVF	FERR	PERR
Access		R/W	R/W	R/W	R	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

**Bit 6 – TXE** Transmitter Empty  
This bit will always read back as zero.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

**Bit 5 – COLL** Collision Detected  
This bit is cleared by writing '1' 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 '0' to this bit has no effect.

Writing '1' to this bit will clear it.

**Bit 4 – ISF** Inconsistent Sync Field  
This bit is cleared by writing '1' 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 '0' to this bit has no effect.

Writing '1' 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 '0' to this bit has no effect.

Writing '1' 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 '1' 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 '0' to this bit has no effect.

Writing '1' 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 '1' 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 '0' to this bit has no effect.

Writing '1' 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 '1' to the bit or by disabling the receiver.

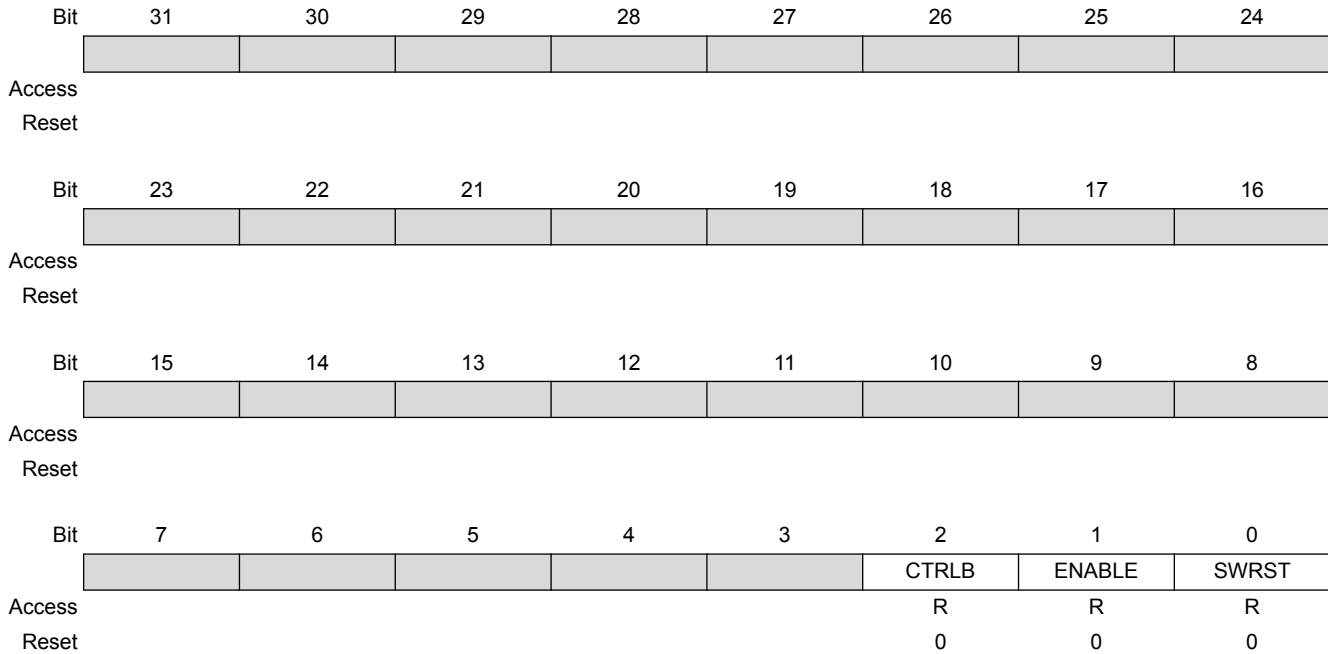
This bit is set if parity checking is enabled (CTRLA.FORM is 0x1, 0x5) and a parity error is detected.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

### 28.8.10 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x1C  
**Reset:** 0x00000000  
**Property:** -



**Bit 2 – CTRLB** CTRLB Synchronization Busy

Writing to the CTRLB register when the SERCOM is enabled requires synchronization. When writing to CTRLB the SYNCBUSY.CTRLB bit will be set until synchronization is complete. If CTRLB is written while SYNCBUSY.CTRLB is asserted, an APB error will be generated.

Value	Description
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 SYNCBUSY.ENABLE bit will be set until synchronization is complete.

Value	Description
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.

---

---

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

**28.8.11 Data**

**Name:** DATA  
**Offset:** 0x28  
**Reset:** 0x0000  
**Property:** -

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

**28.8.12 Debug Control**

**Name:** DBGCTRL  
**Offset:** 0x30  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

**Bit 0 – DBGSTOP** Debug Stop Mode

This bit controls the baud-rate generator functionality when the CPU is halted by an external debugger.

Value	Description
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.



## 29. SERCOM SPI – SERCOM Serial Peripheral Interface

### 29.1 Overview

The serial peripheral interface (SPI) is one of the available modes in the Serial Communication Interface (SERCOM).

The SPI uses the SERCOM transmitter and receiver configured as shown in [Block Diagram](#). Each side, master and slave, depicts a separate SPI containing a shift register, a transmit buffer and a two-level receive buffer. In addition, the SPI master uses the SERCOM baud-rate generator, while the SPI slave can use the SERCOM address match logic. Labels in capital letters are synchronous to CLK\_SERCOMx\_APB and accessible by the CPU, while labels in lowercase letters are synchronous to the SCK clock.

#### Related Links

[SERCOM – Serial Communication Interface](#)

### 29.2 Features

SERCOM SPI includes the following features:

- Full-duplex, four-wire interface (MISO, MOSI, SCK,  $\overline{SS}$ )
- One-level transmit buffer, two-level receive buffer
- 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,  $f_{SCK}=1/t_{SCK}^{(1)}$
  - 8-bit clock generator
  - Hardware controlled  $\overline{SS}$

1. For  $t_{SCK}$  and  $t_{SSCK}$  values, refer to SPI Timing Characteristics.

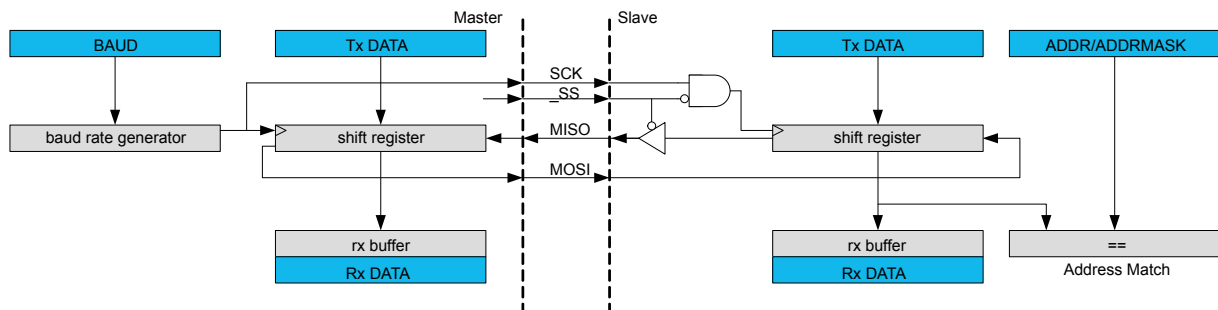
#### Related Links

[SERCOM – Serial Communication Interface](#)

[Features](#)

### 29.3 Block Diagram

Figure 29-1. Full-Duplex SPI Master Slave Interconnection



### 29.4 Signal Description

Table 29-1. SERCOM SPI Signals

Signal Name	Type	Description
PAD[3:0]	Digital I/O	General SERCOM pins

One signal can be mapped to one of several pins.

#### Related Links

[I/O Multiplexing and Considerations](#)

### 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

In order to use the SERCOM's I/O lines, the I/O pins must be configured using the IO Pin Controller (PORT).

When the SERCOM is configured for SPI operation, the SERCOM controls the direction and value of the I/O pins according to the table below. Both PORT control bits PINCFGn.PULLEN and PINCFGn.DRVSTR are still effective. 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 the Master Slave Select Enable bit in the Control B register (CTRLB.MSEN) is '1'.

Table 29-2. SPI Pin Configuration

Pin	Master SPI	Slave SPI
MOSI	Output	Input
MISO	Input	Output
SCK	Output	Input

The combined configuration of PORT, the Data In Pinout and the Data Out Pinout bit groups in the Control A register (CTRLA.DIPO and CTRLA.DOPO) define the physical position of the SPI signals in the table above.

**Related Links**[PORT: IO Pin Controller](#)**29.5.2 Power Management**

This peripheral can continue to operate in any sleep mode where its source clock is running. The interrupts can wake up the device from sleep modes.

**Related Links**[PM – Power Manager](#)**29.5.3 Clocks**

The SERCOM bus clock (CLK\_SERCOMx\_APB) can be enabled and disabled in the Power Manager. Refer to *Peripheral Clock Masking* for details and default status of this clock.

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.

This generic clock is asynchronous to the bus clock (CLK\_SERCOMx\_APB). Therefore, writes to certain registers will require synchronization to the clock domains.

**Related Links**[GCLK - Generic Clock Controller](#)[Peripheral Clock Masking](#)[Synchronization](#)**29.5.4 DMA**

The DMA request lines are connected to the DMA Controller (DMAC). In order to use DMA requests with this peripheral the DMAC must be configured first. Refer to *DMAC – Direct Memory Access Controller* for details.

**Related Links**[DMAC – Direct Memory Access Controller](#)**29.5.5 Interrupts**

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. Refer to *Nested Vector Interrupt Controller* for details.

**Related Links**[Nested Vector Interrupt Controller](#)**29.5.6 Events**

Not applicable.

**29.5.7 Debug Operation**

When the CPU is halted in debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

**29.5.8 Register Access Protection**

Registers with write-access can be write-protected optionally by the peripheral access controller (PAC).

PAC Write-Protection is not available for the following registers:

- Interrupt Flag Clear and Status register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)

Optional PAC Write-Protection is denoted by the "PAC Write-Protection" property in each individual register description.

Write-protection does not apply to accesses through an external debugger.

### Related Links

[PAC - Peripheral Access Controller](#)

### 29.5.9 Analog Connections

Not applicable.

## 29.6 Functional Description

### 29.6.1 Principle of Operation

The SPI is a high-speed synchronous data transfer interface. It allows high-speed communication between the device and peripheral devices.

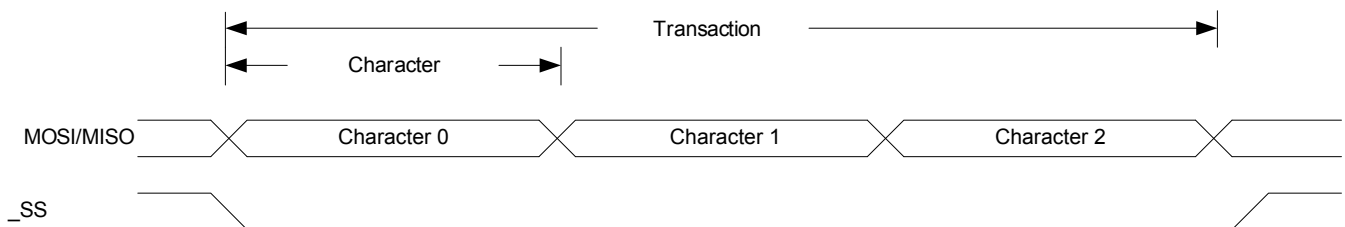
The SPI can operate as master or slave. As master, the SPI initiates and controls all data transactions. The SPI is single buffered for transmitting and double buffered for receiving.

When transmitting data, the Data register can be loaded with the next character to be transmitted during the current transmission.

When receiving, the data is transferred to the two-level receive buffer, and the receiver is ready for a new character.

The SPI transaction format is shown in [SPI Transaction Format](#). Each transaction can contain one or more characters. The character size is configurable, and can be either 8 or 9 bits.

**Figure 29-2. SPI Transaction Format**



The SPI master must pull the slave select line ( $\overline{SS}$ ) of the desired slave low to initiate a transaction. The master and slave prepare data to send via 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); data is shifted from slave to master on the Master Input Slave Output line (MISO).

Each time character is shifted out from the master, a character will be shifted out from the slave simultaneously. To signal the end of a transaction, the master will pull the  $\overline{SS}$  line high

### 29.6.2 Basic Operation

#### 29.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the SPI is disabled (CTRLA.ENABLE=0):

- 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)

When the SPI is enabled or is being enabled (CTRLA.ENABLE=1), any writing to these registers will be discarded.

when the SPI is being disabled, writing to these registers will be completed after the disabling.

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

Initialize the SPI by following these steps:

1. Select SPI mode in master / slave operation in the Operating Mode bit group in the CTRLA register (CTRLA.MODE= 0x2 or 0x3 ).
2. Select transfer mode for the Clock Polarity bit and the Clock Phase bit in the CTRLA register (CTRLA.CPOL and CTRLA.CPHA) if desired.
3. Select the Frame Format value in the CTRLA register (CTRLA.FORM).
4. Configure the Data In Pinout field in the Control A register (CTRLA.DIPO) for SERCOM pads of the receiver.
5. Configure the Data Out Pinout bit group in the Control A register (CTRLA.DOPO) for SERCOM pads of the transmitter.
6. Select the Character Size value in the CTRLB register (CTRLB.CHSIZE).
7. Write the Data Order bit in the CTRLA register (CTRLA.DORD) for data direction.
8. If the SPI is used in master mode:
  - 8.1. Select the desired baud rate by writing to the Baud register (BAUD).
  - 8.2. If Hardware SS control is required, write '1' to the Master Slave Select Enable bit in CTRLB register (CTRLB.MSSEN).
9. Enable the receiver by writing the Receiver Enable bit in the CTRLB register (CTRLB.RXEN=1).

#### 29.6.2.2 Enabling, Disabling, and Resetting

This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST) will reset all registers of this peripheral to their initial states, except the DBGCTRL register, and the peripheral is disabled.

Refer to the CTRLA register description for details.

#### 29.6.2.3 Clock Generation

In SPI master operation (CTRLA.MODE=0x3), the serial clock (SCK) is generated internally by the SERCOM baud-rate generator.

In SPI mode, the baud-rate generator is set to synchronous mode. The 8-bit Baud register (BAUD) value is used for generating SCK and clocking the shift register. Refer to *Clock Generation – Baud-Rate Generator* 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.

### Related Links

[Clock Generation – Baud-Rate Generator](#)

[Asynchronous Arithmetic Mode BAUD Value Selection](#)

### 29.6.2.4 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 DATA register will update the Transmit Data register. Reading the DATA register will return the contents of the Receive Data register.

### 29.6.2.5 SPI Transfer Modes

There are four combinations of SCK phase and polarity to transfer serial data. The SPI data transfer modes are shown in [SPI Transfer Modes \(Table\)](#) and [SPI Transfer Modes \(Figure\)](#).

SCK phase is configured by the Clock Phase bit in the CTRLA register (CTRLA.CPHA). SCK polarity is programmed by the Clock Polarity bit in the CTRLA register (CTRLA.CPOL). Data bits are shifted out and latched in on opposite edges of the SCK signal. This ensures sufficient time for the data signals to stabilize.

**Table 29-3. SPI Transfer Modes**

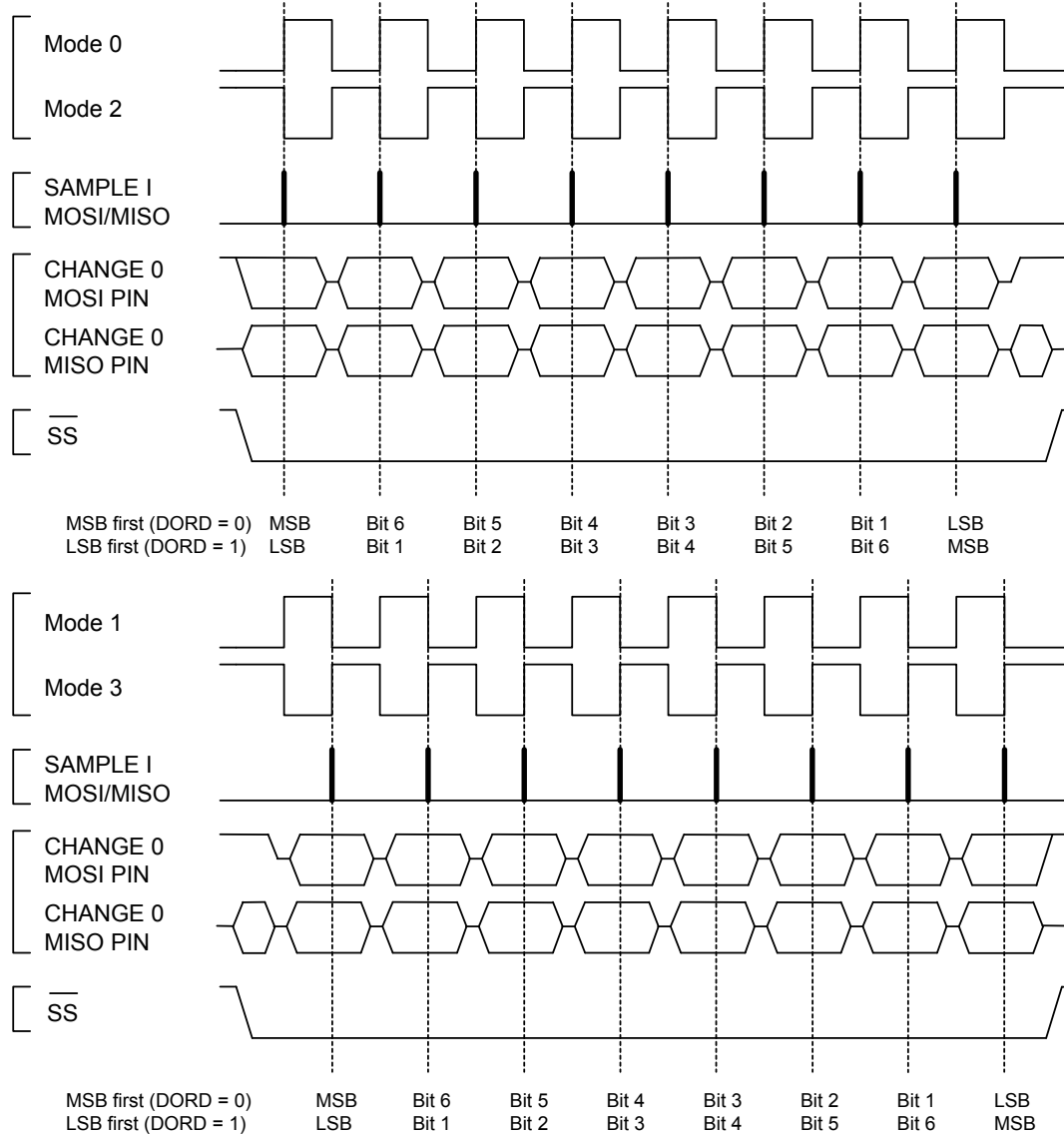
Mode	CPOL	CPHA	Leading Edge	Trailing Edge
0	0	0	Rising, sample	Falling, setup
1	0	1	Rising, setup	Falling, sample
2	1	0	Falling, sample	Rising, setup
3	1	1	Falling, setup	Rising, sample

### Note:

Leading edge is the first clock edge in a clock cycle.

Trailing edge is the second clock edge in a clock cycle.

Figure 29-3. SPI Transfer Modes



29.6.2.6 Transferring Data

**Master**

In master mode (CTRLA.MODE=0x3), when Master Slave Enable Select (CTRLB.MSSEN) is '1', hardware will control the  $\overline{SS}$  line.

When Master Slave Select Enable (CTRLB.MSSEN) is '0', the  $\overline{SS}$  line must be configured as an output.  $\overline{SS}$  can be assigned to any general purpose I/O pin. When the SPI is ready for a data transaction, software must pull the  $\overline{SS}$  line low.

When writing a character to the Data register (DATA), the character will be transferred to the shift register. Once the content of TxDATA has been transferred to the shift register, the Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) will be set. And a new character can be written to DATA.

Each time one character is shifted out from the master, another character will be shifted in from the slave simultaneously. If the receiver is enabled (CTRLA.RXEN=1), 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) will be set. When the transaction is finished, the master must pull the  $\overline{SS}$  line high to notify the slave. If Master Slave Select Enable (CTRLB.MSSEN) is set to '0', the software must pull the  $\overline{SS}$  line high.

#### Slave

In slave mode (CTRLA.MODE=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 content of TxDATA has been loaded into the shift register, INTFLAG.DRE will be set, and new data can be written to DATA.

Similar to the master, the slave will receive one character for each character transmitted. A character will be transferred into the two-level receive buffer within the same clock cycle its last data bit is received. The received character can be retrieved from DATA when the 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) will be set.

After DATA is written it takes up to three SCK clock cycles until the content of DATA is ready to be loaded into the shift register 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](#).

When transmitting several characters in one SPI transaction, the data has to be written into DATA register with at least three SCK clock cycles left in the current character transmission. If this criteria is not met, the previously received character will be transmitted.

Once the DATA register is empty, it takes three CLK\_SERCOM\_APB cycles for INTFLAG.DRE to be set.

#### 29.6.2.7 Receiver Error Bit

The SPI receiver has one error bit: the Buffer Overflow bit (BUFOVF), which can be read from the Status register (STATUS). Once an error happens, the bit will stay set until it is cleared by writing '1' to it. The bit is also automatically cleared when the receiver is disabled.

There are two methods for buffer overflow notification, selected by the immediate buffer overflow notification bit in the Control A register (CTRLA.IBON):

If CTRLA.IBON=1, STATUS.BUFOVF is raised immediately upon buffer overflow. Software can then empty the receive FIFO by reading RxDATA until the receiver complete interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) goes low.

If CTRLA.IBON=0, 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.



**29.6.3 Additional Features**

**29.6.3.1 Address Recognition**

When the SPI is configured for slave operation (CTRLA.MODE=0x2) with address recognition (CTRLA.FORM is 0x2), the SERCOM address recognition logic is enabled: the first character in a transaction is checked for an address match.

If there is a match, 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 the device is in sleep mode, an address match can wake up the device in order to process the transaction.

If there is no match, 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).

Preload must be disabled (CTRLB.PLOADEN=0) in order to use this mode.

**Related Links**

[Address Match and Mask](#)

**29.6.3.2 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.

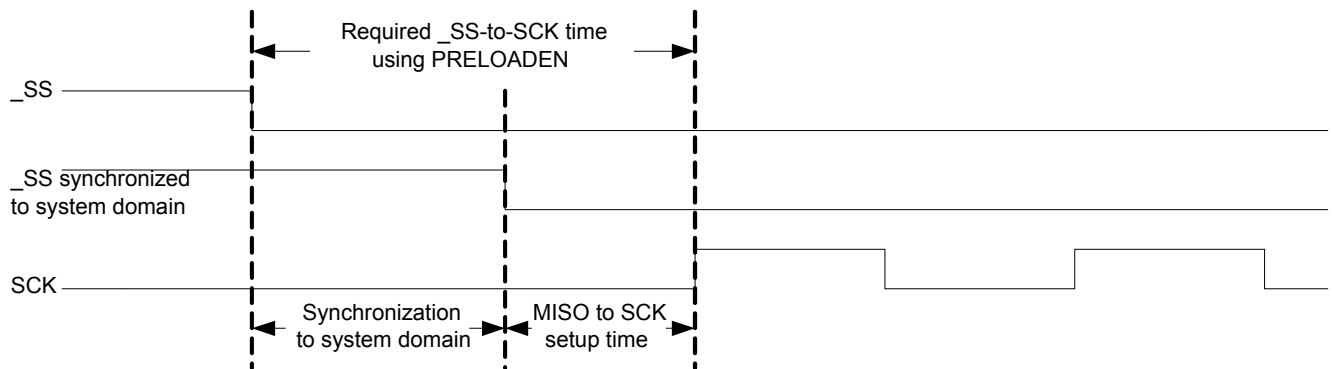
Preloading can be used to preload data into the shift register while  $\overline{SS}$  is high: this eliminates sending a dummy character when starting a transaction. If the shift register is not preloaded, the current contents of the shift register will be shifted out.

Only one data character will be preloaded into the shift register while the synchronized  $\overline{SS}$  signal is high. If the next character is written to DATA before  $\overline{SS}$  is pulled low, the second character will be stored in DATA until transfer begins.

For proper preloading, sufficient time must elapse between  $\overline{SS}$  going low and the first SCK sampling edge, as in [Timing Using Preloading](#). See also *Electrical Characteristics* for timing details.

Preloading is enabled by writing '1' to the Slave Data Preload Enable bit in the CTRLB register (CTRLB.PLOADEN).

**Figure 29-4. Timing Using Preloading**



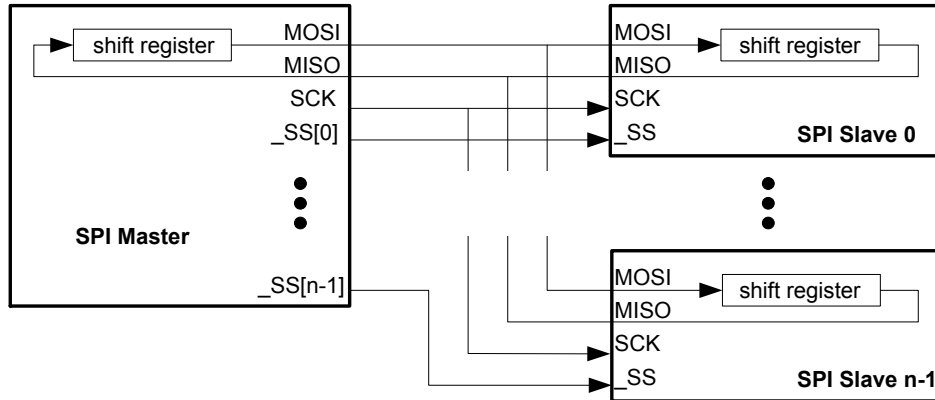
**Related Links**

[Electrical Characteristics](#)

29.6.3.3 Master with Several Slaves

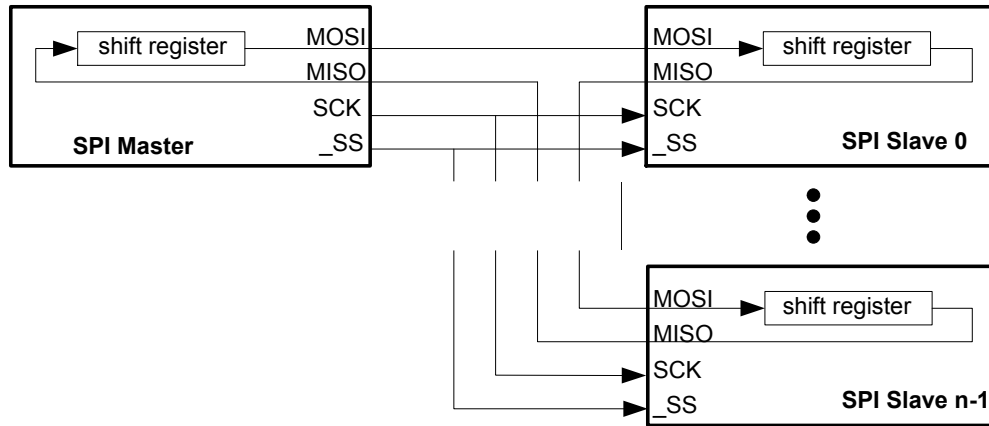
Master with multiple slaves in parallel is only available when Master Slave Select Enable (CTRLB.MSSEN) is set to zero and hardware  $\overline{SS}$  control is disabled. If the bus consists of several SPI slaves, an SPI master can use general purpose I/O pins to control the  $\overline{SS}$  line to each of the slaves on the bus, as shown in [Multiple Slaves in Parallel](#). In this configuration, the single selected SPI slave will drive the tri-state MISO line.

Figure 29-5. Multiple Slaves in Parallel



Another configuration is multiple slaves in series, as in [Multiple Slaves in Series](#). 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), the  $\overline{SS}$  line can be controlled either by hardware or user software and normal GPIO.

Figure 29-6. Multiple Slaves in Series



29.6.3.4 Loop-Back Mode

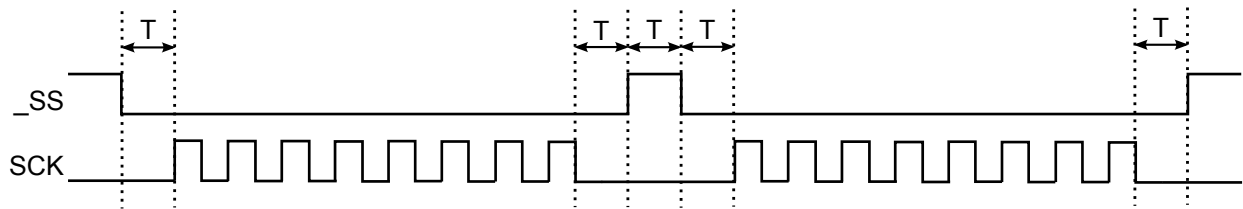
For loop-back mode, configure the Data In Pinout (CTRLA.DIPO) and Data Out Pinout (CTRLA.DOPO) to use the same data pins for transmit and receive. The loop-back is through the pad, so the signal is also available externally.

29.6.3.5 Hardware Controlled  $\overline{SS}$

In master mode, a single  $\overline{SS}$  chip select can be controlled by hardware by writing the Master Slave Select Enable (CTRLB.MSSEN) bit to '1'. 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 [Hardware Controlled  \$\overline{SS}\$](#) , the time T is between one and two baud cycles depending on the SPI transfer mode.

**Figure 29-7. Hardware Controlled  $\overline{SS}$**



T = 1 to 2 baud cycles

When CTRLB.MSSEN=0, the  $\overline{SS}$  pin(s) is/are controlled by user software and normal GPIO.

### 29.6.3.6 Slave Select Low Detection

In slave mode, the SPI can wake 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 up if applicable.

## 29.6.4 DMA, Interrupts, and Events

### 29.6.4.1 DMA Operation

The SPI generates the following DMA requests:

### 29.6.4.2 Interrupts

The SPI has the following interrupt sources. These are asynchronous interrupts, and can wake up the device from any sleep mode:

- Data Register Empty (DRE)
- Receive Complete (RXC)
- Transmit Complete (TXC)
- Slave Select Low (SSL)
- Error (ERROR)

Each interrupt source has its own interrupt flag. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the interrupt condition is met. Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' to the corresponding bit in the Interrupt Enable Clear register (INTENCLR).

An interrupt request is generated when the interrupt flag is set and if the corresponding interrupt is enabled. The interrupt request remains active until either the interrupt flag is cleared, the interrupt is disabled, or the SPI is reset. For details on clearing interrupt flags, refer to the INTFLAG register description.

The value of INTFLAG indicates which interrupt is executed. Note that interrupts must be globally enabled for interrupt requests. Refer to *Nested Vector Interrupt Controller* for details.

### Related Links

[Nested Vector Interrupt Controller](#)

### 29.6.4.3 Events

Not applicable.

### 29.6.5 Sleep Mode Operation

The behavior in sleep mode is depending on the master/slave configuration and the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY):

- Master operation, CTRLA.RUNSTDBY=1: The peripheral clock GCLK\_SERCOM\_CORE will continue to run in idle sleep mode and in standby sleep mode. Any interrupt can wake up the device.
- Master operation, CTRLA.RUNSTDBY=0: GLK\_SERCOMx\_CORE will be disabled after the ongoing transaction is finished. Any interrupt can wake up the device.
- Slave operation, CTRLA.RUNSTDBY=1: The Receive Complete interrupt can wake up the device.
- Slave operation, CTRLA.RUNSTDBY=0: All reception will be dropped, including the ongoing transaction.

### 29.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in the CTRLA register (CTRLA.SWRST)
- Enable bit in the CTRLA register (CTRLA.ENABLE)
- Receiver Enable bit in the CTRLB register (CTRLB.RXEN)

**Note:** CTRLB.RXEN is write-synchronized somewhat differently. See also *CTRLB* register for details.

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

#### Related Links

[Register Synchronization](#)

# ATSAMHAXEXXA

## SERCOM SPI – SERCOM Serial Peripheral Interface

### 29.7 Register Summary

Offset	Name	Bit Pos.							
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]		ENABLE	SWRST
		15:8							IBON
		23:16			DIPO[1:0]				DOPO[1:0]
		31:24		DORD	CPOL	CPHA	FORM[3:0]		
0x04	CTRLB	7:0		PLOADEN				CHSIZE[2:0]	
		15:8	AMODE[1:0]		MSEN			SSDE	
		23:16						RXEN	
		31:24							
0x08 ... 0x0B	Reserved								
0x0C	BAUD	7:0	BAUD[7:0]						
0x0D ... 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		
		15:8							
0x1C	SYNCBUSY	7:0					CTRLB	ENABLE	SWRST
		15:8							
		23:16							
		31:24							
0x20 ... 0x23	Reserved								
0x24	ADDR	7:0	ADDR[7:0]						
		15:8							
		23:16	ADDRMASK[7:0]						
		31:24							
0x28	DATA	7:0	DATA[7:0]						
		15:8							DATA[8:8]
0x2A ... 0x2F	Reserved								
0x30	DBGCTRL	7:0							DBGSTOP

## 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 require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Refer to [Synchronization](#)

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

Refer to [Register Access Protection](#).

### 29.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
		DORD	CPOL	CPHA	FORM[3:0]			
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
			DIPO[1:0]				DOPO[1:0]	
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0
Bit	15	14	13	12	11	10	9	8
								IBON
Access								R/W
Reset								0
Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

#### Bit 30 – DORD Data Order

This bit selects the data order when a character is shifted out from the shift register.

This bit is not synchronized.

Value	Description
0	MSB is transferred first.
1	LSB is transferred first.

#### Bit 29 – CPOL Clock Polarity

In combination with the Clock Phase bit (CPHA), this bit determines the SPI transfer mode.

This bit is not synchronized.

Value	Description
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.

#### Bit 28 – CPHA Clock Phase

In combination with the Clock Polarity bit (CPOL), this bit determines the SPI transfer mode.

# ATSAMHAXEXXA

## SERCOM SPI – SERCOM Serial Peripheral Interface

This bit is not synchronized.

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

Value	Description
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.

### Bits 27:24 – FORM[3:0] Frame Format

This bit field selects the various frame formats supported by the SPI in slave mode. When the 'SPI frame with address' format is selected, the first byte received is checked against the ADDR register.

FORM[3:0]	Name	Description
0x0	SPI	SPI frame
0x1	-	Reserved
0x2	SPI_ADDR	SPI frame with address
0x3-0xF	-	Reserved

### 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.

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 17:16 – DOPO[1:0] 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 ( $\overline{SS}$ ) is controlled by DOPO, while in master operation the  $\overline{SS}$  line is controlled by the port configuration.

In master operation, DO is MOSI.

In slave operation, DO is MISO.

These bits are not synchronized.



DOPO	DO	SCK	Slave $\overline{SS}$	Master $\overline{SS}$
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

### Bit 8 – IBON Immediate Buffer Overflow Notification

This bit controls when the buffer overflow status bit (STATUS.BUFOVF) is set when a buffer overflow occurs.

This bit is not synchronized.

Value	Description
0	STATUS.BUFOVF is set when it occurs in the data stream.
1	STATUS.BUFOVF is set immediately upon buffer overflow.

### Bit 7 – RUNSTDBY Run In Standby

This bit defines the functionality in standby sleep mode.

These bits are not synchronized.

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	Ongoing transaction continues, wake on Receive Complete interrupt.	Generic clock is enabled while in sleep modes. All interrupts can wake up the device.

### Bits 4:2 – MODE[2:0] 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

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.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled or being enabled.

**Bit 0 – SWRST** Software Reset

Writing '0' to this bit has no effect.

Writing '1' to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.

Writing "1" 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.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

### 29.8.2 Control B

**Name:** CTRLB  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

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

#### Bit 17 – RXEN Receiver Enable

Writing '0' 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 '1' 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 '1'.

Writing '1' 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 '1'.

This bit is not enable-protected.

Value	Description
0	The receiver is disabled or being enabled.
1	The receiver is enabled or it will be enabled when SPI is enabled.

#### Bits 15:14 – AMODE[1:0] 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.

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 ( $\overline{SS}$ ) control.

Value	Description
0	Hardware $\overline{SS}$ control is disabled.
1	Hardware $\overline{SS}$ control is enabled.

### Bit 9 – SSDE Slave Select Low Detect Enable

This bit enables wake up when the slave select ( $\overline{SS}$ ) pin transitions from high to low.

Value	Description
0	$\overline{SS}$ low detector is disabled.
1	$\overline{SS}$ low detector is enabled.

### 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  $\overline{SS}$  line is high when DATA is written, it will be transferred immediately to the shift register.

### Bits 2:0 – CHSIZE[2:0] Character Size

CHSIZE[2:0]	Name	Description
0x0	8BIT	8 bits
0x1	9BIT	9 bits
0x2-0x7	-	Reserved

### 29.8.3 Baud Rate

**Name:** BAUD  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection, 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[7:0] Baud Register

These bits control the clock generation, as described in the *SERCOM Clock Generation – Baud-Rate Generator*.

#### Related Links

[Clock Generation – Baud-Rate Generator](#)

[Asynchronous Arithmetic Mode BAUD Value Selection](#)

29.8.4 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

Bit	7	6	5	4	3	2	1	0
	ERROR				SSL	RXC	TXC	DRE
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

**Bit 7 – ERROR** Error Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

**Bit 3 – SSL** Slave Select Low Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Slave Select Low Interrupt Enable bit, which disables the Slave Select Low interrupt.

Value	Description
0	Slave Select Low interrupt is disabled.
1	Slave Select Low interrupt is enabled.

**Bit 2 – RXC** Receive Complete Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Receive Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

Value	Description
0	Receive Complete interrupt is disabled.
1	Receive Complete interrupt is enabled.

**Bit 1 – TXC** Transmit Complete Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Transmit Complete Interrupt Enable bit, which disable the Transmit Complete interrupt.

Value	Description
0	Transmit Complete interrupt is disabled.
1	Transmit Complete interrupt is enabled.

**Bit 0 – DRE** Data Register Empty Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Data Register Empty Interrupt Enable bit, which disables the Data Register Empty interrupt.

Value	Description
0	Data Register Empty interrupt is disabled.
1	Data Register Empty interrupt is enabled.

29.8.5 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x16  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

Bit	7	6	5	4	3	2	1	0
	ERROR				SSL	RXC	TXC	DRE
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

**Bit 7 – ERROR** Error Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

**Bit 3 – SSL** Slave Select Low Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Slave Select Low Interrupt Enable bit, which enables the Slave Select Low interrupt.

Value	Description
0	Slave Select Low interrupt is disabled.
1	Slave Select Low interrupt is enabled.

**Bit 2 – RXC** Receive Complete Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Receive Complete Interrupt Enable bit, which enables the Receive Complete interrupt.

Value	Description
0	Receive Complete interrupt is disabled.
1	Receive Complete interrupt is enabled.

**Bit 1 – TXC** Transmit Complete Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Transmit Complete Interrupt Enable bit, which enables the Transmit Complete interrupt.

Value	Description
0	Transmit Complete interrupt is disabled.
1	Transmit Complete interrupt is enabled.



**Bit 0 – DRE** Data Register Empty Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Data Register Empty Interrupt Enable bit, which enables the Data Register Empty interrupt.

Value	Description
0	Data Register Empty interrupt is disabled.
1	Data Register Empty interrupt is enabled.

29.8.6 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/W	R	R/W	R
Reset	0				0	0	0	0

**Bit 7 – ERROR** Error

This flag is cleared by writing '1' 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 '0' to this bit has no effect.

Writing '1' to this bit will clear the flag.

**Bit 3 – SSL** Slave Select Low

This flag is cleared by writing '1' 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 '0' to this bit has no effect.

Writing '1' 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 '0' to this bit has no effect.

Writing '1' to this bit has no effect.

**Bit 1 – TXC** Transmit Complete

This flag is cleared by writing '1' 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 '0' to this bit has no effect.

Writing '1' to this bit will clear the flag.

**Bit 0 – DRE** Data Register Empty

This flag is cleared by writing new data to DATA.

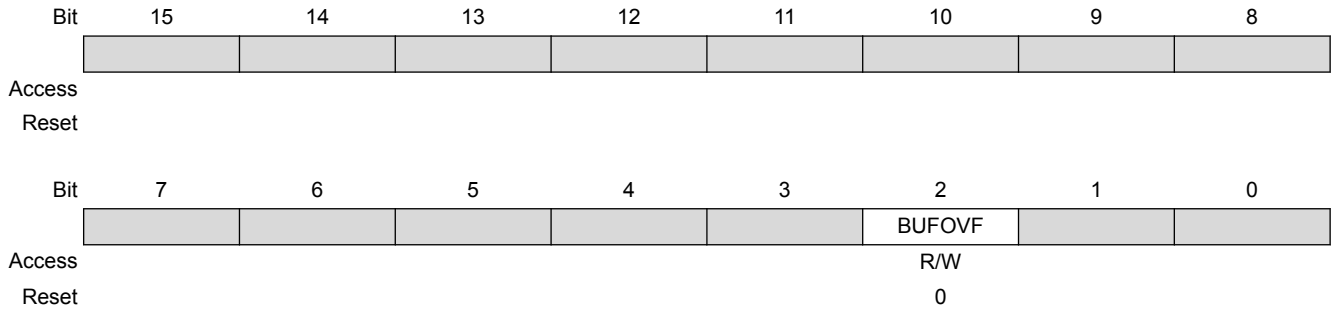
This flag is set when DATA is empty and ready for new data to transmit.

Writing '0' to this bit has no effect.

Writing '1' to this bit has no effect.

### 29.8.7 Status

**Name:** STATUS  
**Offset:** 0x1A  
**Reset:** 0x0000  
**Property:** –



#### 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 '1' to the bit or by disabling the receiver.

This bit is set when a buffer overflow condition is detected. See also [CTRLA.IBON](#) for overflow handling.

When set, the corresponding RxDATA will be zero.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

Value	Description
0	No Buffer Overflow has occurred.
1	A Buffer Overflow has occurred.

### 29.8.8 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x1C  
**Reset:** 0x00000000  
**Property:** -

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

#### Bit 2 – CTRLB CTRLB Synchronization Busy

Writing to the CTRLB when the SERCOM is enabled requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.CTRLB=1 until synchronization is complete. If CTRLB is written while SYNCBUSY.CTRLB=1, an APB error will be generated.

Value	Description
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. Ongoing synchronization is indicated by SYNCBUSY.ENABLE=1 until synchronization is complete.

Value	Description
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. Ongoing synchronization is indicated by SYNCBUSY.SWRST=1 until synchronization is complete.

---

---

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

### 29.8.9 Address

**Name:** ADDR  
**Offset:** 0x24  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
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								
Reset								
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 23:16 – ADDRMASK[7:0] Address Mask

These bits hold the address mask when the transaction format with address is used (CTRLA.FORM, CTRLB.AMODE).

#### Bits 7:0 – ADDR[7:0] Address

These bits hold the address when the transaction format with address is used (CTRLA.FORM, CTRLB.AMODE).

### 29.8.10 Data

**Name:** DATA  
**Offset:** 0x28  
**Reset:** 0x0000  
**Property:** –

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



### 29.8.11 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x30  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### Bit 0 – DBGSTOP Debug Stop Mode

This bit controls the functionality when the CPU is halted by an external debugger.

Value	Description
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.

## 30. SERCOM I<sup>2</sup>C – Inter-Integrated Circuit

### 30.1 Overview

The inter-integrated circuit ( I<sup>2</sup>C) interface is one of the available modes in the serial communication interface (SERCOM).

The I<sup>2</sup>C interface uses the SERCOM transmitter and receiver configured as shown in [Figure 30-1](#). Labels in capital letters are registers accessible by the CPU, while lowercase labels are internal to the SERCOM.

A SERCOM instance can be configured to be either an I<sup>2</sup>C master or an I<sup>2</sup>C slave. Both master and slave have an interface containing a shift register, a transmit buffer and a receive buffer. In addition, the I<sup>2</sup>C master uses the SERCOM baud-rate generator, while the I<sup>2</sup>C slave uses the SERCOM address match logic.

#### Related Links

[SERCOM – Serial Communication Interface](#)

### 30.2 Features

SERCOM I<sup>2</sup>C includes the following features:

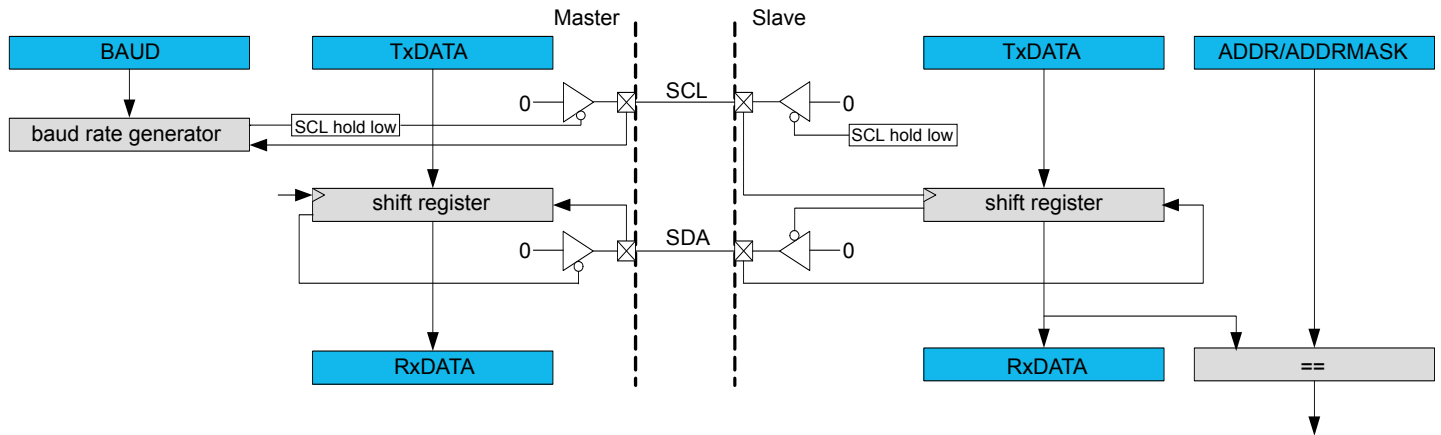
- Master or slave operation
- Can be used with DMA
- Philips I<sup>2</sup>C compatible
- SMBus™ compatible
- PMBus compatible
- Support of 100kHz and 400kHz, 1MHz and 3.4MHz I<sup>2</sup>C mode
- 4-Wire operation supported
- Physical interface includes:
  - Slew-rate limited outputs
  - Filtered inputs
- 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

#### Related Links

[Features](#)

### 30.3 Block Diagram

Figure 30-1. I<sup>2</sup>C Single-Master Single-Slave Interconnection



### 30.4 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 operation)
PAD[3]	Digital I/O	SCL_OUT (4-wire operation)

One signal can be mapped on several pins.

Not all the pins are I<sup>2</sup>C pins.

**Related Links**

- [I/O Multiplexing and Considerations](#)
- [4-Wire Mode](#)

### 30.5 Product Dependencies

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

#### 30.5.1 I/O Lines

In order to use the I/O lines of this peripheral, the I/O pins must be configured using the I/O Pin Controller (PORT).

When the SERCOM is used in I<sup>2</sup>C mode, the SERCOM controls the direction and value of the I/O pins. Both PORT control bits PINCFGn.PULLEN and PINCFGn.DRVSTR are still effective. If the receiver or transmitter is disabled, these pins can be used for other purposes.

**Related Links**

- [PORT: IO Pin Controller](#)

### 30.5.2 Power Management

This peripheral can continue to operate in any sleep mode where its source clock is running. The interrupts can wake up the device from sleep modes.

#### Related Links

[PM – Power Manager](#)

### 30.5.3 Clocks

The SERCOM bus clock (CLK\_SERCOMx\_APB) can be enabled and disabled in the Power Manager. Refer to *Peripheral Clock Masking* for details and default status of this clock.

Two generic clocks are used by SERCOM, GCLK\_SERCOMx\_CORE and GCLK\_SERCOM\_SLOW. The core clock (GCLK\_SERCOMx\_CORE) can clock the I<sup>2</sup>C when working as a master. The slow clock (GCLK\_SERCOM\_SLOW) is required only for certain functions, e.g. SMBus timing. These two clocks must be configured and enabled in the Generic Clock Controller (GCLK) before using the I<sup>2</sup>C.

These generic clocks are 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](#) for further details.

#### Related Links

[GCLK - Generic Clock Controller](#)

[PM – Power Manager](#)

[Peripheral Clock Masking](#)

### 30.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). In order to use DMA requests with this peripheral the DMAC must be configured first. Refer to *DMAC – Direct Memory Access Controller* for details.

#### Related Links

[DMAC – Direct Memory Access Controller](#)

### 30.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. Refer to *Nested Vector Interrupt Controller* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 30.5.6 Events

Not applicable.

### 30.5.7 Debug Operation

When the CPU is halted in debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

### 30.5.8 Register Access Protection

Registers with write-access can be write-protected optionally by the peripheral access controller (PAC).

PAC Write-Protection is not available for the following registers:

- Interrupt Flag Clear and Status register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)
- Address register (ADDR)

Optional PAC Write-Protection is denoted by the "PAC Write-Protection" property in each individual register description.

Write-protection does not apply to accesses through an external debugger.

**Related Links**

[PAC - Peripheral Access Controller](#)

### **30.5.9 Analog Connections**

Not applicable.

## **30.6 Functional Description**

### **30.6.1 Principle of Operation**

The I<sup>2</sup>C interface uses two physical lines for communication:

- Serial Data Line (SDA) for data transfer
- Serial Clock Line (SCL) for the bus clock

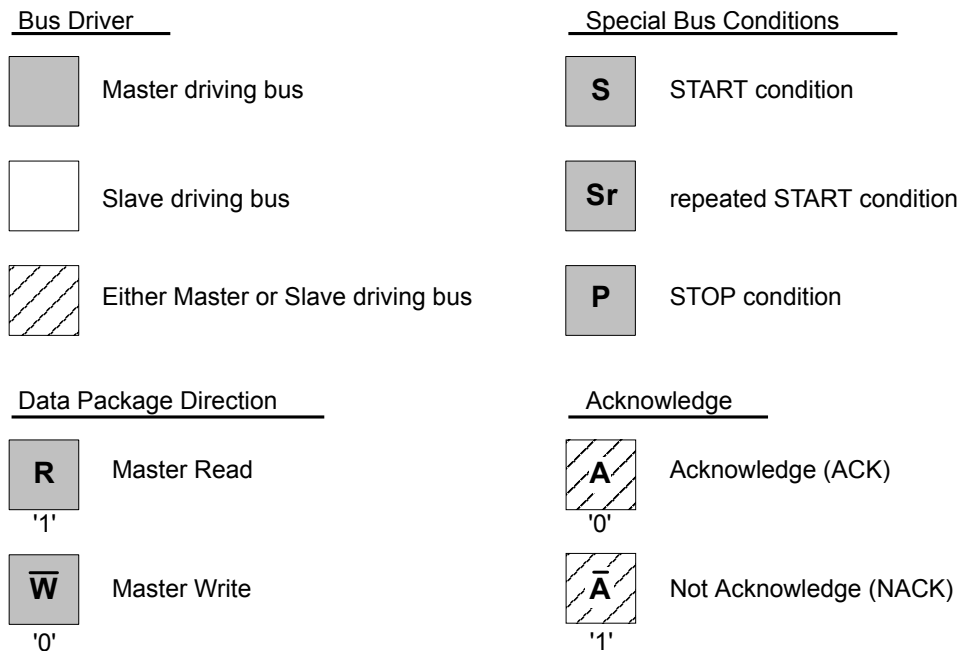
A transaction starts with the I<sup>2</sup>C master sending the start condition, followed by a 7-bit address and a direction bit (read or write to/from the slave).

The addressed I<sup>2</sup>C slave will then acknowledge (ACK) the address, and data packet transactions can begin. Every 9-bit data packet consists of 8 data bits followed by a one-bit reply indicating whether the data was acknowledged or not.

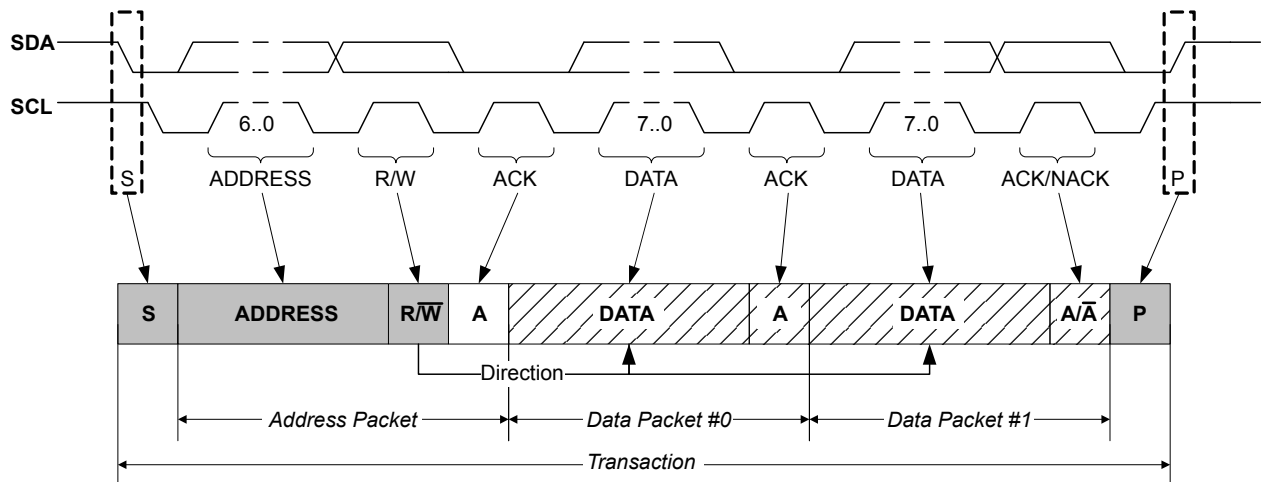
If a data packet is not acknowledged (NACK), whether by the I<sup>2</sup>C slave or master, the I<sup>2</sup>C master takes action by either terminating the transaction by sending the stop condition, or by sending a repeated start to transfer more data.

The figure below illustrates the possible transaction formats and [Transaction Diagram Symbols](#) explains the transaction symbols. These symbols will be used in the following descriptions.

**Figure 30-2. Transaction Diagram Symbols**



**Figure 30-3. Basic I<sup>2</sup>C Transaction Diagram**



## 30.6.2 Basic Operation

### 30.6.2.1 Initialization

The following registers are enable-protected, meaning they can be written only when the I<sup>2</sup>C interface is disabled (CTRLA.ENABLE is '0'):

- Control A register (CTRLA), except Enable (CTRLA.ENABLE) and Software Reset (CTRLA.SWRST) bits
- Control B register (CTRLB), except Acknowledge Action (CTRLB.ACKACT) and Command (CTRLB.CMD) bits
- Baud register (BAUD)
- Address register (ADDR) in slave operation.

When the I<sup>2</sup>C is enabled or is being enabled (CTRLA.ENABLE=1), writing to these registers will be discarded. If the I<sup>2</sup>C is being disabled, writing to these registers will be completed after the disabling.

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

Before the I<sup>2</sup>C is enabled it must be configured as outlined by the following steps:

1. Select I<sup>2</sup>C Master or Slave mode by writing 0x4 (Slave mode) or 0x5 (Master mode) to the Operating Mode bits in the CTRLA register (CTRLA.MODE).
2. If desired, select the SDA Hold Time value in the CTRLA register (CTRLA.SDAHOLD).
3. If desired, enable smart operation by setting the Smart Mode Enable bit in the CTRLB register (CTRLB.SMEN).
4. If desired, enable SCL low time-out by setting the SCL Low Time-Out bit in the Control A register (CTRLA.LOWTOUT).
5. In Master mode:
  - 5.1. Select the inactive bus time-out in the Inactive Time-Out bit group in the CTRLA register (CTRLA.INACTOUT).
  - 5.2. Write the Baud Rate register (BAUD) to generate the desired baud rate.

In Slave mode:

- 5.1. Configure the address match configuration by writing the Address Mode value in the CTRLB register (CTRLB.AMODE).
- 5.2. Set the Address and Address Mask value in the Address register (ADDR.ADDR and ADDR.ADDRMASK) according to the address configuration.

#### **30.6.2.2 Enabling, Disabling, and Resetting**

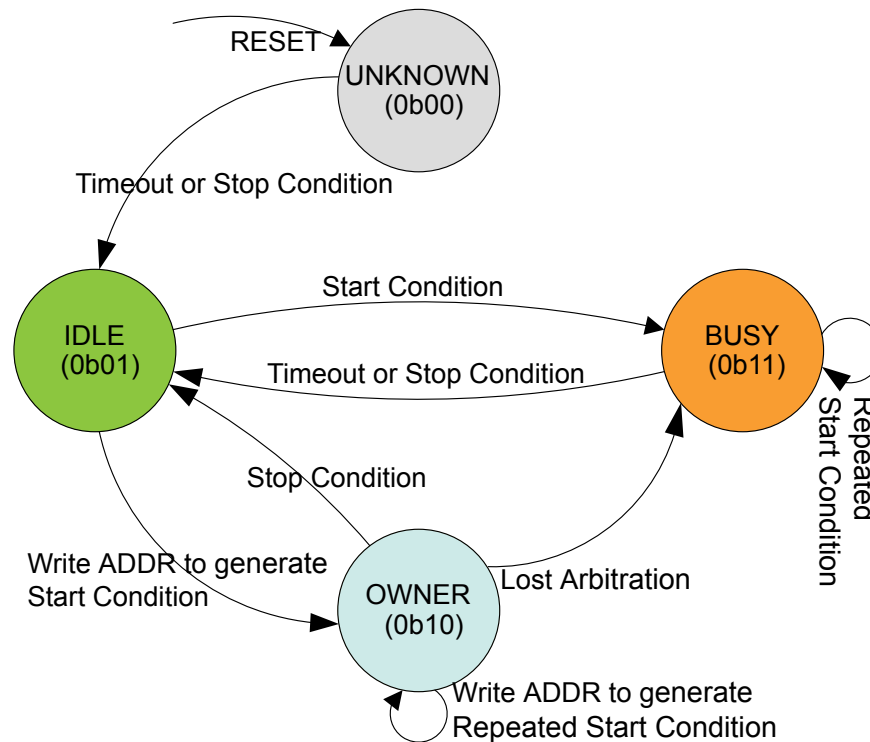
This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST) will reset all registers of this peripheral to their initial states, except the DBGCTRL register, and the peripheral is disabled.

#### **30.6.2.3 I<sup>2</sup>C Bus State Logic**

The bus state logic includes several logic blocks that continuously monitor the activity on the I<sup>2</sup>C bus lines in all sleep modes with running GCLK\_SERCOM\_x clocks. 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 [Bus State Diagram](#). 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 30-4. Bus State Diagram**



The bus state machine is active when the I<sup>2</sup>C master is enabled.

After the I<sup>2</sup>C master has been enabled, the bus state is UNKNOWN (0b00). From the UNKNOWN state, the bus will transition to IDLE (0b01) by either:

- Forcing by writing 0b01 to STATUS.BUSSTATE
- A stop condition is detected on the bus
- If the inactive bus time-out is configured for SMBus compatibility (CTRLA.INACTOUT) and a time-out occurs.

**Note:** Once a known bus state is established, the bus state logic will not re-enter the UNKNOWN state.

When the bus is IDLE it is ready for a new transaction. If a start condition is issued on the bus by another I<sup>2</sup>C master in a multi-master setup, the bus becomes BUSY (0b11). The bus will re-enter IDLE either when a stop condition is detected, or when a time-out occurs (inactive bus time-out needs to be configured).

If a start condition is generated internally by writing the Address bit group in the Address register (ADDR.ADDR) while IDLE, the OWNER state (0b10) is entered. If the complete transaction was performed without interference, i.e., arbitration was not lost, the I<sup>2</sup>C master can issue a stop condition, which will change the bus state back to IDLE.

However, if a packet collision is detected while in 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.

**Note:** Violating the protocol may cause the I<sup>2</sup>C to hang. If this happens it is possible to recover from this state by a software reset (CTRLA.SWRST='1').

**Related Links**

[CTRLA](#)



30.6.2.4 I<sup>2</sup>C Master Operation

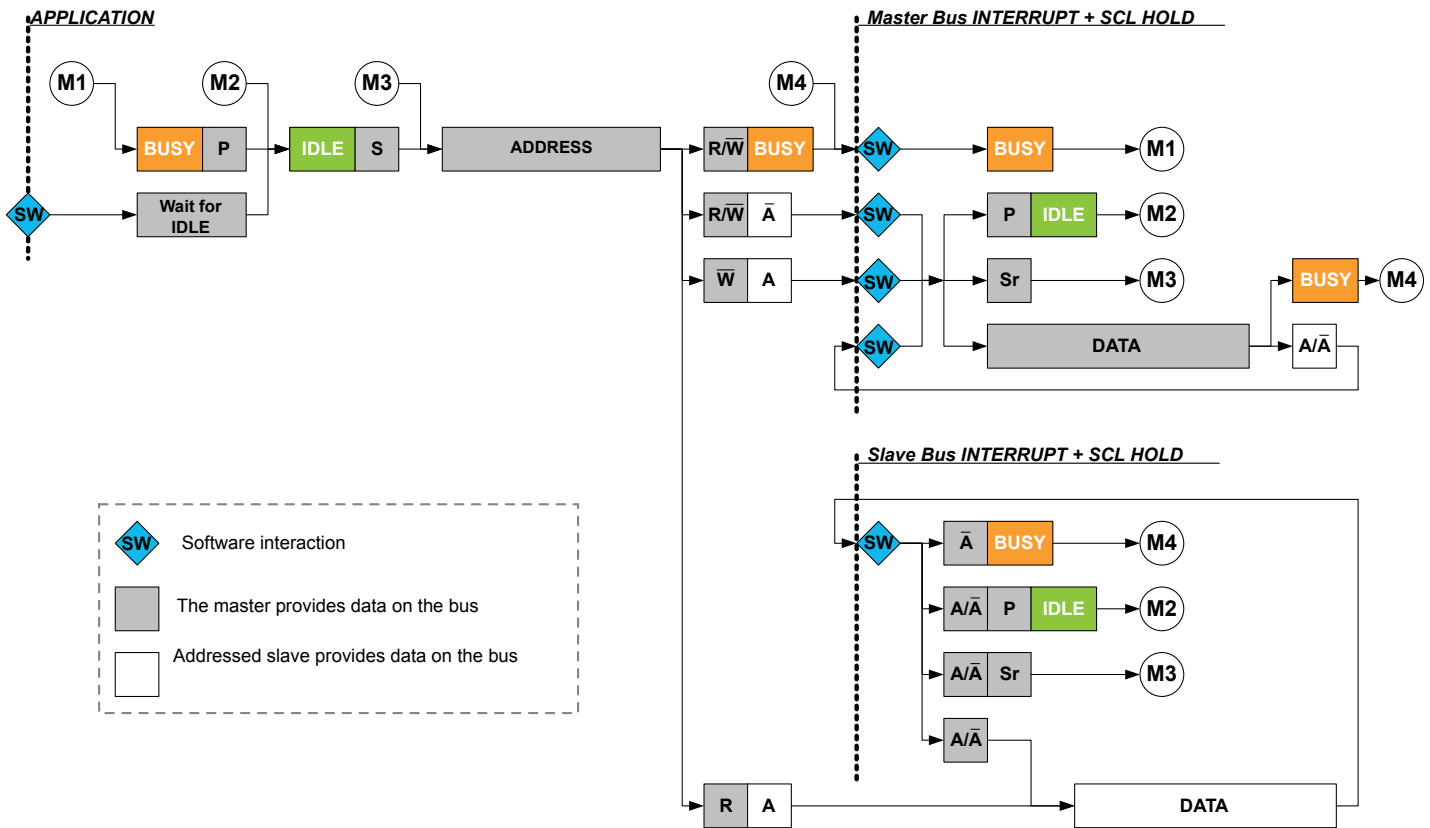
The I<sup>2</sup>C master is byte-oriented and interrupt based. The number of interrupts generated is kept at a minimum by automatic handling of most incidents. The software driver complexity and code size are reduced by auto-triggering of operations, and a special smart mode, which can be enabled by the Smart Mode Enable bit in the Control A register (CTRLA.SMEN).

The I<sup>2</sup>C master has two interrupt strategies.

When SCL Stretch Mode (CTRLA.SCLSM) is '0', SCL is stretched before or after the acknowledge bit. In this mode the I<sup>2</sup>C master operates according to [Master Behavioral Diagram \(SCLSM=0\)](#). The circles labelled "Mn" (M1, M2..) indicate the nodes the bus logic can jump to, based on software or hardware interaction.

This diagram is used as reference for the description of the I<sup>2</sup>C master operation throughout the document.

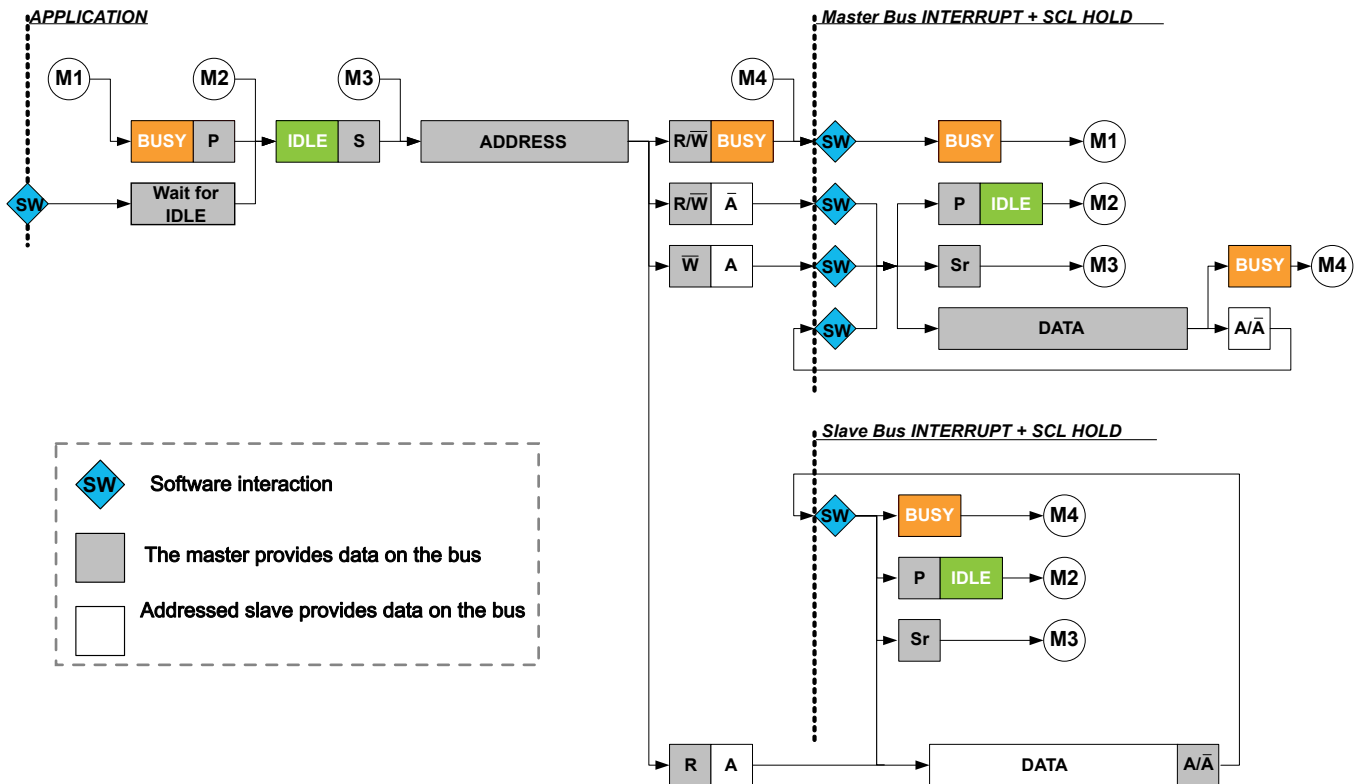
Figure 30-5. I<sup>2</sup>C Master Behavioral Diagram (SCLSM=0)



In the second strategy (CTRLA.SCLSM=1), interrupts only occur after the ACK bit, as in [Master Behavioral Diagram \(SCLSM=1\)](#). This strategy can be used when it is not necessary to check DATA before acknowledging.

**Note:** I<sup>2</sup>C High-speed (*Hs*) mode requires CTRLA.SCLSM=1.

**Figure 30-6. I<sup>2</sup>C Master Behavioral Diagram (SCLSM=1)**



### Master Clock Generation

The SERCOM peripheral supports several I<sup>2</sup>C bidirectional modes:

- Standard mode (*Sm*) up to 100kHz
- Fast mode (*Fm*) up to 400kHz
- Fast mode Plus (*Fm+*) up to 1MHz
- High-speed mode (*Hs*) up to 3.4MHz

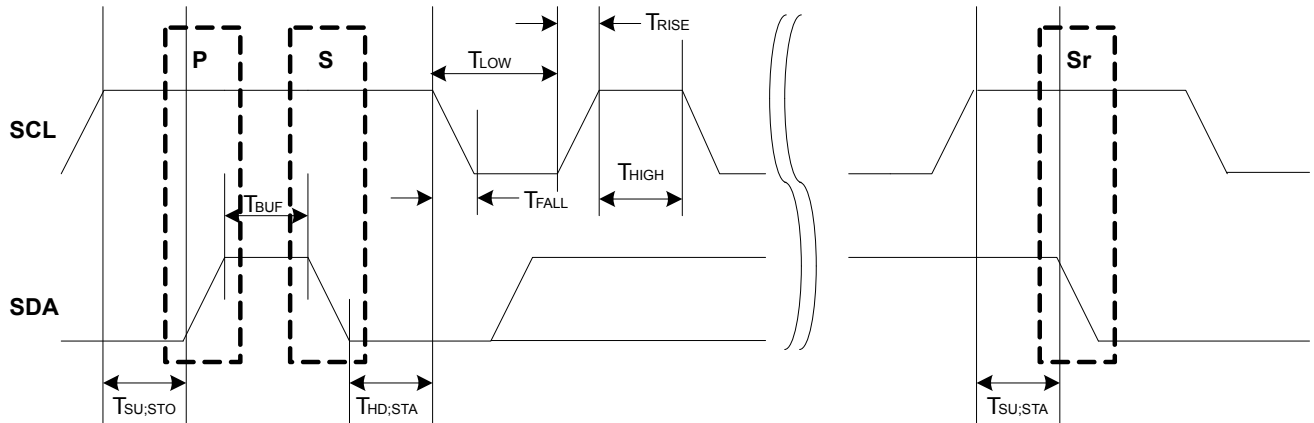
The Master clock configuration for *Sm*, *Fm*, and *Fm+* are described in [Clock Generation \(Standard-Mode, Fast-Mode, and Fast-Mode Plus\)](#). For *Hs*, refer to [Master Clock Generation \(High-Speed Mode\)](#).

### Clock Generation (Standard-Mode, Fast-Mode, and Fast-Mode Plus)

In I<sup>2</sup>C *Sm*, *Fm*, and *Fm+* mode, the Master clock (SCL) frequency is determined as described in this section:

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 30-7. SCL Timing



The following parameters are timed using the SCL low time period  $T_{LOW}$ . This comes from the Master Baud Rate Low bit group in the Baud Rate register (BAUD.BAUDLOW). When BAUD.BAUDLOW=0, or the Master Baud Rate bit group in the Baud Rate register (BAUD.BAUD) determines it.

- $T_{LOW}$  – 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*.
- $T_{FALL}$  is determined by the open-drain current limit and bus impedance; can typically be regarded as zero. Refer to *Electrical Characteristics* 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:

$$f_{SCL} = \frac{f_{GCLK}}{10 + 2BAUD + f_{GCLK} \cdot T_{RISE}}$$

When BAUD.BAUDLOW is non-zero, the following formula determines the SCL frequency:

$$f_{SCL} = \frac{f_{GCLK}}{10 + BAUD + BAUDLOW + f_{GCLK} \cdot T_{RISE}}$$

The following formulas can determine the SCL  $T_{LOW}$  and  $T_{HIGH}$  times:

$$T_{LOW} = \frac{BAUDLOW + 5}{f_{GCLK}}$$

$$T_{HIGH} = \frac{BAUD + 5}{f_{GCLK}}$$

**Note:** The I<sup>2</sup>C standard *Fm+* (Fast-mode plus) requires a nominal high to low SCL ratio of 1:2, and BAUD should be set accordingly. At a minimum, BAUD.BAUD and/or BAUD.BAUDLOW must be non-zero.

**Startup Timing** The minimum time between SDA transition and SCL rising edge is 6 APB cycles when the DATA register is written in smart mode. If a greater startup time is required due to long rise times, the time between DATA write and IF clear must be controlled by software.

**Note:** When timing is controlled by user, the Smart Mode cannot be enabled.

#### Related Links

[Electrical Characteristics](#)

#### **Master Clock Generation (High-Speed Mode)**

For I<sup>2</sup>C *Hs* transfers, there is no SCL synchronization. Instead, the SCL frequency is determined by the GCLK\_SERCOMx\_CORE frequency ( $f_{GCLK}$ ) and the High-Speed Baud setting in the Baud register (BAUD.HSBAUD). When BAUD.HSBAUDLOW=0, the HSBAUD value will determine both SCL high and SCL low. In this case the following formula determines the SCL frequency.

$$f_{SCL} = \frac{f_{GCLK}}{2 + 2 \cdot HS\ BAUD}$$

When HSBAUDLOW is non-zero, the following formula determines the SCL frequency.

$$f_{SCL} = \frac{f_{GCLK}}{2 + HS\ BAUD + HSBAUDLOW}$$

**Note:** The I<sup>2</sup>C standard *Hs* (High-speed) requires a nominal high to low SCL ratio of 1:2, and HSBAUD should be set accordingly. At a minimum, BAUD.HSBAUD and/or BAUD.HSBAUDLOW must be non-zero.

#### **Transmitting Address Packets**

The I<sup>2</sup>C master starts a bus transaction by writing the I<sup>2</sup>C slave address to ADDR.ADDR and the direction bit, as described in [Principle of Operation](#). If the bus is busy, the I<sup>2</sup>C master will wait until the bus becomes idle before continuing the operation. When the bus is idle, the I<sup>2</sup>C master will issue a start condition on the bus. The I<sup>2</sup>C master will then transmit an address packet using the address written to ADDR.ADDR. After the address packet has been transmitted by the I<sup>2</sup>C master, one of four cases will arise according to 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 Status and Clear 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 I<sup>2</sup>C master is no longer allowed to execute 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 moment, 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 there is no I<sup>2</sup>C slave device responding to the address packet, then the INTFLAG.MB interrupt flag and STATUS.RXNACK will be set. The clock hold is active at this point, preventing further activity on the bus.

The missing ACK response can indicate that the I<sup>2</sup>C slave is busy with other tasks or sleeping. Therefore, it is not able to respond. In this event, the next step can be either issuing a stop condition (recommended) or resending the address packet by a repeated start condition. When using SMBus logic, the slave must ACK the address. If there is no response, it means that the slave is not available on the bus.

### **Case 3: Address packet transmit complete – Write packet, Master on Bus set**

If the I<sup>2</sup>C master receives an acknowledge response from the I<sup>2</sup>C slave, INTFLAG.MB will be set and STATUS.RXNACK will be 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 I<sup>2</sup>C operation to continue:

- Initiate a data transmit operation 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 I<sup>2</sup>C master receives an ACK from the I<sup>2</sup>C slave, the I<sup>2</sup>C master proceeds to receive the next byte of data from the I<sup>2</sup>C slave. When the first data byte is received, the Slave on Bus bit in the Interrupt Flag register (INTFLAG.SB) will be set and STATUS.RXNACK will be 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 I<sup>2</sup>C operation to continue:

- Let the I<sup>2</sup>C master continue to read data by acknowledging the data received. ACK can be sent by software, or automatically in smart mode.
- Transmit a new address packet.
- Terminate the transaction by issuing a stop condition.

**Note:** An ACK or NACK will be automatically transmitted 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 Master Write (see [Figure 30-3](#)) was transmitted successfully, INTFLAG.MB will be set. The I<sup>2</sup>C master will start transmitting data via the I<sup>2</sup>C bus by writing to DATA.DATA, and monitor continuously for packet collisions. I

If a collision is detected, the I<sup>2</sup>C master will lose arbitration and STATUS.ARBLOST will be set. If the transmit was successful, the I<sup>2</sup>C master will receive an ACK bit from the I<sup>2</sup>C slave, and STATUS.RXNACK will be cleared. INTFLAG.MB will be set in both cases, regardless of arbitration outcome.

It is recommended to read STATUS.ARBLOST and handle the arbitration lost condition in the beginning of the I<sup>2</sup>C Master on Bus interrupt. 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 I<sup>2</sup>C master is not allowed to continue transmitting data packets if a NACK is received from the I<sup>2</sup>C slave.

**Receiving Data Packets (SCLSM=0)**

When INTFLAG.SB is set, the I<sup>2</sup>C master will already have received one data packet. The I<sup>2</sup>C master must respond by sending either an ACK or NACK. Sending a NACK may be unsuccessful when arbitration is lost during the transmission. In this case, a lost arbitration will prevent setting INTFLAG.SB. Instead, INTFLAG.MB will 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 I<sup>2</sup>C master will already have received one data packet and transmitted an ACK or NACK, depending on CTRLB.ACKACT. At this point, CTRLB.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 the smart mode.

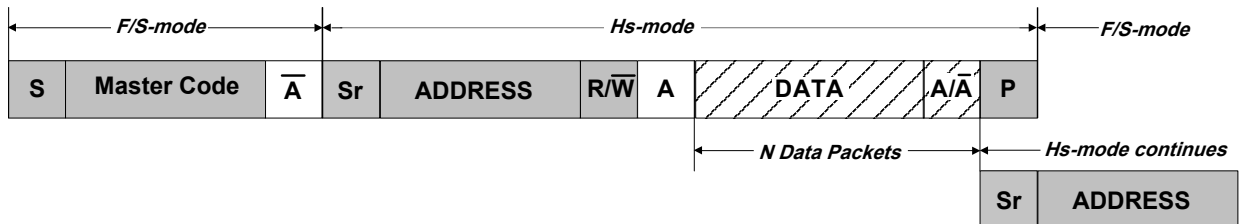
**High-Speed Mode**

High-speed transfers are a multi-step process, see [High Speed Transfer](#).

First, a master code (0b00001nnn, where 'nnn' is a unique master code) is transmitted in Full-speed mode, followed by a NACK since no slaves 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.ADDR) and writing the high-speed bit (ADDR.HS) to '0'.

After the master code and NACK have been transmitted, the master write interrupt will be asserted. In the meanwhile, the slave address can be written to the ADDR.ADDR register together with ADDR.HS=1. Now in High-speed mode, the master will generate a repeated start, followed by the slave address with RW-direction. 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 ADDR.ADDR to be transmitted.

**Figure 30-8. High Speed Transfer**



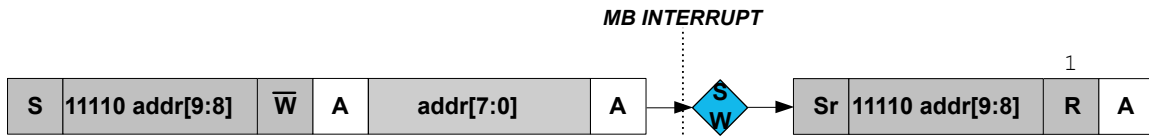
Transmitting in High-speed mode requires the I<sup>2</sup>C master to be configured in High-speed mode (CTRLA.SPEED=0x2) and the SCL clock stretch mode (CTRLA.SCLSM) bit set to '1'.

**10-Bit Addressing**

When 10-bit addressing is enabled by the Ten Bit Addressing Enable bit in the Address register (ADDR.TENBITEN=1) and the Address bit field ADDR.ADDR is written, the two address bytes will be transmitted, see [10-bit Address Transmission for a Read Transaction](#). 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 direction bit (ADDR.ADDR[0]) being zero.

If the master receives a NACK after the first byte, the write interrupt flag will be raised and the STATUS.RXNACK 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 direction is read-from-slave, 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 30-9. 10-bit Address Transmission for a Read Transaction**



This implies the following procedure for a 10-bit read operation:

1. Write the 10-bit address to ADDR.ADDR[10:1]. ADDR.TENBITEN must be '1', the direction bit (ADDR.ADDR[0]) must be '0' (can be written simultaneously with ADDR).
2. Once the Master on Bus 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).
3. Proceed to transmit data.

### 30.6.2.5 I<sup>2</sup>C Slave Operation

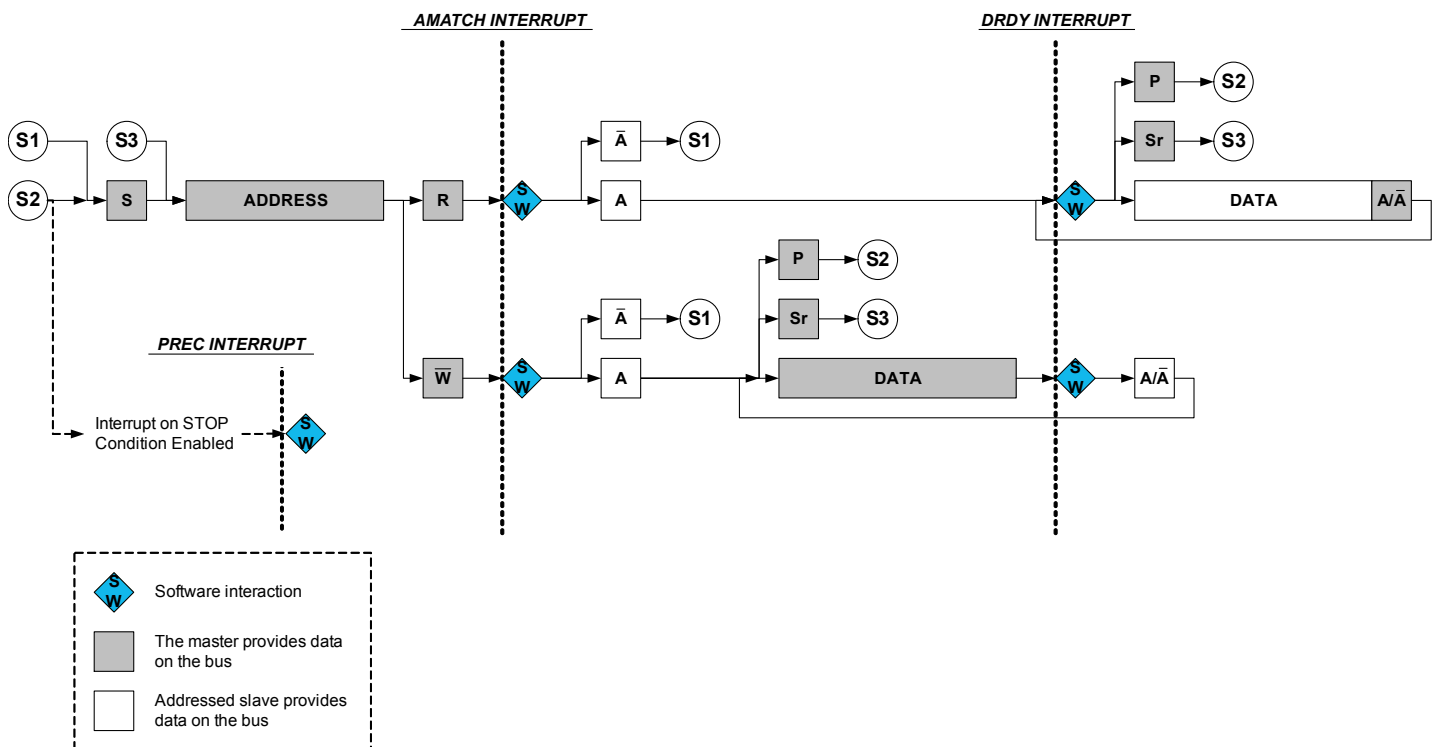
The I<sup>2</sup>C slave is byte-oriented and interrupt-based. The number of interrupts generated is kept at a minimum by automatic handling of most events. The software driver complexity and code size are reduced by auto-triggering of operations, and a special smart mode, which can be enabled by the Smart Mode Enable bit in the Control A register (CTRLA.SMEN).

The I<sup>2</sup>C slave has two interrupt strategies.

When SCL Stretch Mode bit (CTRLA.SCLSM) is '0', SCL is stretched before or after the acknowledge bit. In this mode, the I<sup>2</sup>C slave operates according to [I<sup>2</sup>C Slave Behavioral Diagram \(SCLSM=0\)](#). The circles labelled "Sn" (S1, S2..) indicate the nodes the bus logic can jump to, based on software or hardware interaction.

This diagram is used as reference for the description of the I<sup>2</sup>C slave operation throughout the document.

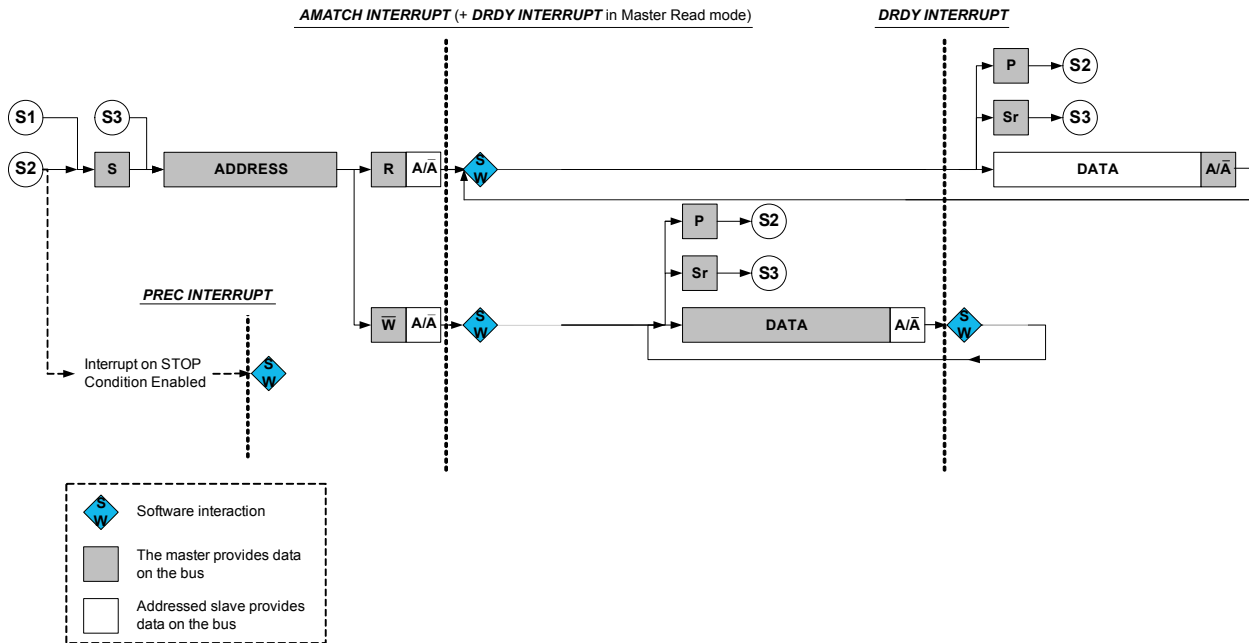
**Figure 30-10. I<sup>2</sup>C Slave Behavioral Diagram (SCLSM=0)**



In the second strategy (CTRLA.SCLSM=1), interrupts only occur after the ACK bit is sent as shown in [Slave Behavioral Diagram \(SCLSM=1\)](#). 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. However, 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:** For I<sup>2</sup>C High-speed mode (*Hs*), SCLSM=1 is required.

**Figure 30-11. I<sup>2</sup>C Slave Behavioral Diagram (SCLSM=1)**



**Receiving Address Packets (SCLSM=0)**

When CTRLA.SCLSM=0, the I<sup>2</sup>C slave stretches the SCL line according to [Figure 30-10](#). When the I<sup>2</sup>C slave is properly configured, it will wait for a start condition.

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 will be rejected, and the I<sup>2</sup>C slave will wait for a new start condition. If the received address is a match, the Address Match bit in the Interrupt Flag register (INTFLAG.AMATCH) will be set.

SCL will be stretched until the I<sup>2</sup>C slave clears INTFLAG.AMATCH. As the I<sup>2</sup>C slave holds the clock by forcing SCL low, the software has unlimited time to respond.

The direction of a transaction is determined by reading the Read / Write Direction bit in the Status register (STATUS.DIR). This 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<sup>2</sup>C slave had a packet collision. A collision causes the SDA and SCL lines to be released without any notification to software. Therefore, the next AMATCH interrupt is 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<sup>2</sup>C master, one of two cases will arise based on transfer direction.



**Case 1: Address packet accepted – Read flag set**

The STATUS.DIR bit is '1', indicating an I<sup>2</sup>C master read operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, I<sup>2</sup>C slave hardware will set the Data Ready bit in the Interrupt Flag register (INTFLAG.DRDY), indicating data are needed for transmit. If a NACK is sent, the I<sup>2</sup>C 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<sup>2</sup>C slave Command bit field in the Control B register (CTRLB.CMD) can be written to '0x3' for both read and write operations as the command execution is dependent on the STATUS.DIR bit. Writing '1' 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<sup>2</sup>C master write operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, the I<sup>2</sup>C slave will wait for data to be received. Data, repeated start or stop can be received.

If a NACK is sent, the I<sup>2</sup>C slave will wait for a new start condition and address match. Typically, software will immediately acknowledge the address packet by sending an ACK/NACK. The I<sup>2</sup>C slave command CTRLB.CMD = 3 can be used for both read and write operation as the command execution is dependent on STATUS.DIR.

Writing '1' 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=1, the I<sup>2</sup>C slave will stretch the SCL line only after an ACK, see [Slave Behavioral Diagram \(SCLSM=1\)](#). When the I<sup>2</sup>C 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 will be rejected and the I<sup>2</sup>C slave will wait for a new start condition.

If the address matches, the acknowledge action as configured by the Acknowledge Action bit Control B register (CTRLB.ACKACT) will be sent and the Address Match bit in the Interrupt Flag register (INTFLAG.AMATCH) is set. SCL will be stretched until the I<sup>2</sup>C slave clears INTFLAG.AMATCH. As the I<sup>2</sup>C 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). This 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, the last packet addressed to the I<sup>2</sup>C 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<sup>2</sup>C master, INTFLAG.AMATCH be set to '1' to clear it.

**Receiving and Transmitting Data Packets**

After the I<sup>2</sup>C 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. After receiving data, the I<sup>2</sup>C slave will send an acknowledge according to CTRLB.ACKACT.

**Case 1: Data received**

INTFLAG.DRDY is set, and SCL is held low, pending for SW interaction.

**Case 2: Data sent**

When a byte transmission is successfully completed, the INTFLAG.DRDY interrupt flag is set. If NACK is received, indicated by STATUS.RXNACK=1, the I<sup>2</sup>C slave must expect a stop or a repeated start to be received. The I<sup>2</sup>C slave must release the data line to allow the I<sup>2</sup>C master to generate a stop or repeated start. Upon detecting a stop condition, the Stop Received bit in the Interrupt Flag register (INTFLAG.PREC) will be set and the I<sup>2</sup>C slave will return to IDLE state.

**High-Speed Mode**

When the I<sup>2</sup>C slave is configured in High-speed mode (*Hs*, CTRLA.SPEED=0x2) and CTRLA.SCLSM=1, 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.

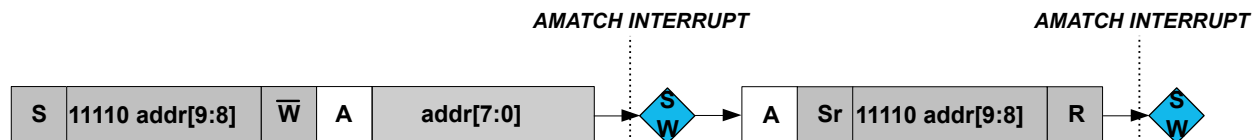
**10-Bit Addressing**

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, see [10-bit Addressing](#).

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 it was addressed by the previous 10-bit address.

**Figure 30-12. 10-bit Addressing**



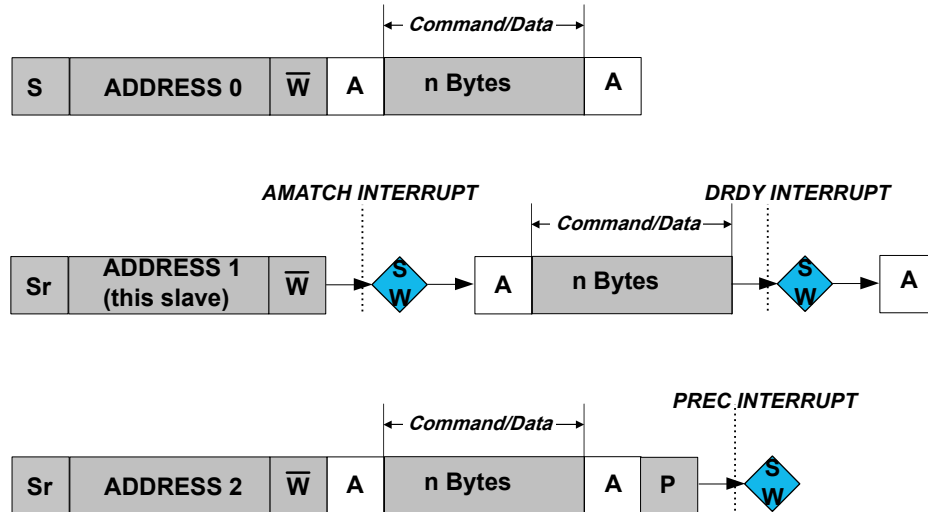
**PMBus Group Command**

When the PMBus Group Command bit in the CTRLB register is set (CTRLB.GCMD=1) and 7-bit addressing is used, INTFLAG.PREC will be set if the slave has been addressed since the last STOP condition. When CTRLB.GCMD=0, a STOP condition without address match will not be set INTFLAG.PREC.

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.

[PMBus Group Command Example](#) shows an example where this slave, bearing ADDRESS 1, is addressed after a repeated START condition. There can be multiple slaves addressed before and after this slave. Eventually, at the end of the group command, a single STOP is generated by the master. At this point a STOP interrupt is asserted.

**Figure 30-13. PMBus Group Command Example**



### 30.6.3 Additional Features

#### 30.6.3.1 SMBus

The I<sup>2</sup>C includes three hardware SCL low time-outs which allow a time-out to occur for SMBus SCL low time-out, master extend time-out, and slave extend time-out. This allows for SMBus functionality. 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 use a 32KHz oscillator. The I<sup>2</sup>C interface also allows for a SMBus compatible SDA hold time.

- $T_{\text{TIMEOUT}}$ : SCL low time of 25..35ms – Measured for a single SCL low period. It is enabled by CTRLA.LOWTOUTEN.
- $T_{\text{LOW:SEXT}}$ : 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. It is enabled by CTRLA.SEXTTOEN.
- $T_{\text{LOW:MEXT}}$ : 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. It is enabled by CTRLA.MEXTTOEN.

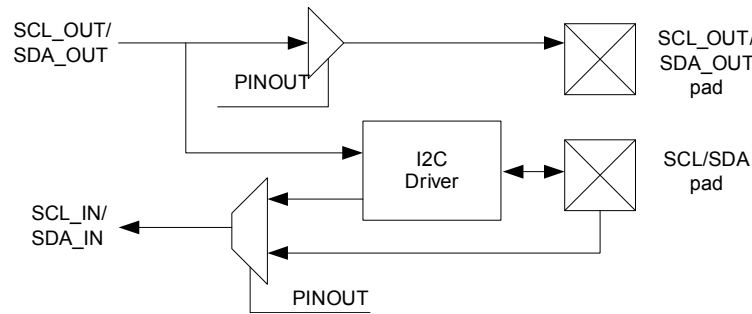
#### 30.6.3.2 Smart Mode

The I<sup>2</sup>C interface has a smart mode that simplifies application code and minimizes the user interaction needed to adhere to the I<sup>2</sup>C protocol. The smart mode accomplishes this by automatically issuing an ACK or NACK (based on the content of CTRLB.ACKACT) as soon as DATA.DATA is read.

#### 30.6.3.3 4-Wire Mode

Writing a '1' to the Pin Usage bit in the Control A register (CTRLA.PINOUT) will enable 4-wire mode operation. In this mode, the internal I<sup>2</sup>C tri-state drivers are bypassed, and an external I<sup>2</sup>C compliant tri-state driver is needed when connecting to an I<sup>2</sup>C bus.

Figure 30-14. I<sup>2</sup>C Pad Interface



30.6.3.4 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 (INTFLAG.SB or INTFLAG.MB) 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.

30.6.4 DMA, Interrupts and Events

Each interrupt source has its own interrupt flag. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the interrupt condition is met. Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' 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 is active until the interrupt flag is cleared, the interrupt is disabled or the I<sup>2</sup>C is reset. See the INTFLAG (Slave) or INTFLAG (Master) register for details on how to clear interrupt flags.

Table 30-1. Module Request for SERCOM I<sup>2</sup>C Slave

Condition	Request		
	DMA	Interrupt	Event
Data needed for transmit (TX) (Slave transmit mode)	Yes (request cleared when data is written)		NA
Data received (RX) (Slave receive mode)	Yes (request cleared when data is read)		
Data Ready (DRDY)		Yes	
Address Match (AMATCH)		Yes	
Stop received (PREC)		Yes	
Error (ERROR)		Yes	

**Table 30-2. Module Request for SERCOM I<sup>2</sup>C Master**

Condition	Request		
	DMA	Interrupt	Event
Data needed for transmit (TX) (Master transmit mode)	Yes (request cleared when data is written)		NA
Data needed for transmit (RX) (Master transmit mode)	Yes (request cleared when data is read)		
Master on Bus (MB)		Yes	
Stop received (SB)		Yes	
Error (ERROR)		Yes	

**30.6.4.1 DMA Operation**

Smart mode must be enabled for DMA operation in the Control B register by writing CTRLB.SMEN=1.

**Slave DMA**

When using the I<sup>2</sup>C 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 I<sup>2</sup>C slave generates the following requests:

**Master DMA**

When using the I<sup>2</sup>C 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<sup>2</sup>C master generates the following requests:

**30.6.4.2 Interrupts**

The I<sup>2</sup>C slave has the following interrupt sources. These are asynchronous interrupts. They can wake-up the device from any sleep mode:

- Error (ERROR)
- Data Ready (DRDY)
- Address Match (AMATCH)
- Stop Received (PREC)

The I<sup>2</sup>C master has the following interrupt sources. These are asynchronous interrupts. They can wake-up the device from any sleep mode:

- Error (ERROR)
- Slave on Bus (SB)
- Master on Bus (MB)

Each interrupt source has its own interrupt flag. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the interrupt condition is met. Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' 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 is active until the interrupt flag is cleared, the interrupt is disabled or the I<sup>2</sup>C is reset. See the INTFLAG register for details on how to clear interrupt flags.

The value of INTFLAG indicates which interrupt is executed. Note that interrupts must be globally enabled for interrupt requests. Refer to *Nested Vector Interrupt Controller* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

#### 30.6.4.3 Events

Not applicable.

#### 30.6.5 Sleep Mode Operation

##### I<sup>2</sup>C 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 '1', the GLK\_SERCOMx\_CORE will also run in standby sleep mode. Any interrupt can wake up the device.

If CTRLA.RUNSTDBY=0, the GLK\_SERCOMx\_CORE will be disabled after any ongoing transaction is finished. Any interrupt can wake up the device.

##### I<sup>2</sup>C Slave Operation

Writing CTRLA.RUNSTDBY=1 will allow the Address Match interrupt to wake up the device.

When CTRLA.RUNSTDBY=0, all receptions will be dropped.

#### 30.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in the CTRLA register (CTRLA.SWRST)
- Enable bit in the CTRLA register (CTRLA.ENABLE)
- Command bits in CTRLB register (CTRLB.CMD)
- Write to Bus State bits in the Status register (STATUS.BUSSTATE)
- Address bits in the Address register (ADDR.ADDR) when in master operation.

The following registers are synchronized when written:

- Data (DATA) when in master operation

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

**Related Links**

[Register Synchronization](#)

### 30.7 Register Summary - I2C Slave

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]		ENABLE	SWRST	
		15:8								
		23:16	SEXTTOEN		SDAHOLD[1:0]				PINOUT	
		31:24		LOWTOUT			SCLSM		SPEED[1:0]	
0x04	CTRLB	7:0								
		15:8	AMODE[1:0]				AACKEN	GCMD	SMEN	
		23:16					ACKACT	CMD[1:0]		
		31:24								
0x08 ... 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
		15:8					LENERR	HS	SEXTTOUT	
0x1C	SYNCBUSY	7:0							ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x20 ... 0x23	Reserved									
0x24	ADDR	7:0	ADDR[6:0]						GENCEN	
		15:8	TENBITEN					ADDR[9:7]		
		23:16	ADDRMASK[6:0]							
		31:24						ADDRMASK[9:7]		
0x28	DATA	7:0	DATA[7:0]							
		15:8								

### 30.8 Register Description - I<sup>2</sup>C Slave

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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#).



Some registers are synchronized when read and/or written. Synchronization is denoted by the "Write-Synchronized" or the "Read-Synchronized" property in each individual register description. For details, refer to [Synchronization](#).

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 30.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

	Bit	31	30	29	28	27	26	25	24	
			LOWTOUT			SCLSM		SPEED[1:0]		
Access			R/W			R/W		R/W	R/W	
Reset			0			0		0	0	
	Bit	23	22	21	20	19	18	17	16	
		SEXTTOEN		SDAHOLD[1:0]						PINOUT
Access		R/W		R/W	R/W				R/W	
Reset		0		0	0				0	
	Bit	15	14	13	12	11	10	9	8	
Access										
Reset										
	Bit	7	6	5	4	3	2	1	0	
		RUNSTDBY			MODE[2:0]			ENABLE	SWRST	
Access		R/W			R/W	R/W	R/W	R/W	R/W	
Reset		0			0	0	0	0	0	

**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 interrupt flags set at the time of time-out will remain set.

Value	Description
0	Time-out disabled.
1	Time-out enabled.

**Bit 27 – SCLSM** SCL Clock Stretch Mode

This bit controls when SCL will be stretched for software interaction.

This bit is not synchronized.

Value	Description
0	SCL stretch according to <a href="#">Figure 30-10</a>
1	SCL stretch only after ACK bit according to <a href="#">Figure 30-11</a>

**Bits 25:24 – SPEED[1:0]** Transfer Speed

These bits define bus speed.

These bits are not synchronized.

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

**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 interrupt flags set at the time of time-out will remain set. If the address was recognized, PREC will be set when a STOP is received.

This bit is not synchronized.

Value	Description
0	Time-out disabled
1	Time-out enabled

**Bits 21:20 – SDAHOLD[1:0]** SDA Hold Time

These bits define the SDA hold time with respect to the negative edge of SCL.

These bits are not synchronized.

Value	Name	Description
0x0	DIS	Disabled
0x1	75	50-100ns hold time
0x2	450	300-600ns hold time
0x3	600	400-800ns hold time

**Bit 16 – PINOUT** Pin Usage

This bit sets the pin usage to either two- or four-wire operation:

This bit is not synchronized.

Value	Description
0	4-wire operation disabled
1	4-wire operation enabled

**Bit 7 – RUNSTDBY** Run in Standby

This bit defines the functionality in standby sleep mode.

This bit is not synchronized.

Value	Description
0	Disabled – All reception is dropped.
1	Wake on address match, if enabled.

**Bits 4:2 – MODE[2:0]** Operating Mode

These bits must be written to 0x04 to select the I<sup>2</sup>C slave serial communication interface of the SERCOM.

These bits are not synchronized.

**Bit 1 – ENABLE** Enable

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 will be cleared when the operation is complete.

This bit is not enable-protected.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled.

**Bit 0 – SWRST** Software Reset

Writing '0' to this bit has no effect.

Writing '1' to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.

Writing '1' 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.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

### 30.8.2 Control B

**Name:** CTRLB  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	[Greyed out bits 31:24]							
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	[Greyed out bits 23:19]					ACKACT	CMD[1:0]	
Access						R/W	W	W
Reset						0	0	0
Bit	15	14	13	12	11	10	9	8
	AMODE[1:0]		[Greyed out bits 13:11]			AACKEN	GCMD	SMEN
Access	R/W	R/W				R/W	R/W	R/W
Reset	0	0				0	0	0
Bit	7	6	5	4	3	2	1	0
	[Greyed out bits 7:0]							
Access								
Reset								

**Bit 18 – ACKACT** Acknowledge Action

This 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=1), the acknowledge action is performed when the DATA register is read.

This bit is not enable-protected.

Value	Description
0	Send ACK
1	Send NACK

**Bits 17:16 – CMD[1:0]** Command

This bit field triggers the slave operation as the below. 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.

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 30-3. 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.

These bits are not write-synchronized.

Value	Name	Description
0x0	MASK	The slave responds to the address written in ADDR.ADDR masked by the value in ADDR.ADDRMASK. See <i>SERCOM – Serial Communication Interface</i> for additional information.
0x1	2_ADDR	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.

**Bit 10 – AACKEN** Automatic Acknowledge Enable

This bit enables the address to be automatically acknowledged if there is an address match.

This bit is not write-synchronized.

Value	Description
0	Automatic acknowledge is disabled.
1	Automatic acknowledge is enabled.

**Bit 9 – GCMD** PMBus Group Command

This bit enables PMBus group command support. When enabled, the Stop Recived interrupt flag (INTFLAG.PREC) will be set when a STOP condition is detected if the slave has been addressed since the last STOP condition on the bus.

This bit is not write-synchronized.

---

---

Value	Description
0	Group command is disabled.
1	Group command is enabled.

**Bit 8 – SMEN** Smart Mode Enable

When smart mode is enabled, data is acknowledged automatically when DATA.DATA is read.

This bit is not write-synchronized.

Value	Description
0	Smart mode is disabled.
1	Smart mode is enabled.

**Related Links**

[SERCOM – Serial Communication Interface](#)

### 30.8.3 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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	6	5	4	3	2	1	0
	ERROR					DRDY	AMATCH	PREC
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

**Bit 7 – ERROR** Error Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

**Bit 2 – DRDY** Data Ready Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Data Ready bit, which disables the Data Ready interrupt.

Value	Description
0	The Data Ready interrupt is disabled.
1	The Data Ready interrupt is enabled.

**Bit 1 – AMATCH** Address Match Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Address Match Interrupt Enable bit, which disables the Address Match interrupt.

Value	Description
0	The Address Match interrupt is disabled.
1	The Address Match interrupt is enabled.

**Bit 0 – PREC** Stop Received Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Stop Received Interrupt Enable bit, which disables the Stop Received interrupt.

Value	Description
0	The Stop Received interrupt is disabled.
1	The Stop Received interrupt is enabled.



### 30.8.4 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x16  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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).

Bit	7	6	5	4	3	2	1	0
	ERROR					DRDY	AMATCH	PREC
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

**Bit 7 – ERROR** Error Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

**Bit 2 – DRDY** Data Ready Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Data Ready bit, which enables the Data Ready interrupt.

Value	Description
0	The Data Ready interrupt is disabled.
1	The Data Ready interrupt is enabled.

**Bit 1 – AMATCH** Address Match Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Address Match Interrupt Enable bit, which enables the Address Match interrupt.

Value	Description
0	The Address Match interrupt is disabled.
1	The Address Match interrupt is enabled.

**Bit 0 – PREC** Stop Received Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Stop Received Interrupt Enable bit, which enables the Stop Received interrupt.

Value	Description
0	The Stop Received interrupt is disabled.
1	The Stop Received interrupt is enabled.

### 30.8.5 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/W	R/W	R/W
Reset	0					0	0	0

**Bit 7 – ERROR** Error

This bit is set when any error is detected. Errors that will set this flag have corresponding status flags in the STATUS register. The corresponding bits in STATUS are SEXTTOUT, LOWTOUT, COLL, and BUSERR.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the flag.

**Bit 2 – DRDY** Data Ready

This flag is set when a I<sup>2</sup>C 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 '0' to this bit has no effect.

Writing '1' to this bit will clear the Data Ready interrupt flag.

**Bit 1 – AMATCH** Address Match

This flag is set when the I<sup>2</sup>C slave address match logic detects that a valid address has been received.

The flag is cleared by hardware when CTRL.CMD is written.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Address Match interrupt flag. 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, unless the PMBus Group Command is enabled in the Control B register (CTRLB.GCMD=1).

This flag is cleared by hardware after a command is issued on the next address match.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Stop Received interrupt flag.

**30.8.6 Status**

**Name:** STATUS  
**Offset:** 0x1A  
**Reset:** 0x0000

	Bit	15	14	13	12	11	10	9	8
						LENERR	HS	SEXTTOUT	
Access						R/W	R/W	R/W	
Reset						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/W	R/W
Reset		0	0		0	0	0	0	0

**Bit 11 – LENERR** Transaction Length Error

This bit is set when the length counter is enabled (LENGTH.LENEN) and a STOP or repeated START is received before or after the length in LENGTH.LEN is reached.

This bit is cleared automatically when responding to a new start condition with ACK or NACK (CTRLB.CMD=0x3) or when INTFLAG.AMATCH is cleared.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the status.

**Bit 10 – HS** High-speed

This bit is set if the slave detects a START followed by a Master Code transmission.

Writing a '0' to this bit has no effect.

Writing a '1' 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.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the status.

Value	Description
0	No SCL low extend time-out has occurred.
1	SCL low extend time-out has occurred.

**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 I2C 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.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the status.

Value	Description
0	No SCL low time-out has occurred.
1	SCL low time-out has occurred.

**Bit 4 – SR** Repeated Start

When INTFLAG.AMATCH is raised due to an address match, SR indicates a repeated start or start condition.

This flag is only valid while the INTFLAG.AMATCH flag is one.

Value	Description
0	Start condition on last address match
1	Repeated start condition on last address match

**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.

Value	Description
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.

Value	Description
0	Master responded with ACK.
1	Master responded with NACK.

**Bit 1 – COLL** Transmit Collision

If set, the I2C slave was not able to transmit a high data or NACK bit, the I2C 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.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the status.

---

---

Value	Description
0	No collision detected on last data byte sent.
1	Collision detected on last data byte sent.

**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 I2C 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.

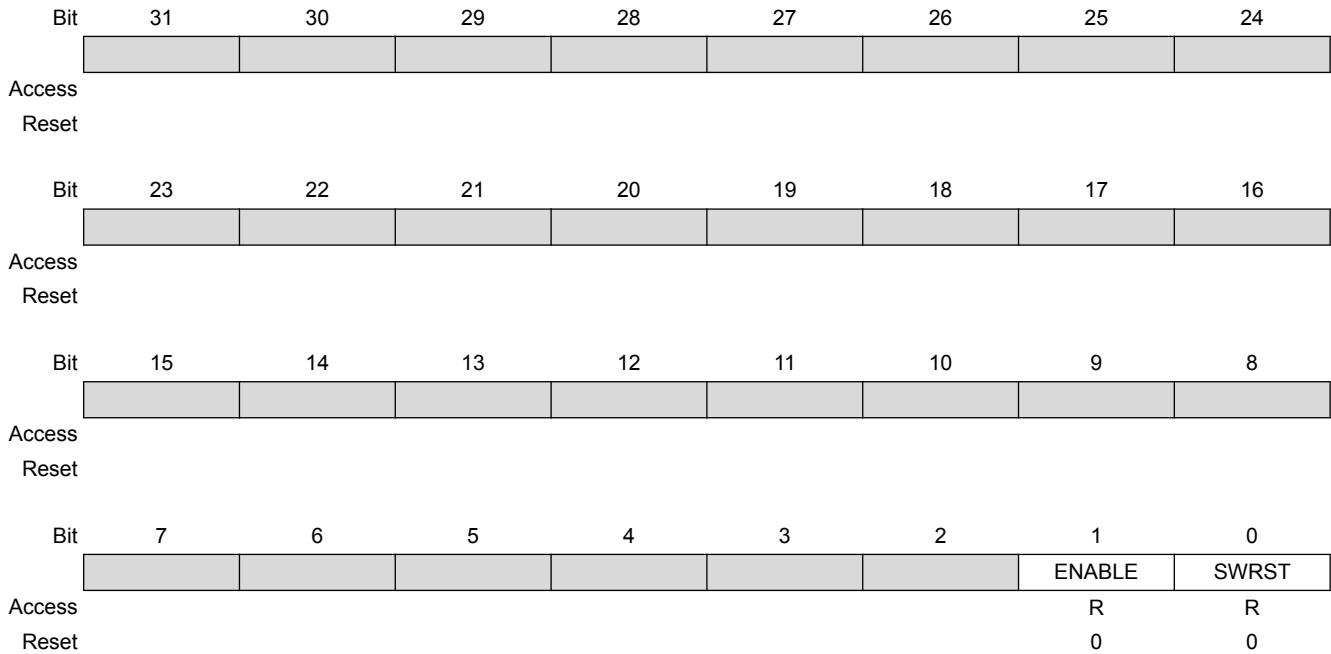
Writing a '1' to this bit will clear the status.

Writing a '0' to this bit has no effect.

Value	Description
0	No bus error detected.
1	Bus error detected.

### 30.8.7 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x1C  
**Reset:** 0x00000000



**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.

Value	Description
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.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

### 30.8.8 Address

**Name:** ADDR  
**Offset:** 0x24  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	ADDRMASK[9:7]							
Access						R/W	R/W	R/W
Reset						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	
Reset	0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8
	TENBITEN					ADDR[9:7]		
Access	R/W					R/W	R/W	R/W
Reset	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 26:17 – ADDRMASK[9:0] Address Mask**

These bits act 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 15 – TENBITEN Ten Bit Addressing Enable**

Value	Description
0	10-bit address recognition disabled.
1	10-bit address recognition enabled.

**Bits 10:1 – ADDR[9:0] Address**

These bits contain the I<sup>2</sup>C 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[6:0].

When using 10-bit addressing (ADDR.TENBITEN=1), the slave address is represented by 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**

A general call address is an address consisting of all-zeroes, including the direction bit (master write).

# ATSAMHAXEXXA

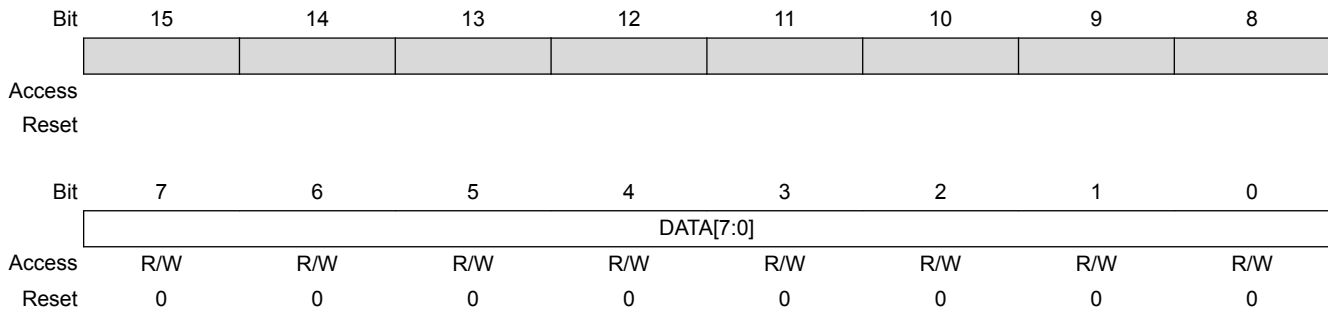
## SERCOM I2C – Inter-Integrated Circuit

Value	Description
0	General call address recognition disabled.
1	General call address recognition enabled.



**30.8.9 Data**

**Name:** DATA  
**Offset:** 0x28  
**Reset:** 0x0000  
**Property:** Write-Synchronized, Read-Synchronized



**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<sup>2</sup>C 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.

### 30.9 Register Summary - I2C Master

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]		ENABLE	SWRST	
		15:8								
		23:16	SEXTTOEN	MEXTTOEN	SDAHOLD[1:0]				PINOUT	
		31:24		LOWTOUT	INACTOUT[1:0]		SCLSM	SPEED[1:0]		
0x04	CTRLB	7:0						QCEN	SMEN	
		15:8								
		23:16					ACKACT	CMD[1:0]		
		31:24								
0x08 ... 0x0B	Reserved									
0x0C	BAUD	7:0	BAUD[7:0]							
		15:8	BAUDLOW[7:0]							
		23:16	HSBAUD[7:0]							
		31:24	HSBAUDLOW[7:0]							
0x10 ... 0x13	Reserved									
0x14	INTENCLR	7:0	ERROR					SB	MB	
0x15	Reserved									
0x16	INTENSET	7:0	ERROR					SB	MB	
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR					SB	MB	
0x18	DATA	7:0	DATA[7:0]							
		15:8								
0x1A	STATUS	7:0	CLKHOLD	LOWTOUT	BUSSTATE[1:0]			RXNACK	ARBLOST	BUSERR
		15:8						LENERR	SEXTTOUT	MEXTTOUT
0x1C	SYNDBUSY	7:0						SYSOP	ENABLE	SWRST
		15:8								
		23:16								
		31:24								
Error for offset 32										
0x21 ... 0x23	Reserved									
0x24	ADDR	7:0	ADDR[7:0]							
		15:8	TENBITEN	HS	LENEN			ADDR[10:8]		
		23:16	LEN[7:0]							
		31:24								
0x28 ... 0x2F	Reserved									
0x30	DBGCTRL	7:0							DBGSTOP	

### **30.10 Register Description - I<sup>2</sup>C Master**

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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#).

Some registers are synchronized when read and/or written. Synchronization is denoted by the "Write-Synchronized" or the "Read-Synchronized" property in each individual register description. For details, refer to [Synchronization](#).

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 30.10.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
		LOWTOUT	INACTOUT[1:0]		SCLSM		SPEED[1:0]	
Access		R/W	R/W	R/W	R/W		R/W	R/W
Reset		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/W
Reset	0	0	0	0				0
Bit	15	14	13	12	11	10	9	8
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

**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.LOWTOUT and STATUS.BUSERR status bits will be set.

This bit is not synchronized.

Value	Description
0	Time-out disabled.
1	Time-out enabled.

**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 I<sup>2</sup>C master or slave is holding the SCL low.

Enabling this option is necessary for SMBus compatibility, but can also be used in a non-SMBus set-up.

Calculated time-out periods are based on a 100kHz baud rate.

These bits are not synchronized.

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)

**Bit 27 – SCLSM** SCL Clock Stretch Mode

This bit controls when SCL will be stretched for software interaction.

This bit is not synchronized.

Value	Description
0	SCL stretch according to <a href="#">Figure 30-5</a> .
1	SCL stretch only after ACK bit, <a href="#">Figure 30-6</a> .

**Bits 25:24 – SPEED[1:0]** Transfer Speed

These bits define bus speed.

These bits are not synchronized.

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

**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 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.

This bit is not synchronized.

Value	Description
0	Time-out disabled
1	Time-out enabled

**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 released. The MEXTTOUT and BUSERR status bits will be set.

This bit is not synchronized.

Value	Description
0	Time-out disabled
1	Time-out enabled

**Bits 21:20 – SDAHOLD[1:0]** SDA Hold Time

These bits define the SDA hold time with respect to the negative edge of SCL.

These bits are not synchronized.

Value	Name	Description
0x0	DIS	Disabled
0x1	75NS	50-100ns hold time
0x2	450NS	300-600ns hold time
0x3	600NS	400-800ns hold time

**Bit 16 – PINOUT** Pin Usage

This bit set the pin usage to either two- or four-wire operation:

This bit is not synchronized.

Value	Description
0	4-wire operation disabled.
1	4-wire operation enabled.

**Bit 7 – RUNSTDBY** Run in Standby

This bit defines the functionality in standby sleep mode.

This bit is not synchronized.

Value	Description
0	GCLK_SERCOMx_CORE is disabled and the I <sup>2</sup> C master will not operate in standby sleep mode.
1	GCLK_SERCOMx_CORE is enabled in all sleep modes.

**Bits 4:2 – MODE[2:0]** Operating Mode

These bits must be written to 0x5 to select the I<sup>2</sup>C master serial communication interface of the SERCOM.

These bits are not synchronized.

**Bit 1 – ENABLE** Enable

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 will be cleared when the operation is complete.

This bit is not enable-protected.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled.

**Bit 0 – SWRST** Software Reset

Writing '0' to this bit has no effect.

Writing '1' to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.

Writing '1' 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.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

**30.10.2 Control B**

**Name:** CTRLB  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	[Greyed out register bits]							
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	[Greyed out register bits]					ACKACT	CMD[1:0]	
Access						R/W	W	W
Reset						0	0	0
Bit	15	14	13	12	11	10	9	8
	[Greyed out register bits]						QCEN	SMEN
Access							R/W	R/W
Reset							0	0
Bit	7	6	5	4	3	2	1	0
	[Greyed out register bits]							
Access								
Reset								

**Bit 18 – ACKACT** Acknowledge Action

This bit defines the I<sup>2</sup>C master's acknowledge behavior after a data byte is received from the I<sup>2</sup>C 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.

This bit is not enable-protected.

This bit is not write-synchronized.

Value	Description
0	Send ACK.
1	Send NACK.

**Bits 17:16 – CMD[1:0]** Command

Writing these bits triggers a master operation as described below. 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 either the Slave on Bus interrupt flag (INTFLAG.SB) or Master on Bus interrupt flag (INTFLAG.MB) is '1'.



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 the System Operation bit in the Synchronization Busy register (SYNCBUSY.SYSOP).

**Table 30-4. 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.

**Bit 9 – QCEN** Quick Command Enable

This bit is not write-synchronized.

Value	Description
0	Quick Command is disabled.
1	Quick Command is enabled.

**Bit 8 – SMEN** Smart Mode Enable

When smart mode is enabled, acknowledge action is sent when DATA.DATA is read.

This bit is not write-synchronized.

Value	Description
0	Smart mode is disabled.
1	Smart mode is enabled.

### 30.10.3 Baud Rate

**Name:** BAUD  
**Offset:** 0x0C  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, 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 non-zero: HSBAUDLOW indicates the SCL low time in High-speed mode according to

$$HSBAUDLOW = f_{GCLK} \cdot T_{LOW} - 1$$

HSBAUDLOW equal to zero: 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**

This bit field indicates the SCL high time in High-speed mode 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.

$$HSBAUD = f_{GCLK} \cdot T_{HIGH} - 1$$

**Bits 15:8 – BAUDLOW[7:0] Master Baud Rate Low**

If this bit field is non-zero, the SCL low time will be described by the value written.

For more information on how to calculate the frequency, see SERCOM [Clock Generation – Baud-Rate Generator](#).

**Bits 7:0 – BAUD[7:0] Master Baud Rate**

This bit field 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 [Clock Generation – Baud-Rate Generator](#).

### 30.10.4 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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	6	5	4	3	2	1	0
	ERROR						SB	MB
Access	R/W						R/W	R/W
Reset	0						0	0

**Bit 7 – ERROR** Error Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

**Bit 1 – SB** Slave on Bus Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Slave on Bus Interrupt Enable bit, which disables the Slave on Bus interrupt.

Value	Description
0	The Slave on Bus interrupt is disabled.
1	The Slave on Bus interrupt is enabled.

**Bit 0 – MB** Master on Bus Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Master on Bus Interrupt Enable bit, which disables the Master on Bus interrupt.

Value	Description
0	The Master on Bus interrupt is disabled.
1	The Master on Bus interrupt is enabled.

### 30.10.5 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x16  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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).

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

**Bit 7 – ERROR** Error Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

**Bit 1 – SB** Slave on Bus Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Slave on Bus Interrupt Enable bit, which enables the Slave on Bus interrupt.

Value	Description
0	The Slave on Bus interrupt is disabled.
1	The Slave on Bus interrupt is enabled.

**Bit 0 – MB** Master on Bus Interrupt Enable  
 Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Master on Bus Interrupt Enable bit, which enables the Master on Bus interrupt.

Value	Description
0	The Master on Bus interrupt is disabled.
1	The Master on Bus interrupt is enabled.

### 30.10.6 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/W	R/W
Reset		0						0	0

**Bit 7 – ERROR** Error

This flag is cleared by writing '1' to it.

This bit is set when any error is detected. Errors that will set this flag have corresponding status bits in the STATUS register. These status bits are LENERR, SEXTTOUT, MEXTTOUT, LOWTOUT, ARBLOST, and BUSERR.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the flag.

**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<sup>2</sup>C 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 '1' 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 '0' to this bit has no effect.

**Bit 0 – MB** Master on Bus

This flag 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, or 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<sup>2</sup>C 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

Writing '1' 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 '0' to this bit has no effect.

### 30.10.7 Status

**Name:** STATUS  
**Offset:** 0x1A  
**Reset:** 0x0000  
**Property:** Write-Synchronized

	Bit	15	14	13	12	11	10	9	8
							LENERR	SEXTTOUT	MEXTTOUT
Access							R/W	R/W	R/W
Reset							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/W	R/W		R	R/W	R/W
Reset		0	0	0	0		0	0	0

**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 '1' to this bit location will clear STATUS.LENERR. This flag is automatically cleared when writing to the ADDR register.

Writing '0' 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.

This bit is automatically cleared when writing to the ADDR register.

Writing '1' to this bit location will clear SEXTTOUT. Normal use of the I<sup>2</sup>C interface does not require the SEXTTOUT flag to be cleared by this method.

Writing '0' 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 '1' to this bit location will clear STATUS.MEXTTOUT. This flag is automatically cleared when writing to the ADDR register.

Writing '0' to this bit has no effect.

This bit is not write-synchronized.

**Bit 7 – CLKHOLD** Clock Hold

This bit is set when the master is holding the SCL line low, stretching the I<sup>2</sup>C clock. Software should consider this bit when INTFLAG.SB or INTFLAG.MB is set.



This bit is cleared when the corresponding interrupt flag is cleared and the next operation is given.

Writing '0' to this bit has no effect.

Writing '1' 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 '1' to this bit location will clear this bit. This flag is automatically cleared when writing to the ADDR register.

Writing '0' 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<sup>2</sup>C bus state.

When in 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 BUSSTATE to idle will set SYNCBUSY.SYSOP.

Value	Name	Description
0x0	UNKNOWN	The bus state is unknown to the I <sup>2</sup> C 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 <sup>2</sup> C master is the current owner of the bus
0x3	BUSY	Some other I <sup>2</sup> C master owns the bus

**Bit 2 – RXNACK** Received Not Acknowledge

This bit indicates whether the last address or data packet sent was acknowledged or not.

Writing '0' to this bit has no effect.

Writing '1' to this bit has no effect.

This bit is not write-synchronized.

Value	Description
0	Slave responded with ACK.
1	Slave responded with NACK.

**Bit 1 – ARBLOST** Arbitration Lost

This bit 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 (INTFLAG.MB) will be set when STATUS.ARBLOST is set.

Writing the ADDR.ADDR register will automatically clear STATUS.ARBLOST.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

This bit is not write-synchronized.

**Bit 0 – BUSERR** Bus Error

This bit 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<sup>2</sup>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 BUSERR.

If the I<sup>2</sup>C 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.

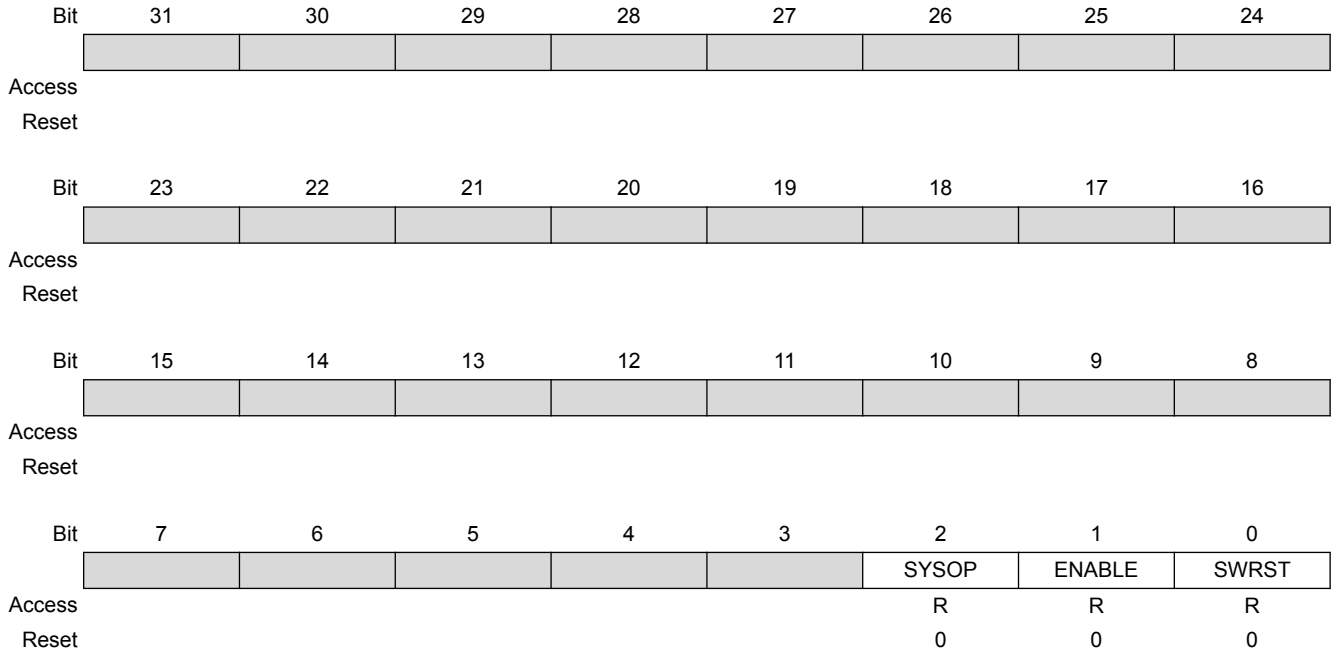
Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

This bit is not write-synchronized.

### 30.10.8 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x1C  
**Reset:** 0x00000000



#### Bit 2 – SYSOP System Operation Synchronization Busy

Value	Description
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.

Value	Description
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.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

**30.10.9 Address**

**Name:** ADDR  
**Offset:** 0x24  
**Reset:** 0x0000  
**Property:** Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
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/W	R/W	R/W
Reset	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 23:16 – LEN[7:0] Transaction Length**

These bits define the transaction length of a DMA transaction from 0 to 255 bytes. The Transfer Length Enable (LENEN) bit must be written to '1' in order to use DMA.

**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.

Value	Description
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.

Value	Description
0	High-speed transfer disabled.
1	High-speed transfer enabled.

**Bit 13 – LENEN Transfer Length Enable**

Value	Description
0	Automatic transfer length disabled.
1	Automatic transfer length enabled.

**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<sup>2</sup>C master will await further operation until the bus becomes IDLE.

IDLE: The I<sup>2</sup>C 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.

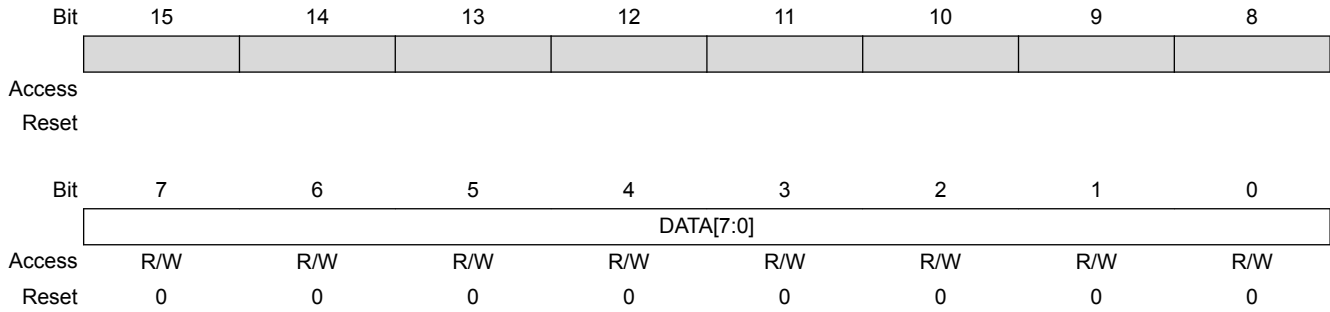
STATUS.BUSERR, STATUS.ARBLOST, INTFLAG.MB and INTFLAG.SB will be cleared when ADDR is written.

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<sup>2</sup>C 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.

**30.10.10 Data**

**Name:** DATA  
**Offset:** 0x18  
**Reset:** 0x0000  
**Property:** Write-Synchronized, Read-Synchronized



**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 is reading the last data byte after the stop condition has been sent.

Accessing DATA.DATA auto-triggers I<sup>2</sup>C 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.

**30.10.11 Debug Control**

**Name:** DBGCTRL  
**Offset:** 0x30  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

**Bit 0 – DBGSTOP** Debug Stop Mode

This bit controls functionality when the CPU is halted by an external debugger.

Value	Description
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.

## 31. TC – Timer/Counter

### 31.1 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).

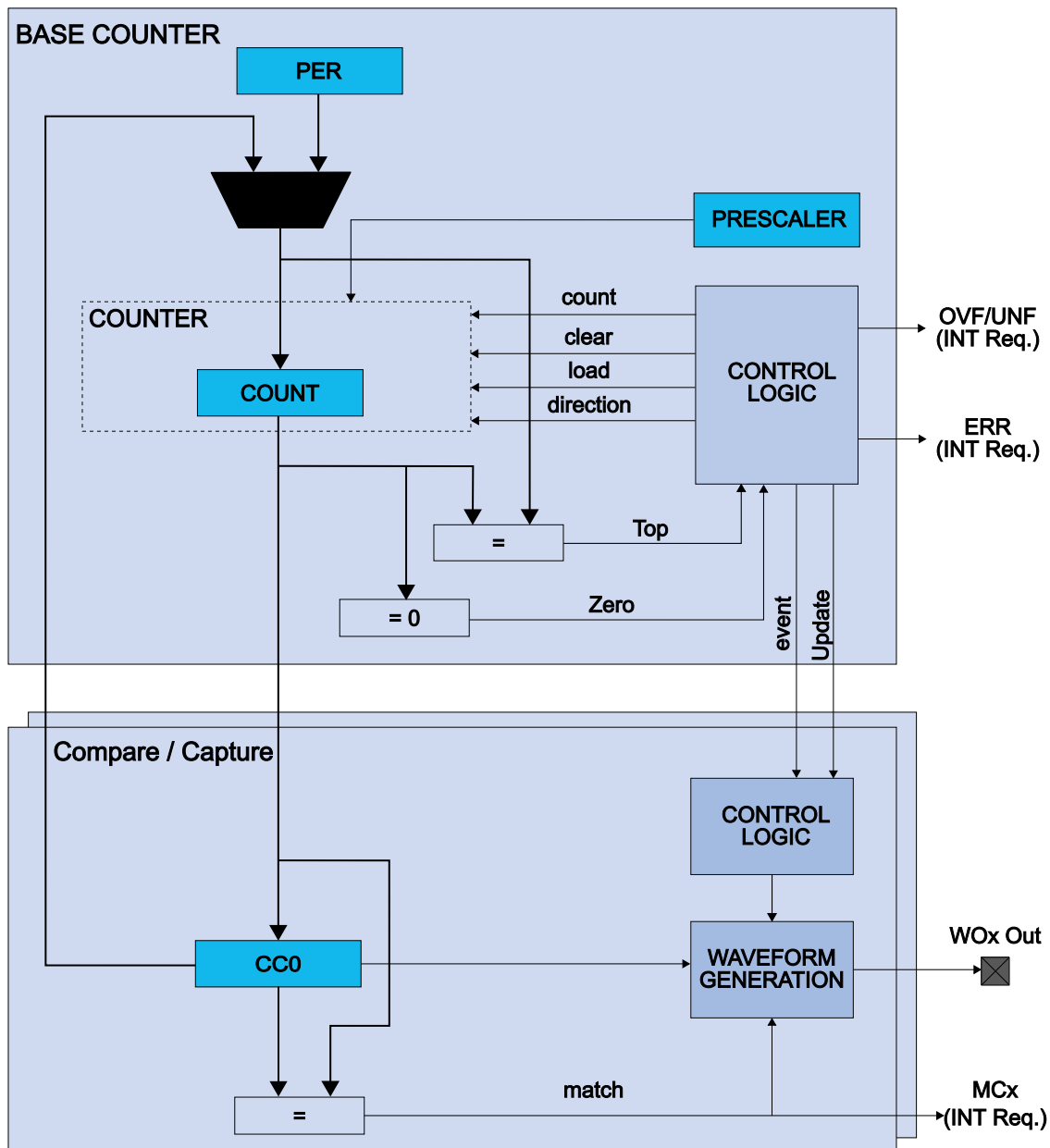
### 31.2 Features

- Selectable configuration
  - Up to five 16-bit Timer/Counters (TC), each configurable as:
    - 8-bit TC with two compare/capture channels
    - 16-bit TC with two compare/capture channels
    - 32-bit TC with two compare/capture channels, by using two TCs
- 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
- Can be used with DMA and to trigger DMA transactions



31.3 Block Diagram

Figure 31-1. Timer/Counter Block Diagram



31.4 Signal Description

Signal Name	Type	Description
WO[1:0]	Digital output	Waveform output

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

**Related Links**[I/O Multiplexing and Considerations](#)**31.5 Product Dependencies**

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

**31.5.1 I/O Lines**

In order to use the I/O lines of this peripheral, the I/O pins must be configured using the I/O Pin Controller (PORT).

**Related Links**[PORT - I/O Pin Controller](#)**31.5.2 Power Management**

This peripheral can continue to operate in any sleep mode where its source clock is running. The interrupts can wake up the device from sleep modes. Events connected to the event system can trigger other operations in the system without exiting sleep modes.

**Related Links**[PM – Power Manager](#)**31.5.3 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”.

The different TC instances are paired, even and odd, starting from TC3, 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](#) for further details.

**Related Links**[Peripheral Clock Masking](#)**31.5.4 DMA**

The DMA request lines are connected to the DMA Controller (DMAC). In order to use DMA requests with this peripheral the DMAC must be configured first. Refer to *DMAC – Direct Memory Access Controller* for details.

**Related Links**[DMAC – Direct Memory Access Controller](#)**31.5.5 Interrupts**

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. Refer to *Nested Vector Interrupt Controller* for details.

**Related Links**

### Nested Vector Interrupt Controller

#### 31.5.6 Events

The events of this peripheral are connected to the Event System.

##### Related Links

[EVSYS – Event System](#)

#### 31.5.7 Debug Operation

When the CPU is halted in debug mode, this peripheral will halt normal operation. This peripheral can be forced to continue operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

#### 31.5.8 Register Access Protection

Registers with write-access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag register (INTFLAG)
- Status register (STATUS)
- Read Request register (READREQ)
- Count register (COUNT)
- Period register (PER)
- Compare/Capture Value registers (CCx)

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

Write-protection does not apply for accesses through an external debugger.

#### 31.5.9 Analog Connections

Not applicable.

### 31.6 Functional Description

#### 31.6.1 Principle of Operation

The following definitions are used throughout the documentation:

**Table 31-1. Timer/Counter Definitions**

Name	Description
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be the same as Period (PER) or the Compare Channel 0 (CC0) register value depending on the waveform generator mode in <a href="#">Waveform Output Operations</a> .
ZERO	The counter is ZERO when it contains all zeroes
MAX	The counter reaches MAX when it contains all ones
UPDATE	The timer/counter signals an update when it reaches ZERO or TOP, depending on the direction settings.

Name	Description
Timer	The timer/counter clock control is handled by an internal source
Counter	The clock control is handled externally (e.g. counting external events)
CC	For compare operations, the CC are referred to as “compare channels” For capture operations, the CC are referred to as “capture channels.”

The counter in the TC can either count events from the Event System, or clock ticks of the GCLK\_TCx clock, which may be divided by the prescaler.

The counter value is passed to the CCx where it can be either compared to user-defined values or captured.

The compare and capture registers (CCx) and counter register (COUNT) can be configured as 8-, 16- or 32-bit registers, with according MAX values. Mode settings determine the maximum range of the counter.

In 8-bit mode, Period Value (PER) is also available. The counter range and the operating frequency determine the maximum time resolution achievable with the TC peripheral.

The TC can be set to count up or down. Under normal operation, the counter value is continuously compared to the TOP or ZERO value to determine whether the counter has reached that value. On a comparison match the TC can request DMA transactions, or generate interrupts or events for the Event System. On a comparison match the TC can request DMA transactions, or generate interrupts or events for the Event System.

In compare operation, the counter value is continuously compared to the values in the CCx registers. In case of a match the TC can request DMA transactions, or generate interrupts or events for the Event System. In waveform generator mode, these comparisons are used to set the waveform period or pulse width.

Capture operation can be enabled to perform input signal period and pulse width measurements, or to capture selectable edges from an internal event from Event System.

### 31.6.2 Basic Operation

#### 31.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the TC is disabled (CTRLA.ENABLE =0):

- Control A register (CTRLA), except the Run Standby (RUNSTDBY), Enable (ENABLE) and Software Reset (SWRST) bits

Enable-protected bits in the CTRLA register can be written at the same time as CTRLA.ENABLE is written to '1', but not at the same time as CTRLA.ENABLE is written to '0'. Enable-protection is denoted by the "Enable-Protected" property in the register description. The following bits are enable-protected:

- Event Action bits in the Event Control register (EVCTRL.EVACT)

Before enabling the TC, the peripheral must be configured by the following steps:

1. Enable the TC bus clock (CLK\_TCx\_APB).
2. Select 8-, 16- or 32-bit counter mode via the TC Mode bit group in the Control A register (CTRLA.MODE). The default mode is 16-bit.
3. Select one wave generation operation in the Waveform Generation Operation bit group in the Control A register (CTRLA.WAVEGEN).

4. If desired, the GCLK\_TCx clock can be prescaled via the Prescaler bit group in the Control A register (CTRLA.PRESCALER).
  - If the prescaler is used, select a prescaler synchronization operation via the Prescaler and Counter Synchronization bit group in the Control A register (CTRLA.PRESYNC).
5. Select one-shot operation by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT).
6. If desired, configure the counting direction 'down' (starting from the TOP value) by writing a '1' to the Counter Direction bit in the Control B register (CTRLBSET.DIR).
7. For capture operation, enable the individual channels to capture in the Capture Channel x Enable bit group in the Control C register (CTRLC.CPTEN).
8. If desired, enable inversion of the waveform output or IO pin input signal for individual channels via the Waveform Output Invert Enable bit group in the Control C register (CTRLC.INVEN).

### 31.6.2.2 Enabling, Disabling and Resetting

The TC is enabled by writing a '1' 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 in order to avoid undefined behavior.

### 31.6.2.3 Prescaler Selection

The GCLK\_TCx is fed into the internal prescaler.

The prescaler consists of a counter that counts up to the selected prescaler value, whereupon the output of the prescaler toggles.

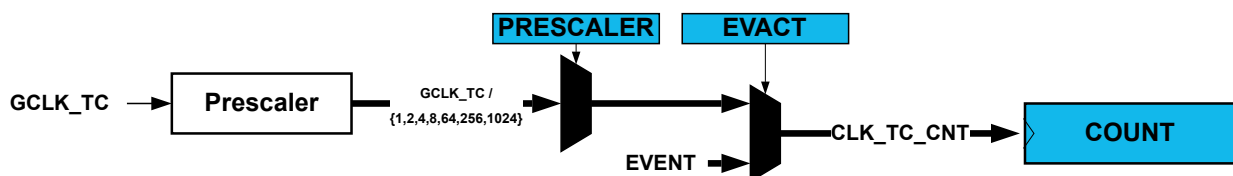
If the prescaler value is higher than one, the counter update condition can be optionally executed on the next GCLK\_TCx clock pulse or the next prescaled clock pulse. For further details, refer to Prescaler (CTRLA.PRESCALER) and Counter Synchronization (CTRLA.PRESYNC) description.

Prescaler outputs 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).

**Note:** When counting events, the prescaler is bypassed.

The joint stream of prescaler ticks and event action ticks is called CLK\_TC\_CNT.

**Figure 31-2. Prescaler**



### 31.6.2.4 Counter Mode

The counter mode is selected by the Mode bit group in the Control A register (CTRLA.MODE). By default, the counter is enabled in the 16-bit counter resolution. Three counter resolutions 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: 16-bit 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. TC3 is paired with TC4, and TC5 is paired with TC6. TC7 does not support 32-bit resolution.

When paired, the TC peripherals are configured using the registers of the even-numbered TC (TC4 or TC6 respectively). The odd-numbered partner (TC3 or TC5 respectively) will act as slave, and the Slave bit in the Status register (STATUS.SLAVE) will be set. The register values of a 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.

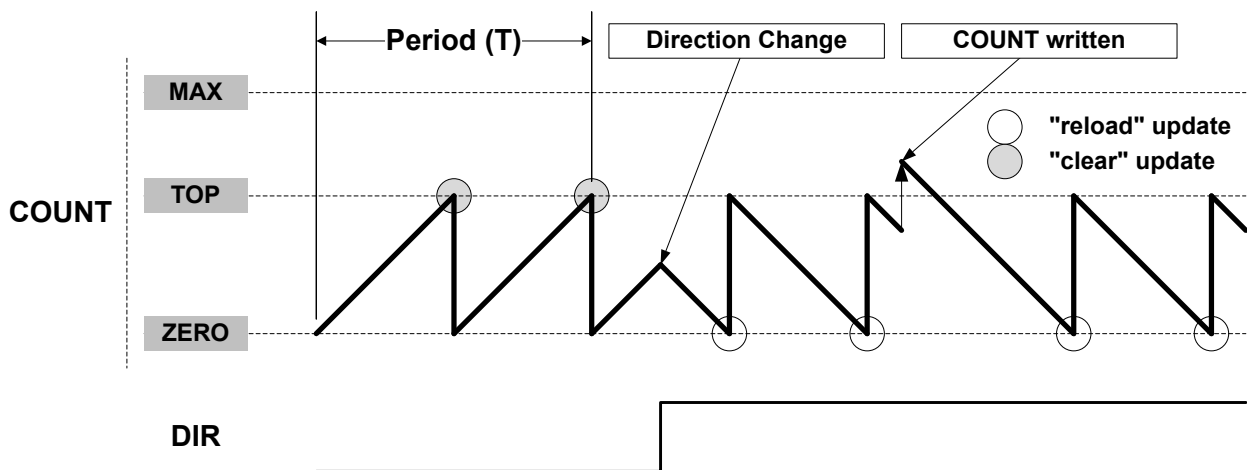
### 31.6.2.5 Counter Operations

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.

The counting direction is set by the Direction bit in the Control B register (CTRLB.DIR). If this bit is zero the counter is counting up, and counting down if CTRLB.DIR=1. The counter will count up or down for each tick (clock or event) until it reaches TOP or ZERO. When it is counting up and TOP is reached, the counter will be set to zero at the next tick (overflow) and the Overflow Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF) will be set. 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.

It is possible to change the counter value (by writing directly in the COUNT register) even when the counter is running. When starting the TC, the COUNT value will be either ZERO or TOP (depending on the counting direction set by CTRLBSET.DIR or CTRLBCLR.DIR), unless a different value has been written to it, or the TC has been stopped at a value other than ZERO. The write access has higher priority than count, clear, or reload. The direction of the counter can also be changed during normal operation. See also the figure below.

**Figure 31-3. Counter Operation**



#### Stop Command and Event Action

A Stop command can be issued from software by using Command bits in the Control B Set register (CTRLBSET.CMD = 0x2, STOP). When a Stop is detected while the counter is running, the counter will retain its current value. All waveforms are cleared and the Stop bit in the Status register is set (STATUS.STOP).

### Re-Trigger Command and Event Action

A re-trigger command can be issued from software by writing the Command bits in the Control B Set register (CTRLBSET.CMD = 0x1, RETRIGGER), or from event when a re-trigger event action is configured in the Event Control register (EVCTRL.EVACT = 0x1, RETRIGGER).

When the command is detected during counting operation, the counter will be reloaded or cleared, depending on the counting direction (CTRLBSET.DIR or CTRLBCLR.DIR). When the re-trigger command is detected while the counter is stopped, the counter will resume counting from the current value in the COUNT register.

**Note:** When a re-trigger event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACT=0x1, RETRIGGER), enabling the counter will not start the counter. The counter will start on the next incoming event and restart on corresponding following event.

### Count Event Action

The TC can count events. When an event is received, the counter increases or decreases the value, depending on direction settings (CTRLBSET.DIR or CTRLBCLR.DIR). The count event action can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT=0x2, COUNT).

### Start Event Action

The TC can start counting operation on an event when previously stopped. In this configuration, the event has no effect if the counter is already counting. When the peripheral is enabled, the counter operation starts when the event is received or when a re-trigger software command is applied.

The Start TC on Event action can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT=0x3, START).

### 31.6.2.6 Compare Operations

By default, the Compare/Capture channel is configured for compare operations.

When using the TC and the Compare/Capture Value registers (CCx) for compare operations, the counter value is continuously compared to the values in the CCx registers. This can be used for timer or for waveform operation.

### Waveform Output Operations

The compare channels can be used for waveform generation on output port pins. To make the waveform available on the connected pin, the following requirements must be fulfilled:

1. Choose a waveform generation mode in the Waveform Generation Operation bit in Waveform register (CTRLA.WAVEGEN).
2. Optionally invert the waveform output by writing the corresponding Waveform Output Invert Enable bit in the Control C register (CTRLC.INVx).
3. Configure the pins with the I/O Pin Controller. Refer to *PORT - I/O Pin Controller* for details.

The counter value is continuously compared with each CCx value. On a comparison match, the Match or Capture Channel x bit in the Interrupt Flag Status and Clear register (INTFLAG.MCx) will be set on the next zero-to-one transition of CLK\_TC\_CNT (see the next figure). An interrupt/and or event can be generated on comparison match when INTENSET.MCx=1 and/or EVCTRL.MCEx=1.

There are four waveform configurations for the Waveform Generation Operation bit group in the Control A register (CTRLA.WAVEGEN) . This will influence how the waveform is generated and impose restrictions on the top value. The configurations are:

- Normal frequency (NFRQ)
- Match frequency (MFRQ)
- Normal pulse-width modulation (NPWM)

- Match pulse-width modulation (MPWM)

When using NPWM or NFRQ configuration, the TOP will be determined by the counter resolution. In 8-bit counter mode, the Period register (PER) is used as TOP, and the TOP can be changed by writing to the PER register. In 16- and 32-bit counter mode, TOP is fixed to the maximum (MAX) value of the counter.

### Related Links

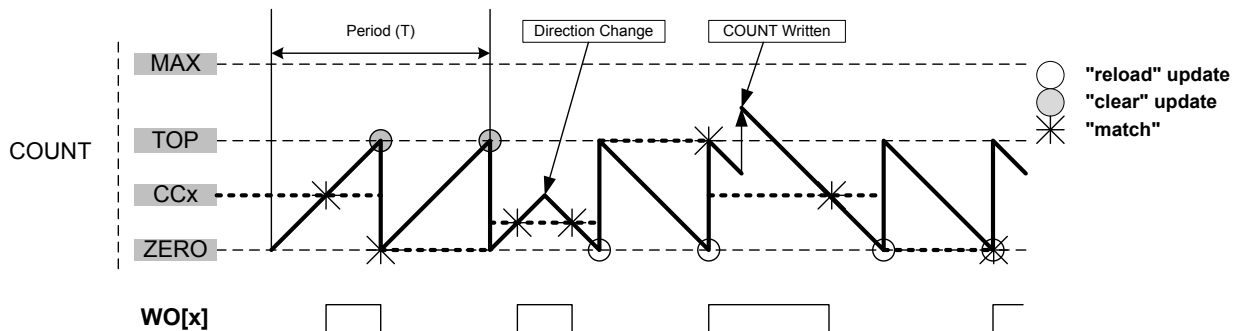
[PORT - I/O Pin Controller](#)

### Frequency Operation

#### Normal Frequency Generation (NFRQ)

For Normal Frequency Generation, the period time (T) is controlled by the period register (PER) for 8-bit counter mode and MAX for 16- and 32-bit mode. The waveform generation output (WO[x]) is toggled on each compare match between COUNT and CCx, and the corresponding Match or Capture Channel x Interrupt Flag (INTFLAG.MCx) will be set.

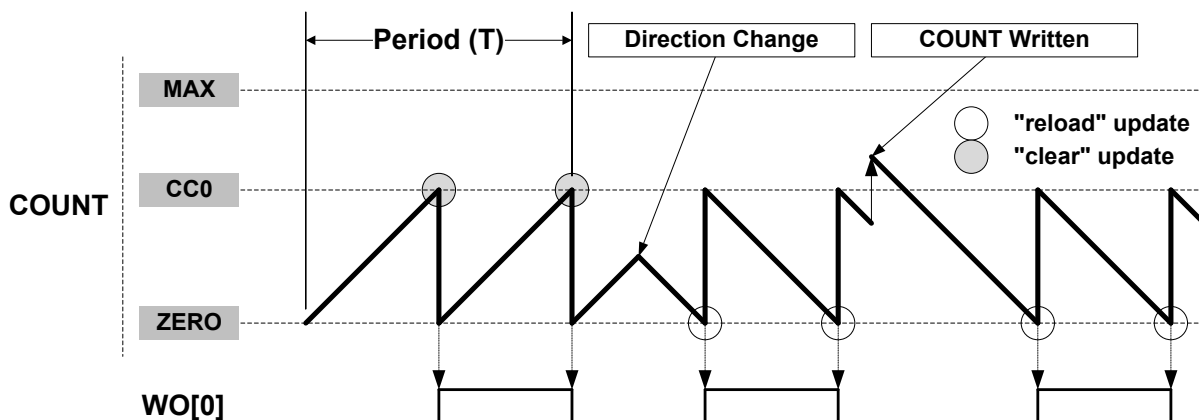
**Figure 31-4. Normal Frequency Operation**



#### Match Frequency Generation (MFRQ)

For Match Frequency Generation, the period time (T) is controlled by the CC0 register instead of PER or MAX. WO[0] toggles on each update condition.

**Figure 31-5. Match Frequency Operation**



### PWM Operation

#### Normal Pulse-Width Modulation Operation (NPWM)

NPWM uses single-slope PWM generation.

For single-slope PWM generation, the period time (T) is controlled by the TOP value, and CCx controls the duty cycle of the generated waveform output. When up-counting, the WO[x] is set at start or compare match between the COUNT and TOP values, and cleared on compare match between COUNT and CCx



register values. When down-counting, the WO[x] is cleared at start or compare match between the COUNT and ZERO values, and set on compare match between COUNT and CCx register values.

The following equation calculates the exact resolution for a single-slope PWM ( $R_{PWM\_SS}$ ) waveform:

$$R_{PWM\_SS} = \frac{\log(TOP+1)}{\log(2)}$$

The PWM frequency ( $f_{PWM\_SS}$ ) depends on TOP value and the peripheral clock frequency ( $f_{GCLK\_TC}$ ), and can be calculated by the following equation:

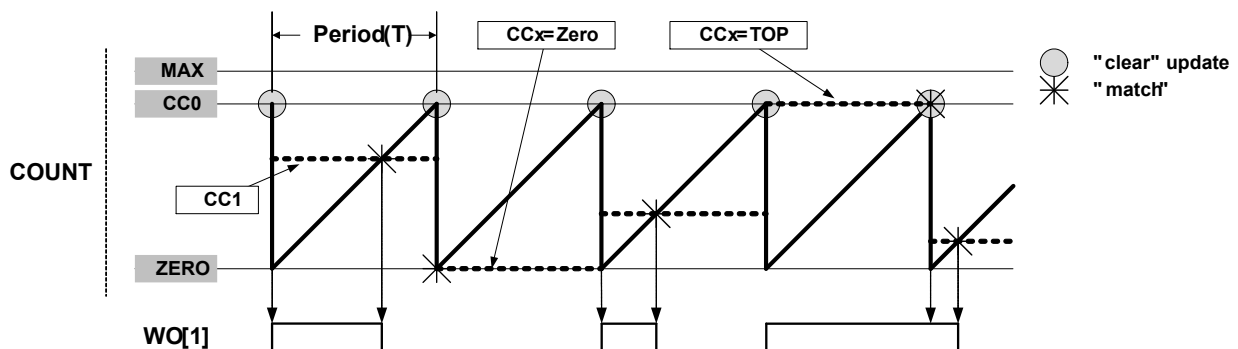
$$f_{PWM\_SS} = \frac{f_{GCLK\_TC}}{N(TOP+1)}$$

Where N represents the prescaler divider used (1, 2, 4, 8, 16, 64, 256, 1024).

### Match Pulse-Width Modulation Operation (MPWM)

In MPWM, the output of WO[1] is depending on CC1 as shown in the figure below. On every overflow/underflow, a one-TC-clock-cycle negative pulse is put out on WO[0] (not shown in the figure).

**Figure 31-6. Match PWM Operation**



The table below shows the update counter and overflow event/interrupt generation conditions in different operation modes.

**Table 31-2. Counter Update and Overflow Event/interrupt Conditions in TC**

Name	Operation	TOP	Update	Output Waveform		OVFIF/Event	
				On Match	On Update	Up	Down
NFRQ	Normal Frequency	PER	TOP/ ZERO	Toggle	Stable	TOP	ZERO
MFRQ	Match Frequency	CC0	TOP/ ZERO	Toggle	Stable	TOP	ZERO
NPWM	Single-slope PWM	PER	TOP/ ZERO	See description above.		TOP	ZERO
MPWM	Single-slope PWM	CC0	TOP/ ZERO	Toggle	Toggle	TOP	ZERO

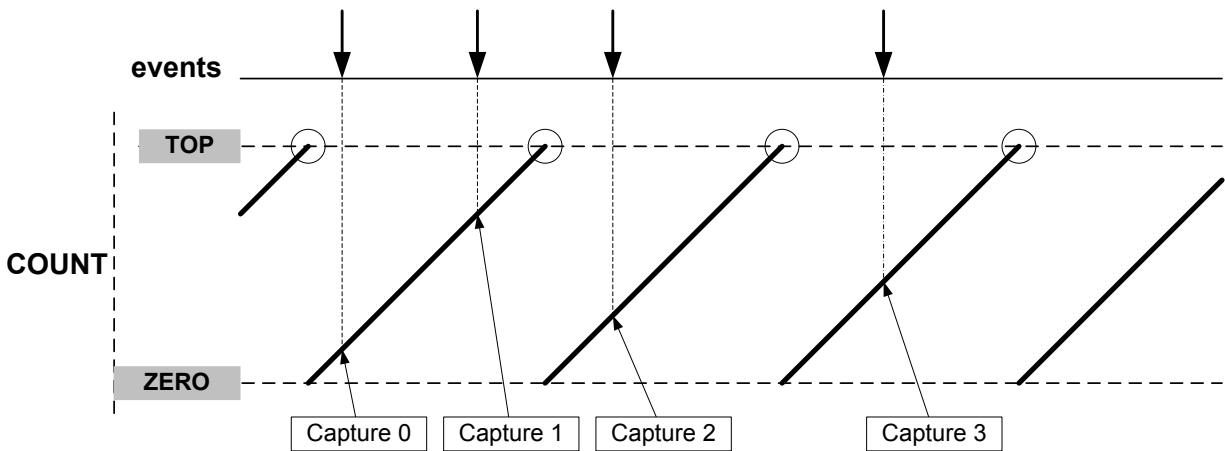
### Changing the Top Value

The counter period is changed by writing a new TOP value to the Period register (PER or CC0, depending on the waveform generation mode). 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. Then, the counter will count one extra cycle before the new TOP value is used.

COUNT and TOP are continuously compared, so when a new TOP value that is lower than current COUNT is written to TOP, COUNT will wrap before a compare match.



Figure 31-9. Input Capture Timing



The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

**Period and Pulse-Width (PPW) 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 and to characterize the frequency  $f$  and duty cycle of an input signal:

$$f = \frac{1}{T} \qquad \text{dutyCycle} = \frac{t_p}{T}$$

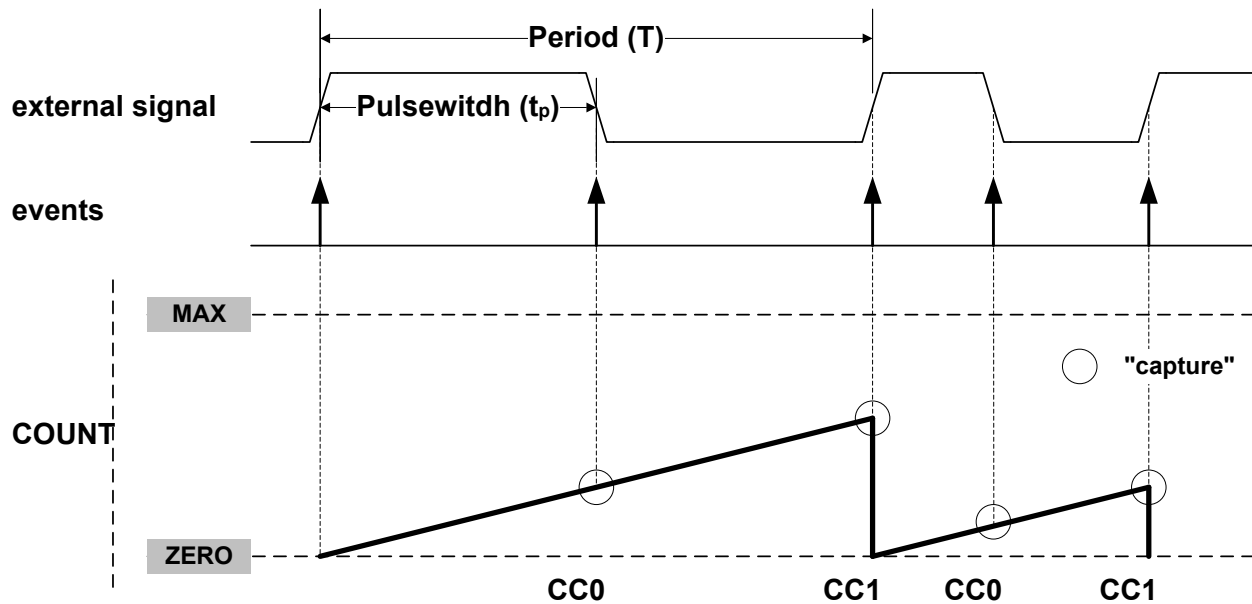
Selecting PWP (pulse-width, period) in the Event Action bit group in the Event Control register (EVCTRL.EVACT) enables the TC to perform one capture action on the rising edge and the other one on the falling edge. The period  $T$  will be captured into CC1 and the pulse width  $t_p$  in CC0. EVCTRL.EVACT=PPW (period and pulse-width) offers identical functionality, but will capture  $T$  into CC0 and  $t_p$  into CC1.

The TC Event Input Invert Enable bit 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=1, the wraparound will happen on the falling edge.

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 a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

Figure 31-10. PWP Capture



### 31.6.3 Additional Features

#### 31.6.3.1 One-Shot Operation

When one-shot is enabled, the counter automatically stops on the next counter overflow or underflow condition. When the counter is stopped, the Stop bit in the Status register (STATUS.STOP) is automatically set and the waveform outputs are set to zero.

One-shot operation is enabled by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT), and disabled by writing a '0' to CTRLBCLR.ONESHOT. When enabled, the TC will count until an overflow or underflow occurs and stops counting operation. The one-shot operation can be restarted by a re-trigger software command, a re-trigger event, or a start event. When the counter restarts its operation, STATUS.STOP is automatically cleared.

### 31.6.4 DMA, Interrupts and Events

Table 31-3. Module Request for TC

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Overflow / Underflow	YES	YES		YES	Cleared on next clock cycle
Channel Compare Match or Capture	YES	YES		YES <sup>1</sup>	For compare channel – Cleared on next clock cycle. For capture channel – cleared when CCx register is read

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Capture Overflow Error	YES				
Synchronization Ready	YES				
Start Counter			YES		
Retrigger Counter			YES		
Increment / Decrement counter			YES		
Simple Capture			YES		
Period Capture			YES		
Pulse Width Capture			YES		

Note: 1. Two DMA requests lines are available, one for each compare/capture channel.

### 31.6.4.1 DMA Operation

The TC can generate the following DMA requests:

- Overflow (OVF): the request is set when an update condition (overflow, underflow) is detected. The request is cleared on next clock cycle.
- Channel Match or Capture (MCx): for a compare channel, the request is set on each compare match detection and cleared on next clock cycle. For a capture channel, the request is set when valid data is present in CCx register, and cleared when CCx register is read.

When using the TC with the DMA OVF request, the new value will be transferred to the register after the update condition. This means that the value is updated after the DMA and synchronization delay, and if the COUNT value has reached the new value before PER or CCx is updated, a match will not happen.

When using the TC with the DMA MCx request and updating CCx with a value that is lower than the current COUNT when down-counting, or higher than the current COUNT when up-counting, this value could cause a new compare match before the counter overflows. This will trigger the next DMA transfer, update CCx again, and the previous value is disregarded from the output signal WO[x].

### 31.6.4.2 Interrupts

The TC has the following interrupt sources:

- Overflow/Underflow (OVF)
- Match or Capture Channel x (MCx)
- 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 '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a '1' 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 either the interrupt flag is cleared, the interrupt is disabled, or the TC is reset. 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* for details.

### Related Links

[Nested Vector Interrupt Controller](#)

### 31.6.4.3 Events

The TC can generate the following output events:

- Overflow/Underflow (OVF)
- Match or Capture (MC)

Writing a '1' to an Event Output bit in the Event Control register (EVCTRL.MCEOx) enables the corresponding output event. The output event is disabled by writing EVCTRL.MCEOx=0.

One of the following event actions can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT):

- Start TC (START)
- Re-trigger TC (RETRIGGER)
- Increment or decrement counter (depends on counter direction)
- Count on event (COUNT)
- Capture Period (PPW and PWP)

Writing a '1' to the TC Event Input bit in the Event Control register (EVCTRL.TCEI) enables input events to the TC. Writing a '0' to this bit disables input events to the TC. The TC requires only asynchronous event inputs. For further details on how configuring the asynchronous events, refer to *EVSYS - Event System*.

### Related Links

[EVSYS – Event System](#)

### 31.6.5 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.

### 31.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in the Control A register (CTRLA.SWRST)

- Enable bit in the Control A register (CTRLA.ENABLE)

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

The following registers are synchronized when written:

- Control B Clear register (CTRLBCLR)
- Control B Set register (CTRLBSET)
- Control C register (CTRLC)
- Count Value register (COUNT)
- Period Value register (PER)
- Compare/Capture Value registers (CCx)

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

The following registers are synchronized when read:

- Control B Clear register (CTRLBCLR)
- Control B Set register (CTRLBSET)
- Control C register (CTRLC)
- Count Value register (COUNT)
- Period Value register (PER)
- Compare/Capture Value registers (CCx)

Required read-synchronization is denoted by the "Read-Synchronized" property in the register description.

### Related Links

[Register Synchronization](#)

## 31.7 Register Summary

**Table 31-4. Register Summary – 8-bit Mode**

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	

Offset	Name	Bit Pos.								
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 31-5. Register Summary – 16-bit Mode**

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]							



Offset	Name	Bit Pos.								
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 31-6. Register Summary – 32-bit Mode**

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]

### 31.8 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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#)

Some registers are synchronized when read and/or written. Synchronization is denoted by the "Write-Synchronized" or the "Read-Synchronized" property in each individual register description. For details, refer to [Synchronization](#).

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 31.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			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/W	R/W		R/W	R/W	R/W	R/W
Reset		0	0		0	0	0	0

#### Bits 13:12 – PRESCSYNC[1:0] Prescaler and Counter Synchronization

These bits select whether the counter should wrap around on the next GCLK\_TCx clock or the next prescaled GCLK\_TCx clock. It also makes it possible to reset the prescaler.

These bits are not synchronized.

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.

This bit is not synchronized.

Value	Description
0	The TC is halted in standby.
1	The TC continues to run in standby.

#### Bits 10:8 – PRESCALER[2:0] Prescaler

These bits select the counter prescaler factor.

These bits are not synchronized.

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

Value	Name	Description
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

### 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”. It also controls whether frequency or PWM waveform generation should be used. How these modes differ can also be seen from “Waveform Output Operations”.

These bits are not synchronized.

**Table 31-7. Waveform Generation Operation**

Value	Name	Operation	Top Value	Waveform Output on Match	Waveform Output on Wraparound
0x0	NFRQ	Normal frequency	PER <sup>(1)</sup> /Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER <sup>(1)</sup> /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.

### Bits 3:2 – MODE[1:0] Timer Counter Mode

These bits select the counter mode.

These bits are not synchronized.

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

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 SYNCBUSY register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

This bit is not enable protected.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

### Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.

Writing a '1' to this bit resets all registers in the TC, except DBGCTRL, to their initial state, and the TC will be disabled.

Writing a '1' 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 SYNCBUSY.SWRST will both be cleared when the reset is complete.

This bit is not enable protected.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

### 31.8.2 Read Request

**Name:** READREQ  
**Offset:** 0x02  
**Reset:** 0x0000

Bit	15	14	13	12	11	10	9	8
	RREQ	RCONT						
Access	W	R/W						
Reset	0	0						

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

Value	Description
0	Continuous synchronization is disabled.
1	Continuous synchronization is enabled.

**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.

### 31.8.3 Control B Clear

**Name:** CTRLBCLR  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to clear bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set register (CTRLBSET).

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

#### Bits 7:6 – CMD[1:0] Command

These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK\_TC clock cycle. When a command has been executed, the CMD bit group will be read back as zero.

Writing 0x0 to these bits has no effect.

Writing a '1' to any of these bits will clear the pending command.

**Table 31-8. Command**

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	-	Reserved

#### Bit 2 – ONESHOT One-Shot on Counter

This bit controls one-shot operation of the TC.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will disable one-shot operation.

Value	Description
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.

#### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the bit and make the counter count up.

---

---

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).



**31.8.4 Control B Set**

**Name:** CTRLBSET  
**Offset:** 0x05  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-synchronized, Write-Synchronized

This register allows the user to set bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Clear register (CTRLBCLR).

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

**Bits 7:6 – CMD[1:0] Command**

These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK\_TC clock cycle. When a command has been executed, the CMD bit group will be read back as zero.

Writing 0x0 to these bits has no effect.

Writing a '1' to any of these bits will clear the pending command.

**Table 31-9. Command**

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	-	Reserved

**Bit 2 – ONESHOT One-Shot on Counter**

This bit controls one-shot operation of the TC.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will disable one-shot operation.

Value	Description
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.

**Bit 0 – DIR Counter Direction**

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the bit and make the counter count up.

---

---

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

### 31.8.5 Control C

**Name:** CTRLC  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-synchronized, Write-Synchronized

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

#### Bits 5,4 – CPTENx Capture Channel x Enable

These bits are used to select the capture or compare operation on channel x.

Writing a '1' to CPTENx enables capture on channel x.

Writing a '0' to CPTENx disables capture on channel x.

#### Bits 1,0 – INVENx Waveform Output x Inversion Enable

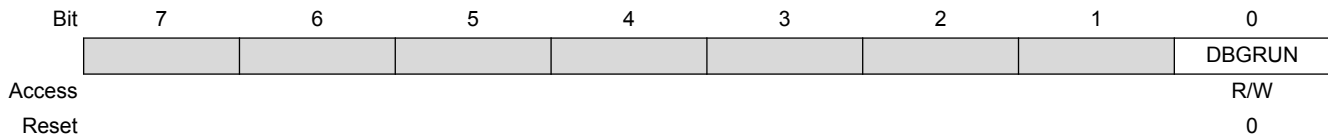
These bits are used to select inversion on the output of channel x.

Writing a '1' to INVENx inverts output from WO[x].

Writing a '0' to INVENx disables inversion of output from WO[x].

### 31.8.6 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### 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.

Value	Description
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.

### 31.8.7 Event Control

**Name:** EVCTRL  
**Offset:** 0x0A  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEOx	MCEOx				OVFEO
Access			R/W	R/W				R/W
Reset			0	0				0
Bit	7	6	5	4	3	2	1	0
			TCEI	TCINV			EVACT[2:0]	
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

#### Bits 13,12 – MCEOx Match or Capture Channel x Event Output Enable [x = 1..0]

These bits enable the generation of an event for every match or capture on channel x.

Value	Description
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.

#### Bit 8 – OVFEO Overflow/Underflow Event Output Enable

This bit enables the Overflow/Underflow event. When enabled, an event will be generated when the counter overflows/underflows.

Value	Description
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.

#### Bit 5 – TCEI TC Event Enable

This bit is used to enable asynchronous input events to the TC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

#### Bit 4 – TCINV TC Inverted Event Input Polarity

This bit inverts the asynchronous input event source.

Value	Description
0	Input event source is not inverted.
1	Input event source is inverted.

#### Bits 2:0 – EVACT[2:0] Event Action

These bits define the event action the TC will perform on an event.

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

31.8.8 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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	6	5	4	3	2	1	0
			MCx	MCx	SYNCRDY		ERR	OVF
Access			R/W	R/W	R/W		R/W	R/W
Reset			0	0	0		0	0

**Bits 5,4 – MCx** Match or Capture Channel x Interrupt Enable [x = 1..0]

Writing a '0' to these bits has no effect.

Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

**Bit 3 – SYNCRDY** Synchronization Ready Interrupt Enable

Writing a '0' 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.

Value	Description
0	The Synchronization Ready interrupt is disabled.
1	The Synchronization Ready interrupt is enabled.

**Bit 1 – ERR** Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

**Bit 0 – OVF** Overflow Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Overflow Interrupt Enable bit, which disables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

### 31.8.9 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x0D  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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).

Bit	7	6	5	4	3	2	1	0
			MCx	MCx	SYNCRDY		ERR	OVF
Access			R/W	R/W	R/W		R/W	R/W
Reset			0	0	0		0	0

**Bits 5,4 – MCx** Match or Capture Channel x Interrupt Enable [x = 1..0]

Writing a '0' to these bits has no effect.

Writing a '1' to MCx will set the corresponding Match or Capture Channel x Interrupt Enable bit, which enables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

**Bit 3 – SYNCRDY** Synchronization Ready Interrupt Enable

Writing a '0' 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.

Value	Description
0	The Synchronization Ready interrupt is disabled.
1	The Synchronization Ready interrupt is enabled.

**Bit 1 – ERR** Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

**Bit 0 – OVF** Overflow Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.



### 31.8.10 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x0E  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
			MCx	MCx	SYNCRDY		ERR	OVF
Access			R/W	R/W	R/W		R/W	R/W
Reset			0	0	0		0	0

#### Bits 5,4 – MCx Match or Capture Channel x [x = 1..0]

This flag is set on a comparison match, or when the corresponding CCx register contains a valid capture value. This flag is set on the next CLK\_TC\_CNT cycle, 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 '1'.

Writing a '0' to one of these bits has no effect.

Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag

In capture operation, this flag is automatically cleared when CCx register is read.

#### Bit 3 – SYNCRDY Synchronization Ready Interrupt Enable

Writing a '0' 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.

Value	Description
0	The Synchronization Ready interrupt is disabled.
1	The Synchronization Ready interrupt is enabled.

#### Bit 1 – ERR Error Interrupt Flag

This flag is set when a new capture occurs on a channel while the corresponding Match or Capture Channel x interrupt flag is set, in which case there is nowhere to store the new capture.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Error interrupt flag.

#### Bit 0 – OVF Overflow Interrupt Flag

This flag is set on the next CLK\_TC\_CNT cycle after an overflow condition occurs, and will generate an interrupt request if INTENCLR.OVF or INTENSET.OVF is '1'.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Overflow interrupt flag.

### 31.8.11 Status

**Name:** STATUS  
**Offset:** 0x0F  
**Reset:** 0x08  
**Property:** -

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

#### 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.

#### Bit 4 – SLAVE Slave Status Flag

This bit is only available in 32-bit mode on the slave TC (i.e., TC1 and/or TC3). The bit is set when the associated master TC (TC0 and TC2, respectively) is set to run in 32-bit mode.

#### Bit 3 – STOP Stop Status Flag

This bit is set when the TC is disabled, on a Stop command, or on an overflow/underflow condition when the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) is '1'.

Value	Description
0	Counter is running.
1	Counter is stopped.

### 31.8.12 Counter Value

### 31.8.12.1 Counter Value, 8-bit Mode

**Name:** COUNT  
**Offset:** 0x10  
**Reset:** 0x00  
**Property:** PAC Write-Protection, 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.

### 31.8.12.2 Counter Value, 16-bit Mode

**Name:** COUNT  
**Offset:** 0x10  
**Reset:** 0x00  
**Property:** PAC Write-Protection, 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.

### 31.8.12.3 Counter Value, 32-bit Mode

**Name:** COUNT  
**Offset:** 0x10  
**Reset:** 0x00  
**Property:** PAC Write-Protection, 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	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 contain the current counter value.

### 31.8.13 Period Value

### 31.8.13.1 Period Value, 8-bit Mode

**Name:** PER  
**Offset:** 0x14  
**Reset:** 0xFF  
**Property:** Write-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	0	0	0	0	0	0	0	1

#### Bits 7:0 – PER[7:0] Period Value

These bits hold the value of the Period Buffer register PERBUF. The value is copied to PER register on UPDATE condition.

### 31.8.14 Compare/Capture

### 31.8.14.1 Channel x Compare/Capture Value, 8-bit Mode

**Name:** CCx  
**Offset:** 0x18+i\*0x1 [i=0..1]  
**Reset:** 0x00  
**Property:** Write-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] Channel x Compare/Capture Value

These bits contain the compare/capture value in 8-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (CTRLA.WAVEGEN), the CC0 register is used as a period register.

### 31.8.14.2 Channel x Compare/Capture Value, 16-bit Mode

**Name:** CCx  
**Offset:** 0x18+i\*0x2 [i=0..1]  
**Reset:** 0x0000  
**Property:** Write-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] Channel x Compare/Capture Value

These bits contain the compare/capture value in 16-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (CTRLA.WAVEGEN), the CC0 register is used as a period register.



### 31.8.14.3 Channel x Compare/Capture Value, 32-bit Mode

**Name:** CCx  
**Offset:** 0x18+i\*0x4 [i=0..1]  
**Reset:** 0x00000000  
**Property:** Write-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] Channel x Compare/Capture Value

These bits contain the compare/capture value in 32-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (CTRLA.WAVEGEN), the CC0 register is used as a period register.

## 32. TCC – Timer/Counter for Control Applications

### 32.1 Overview

The device provides three instances of the Timer/Counter for Control applications (TCC) peripheral, TCC[2:0].

Each TCC instance consists of a counter, a prescaler, compare/capture channels and control logic. The counter can be set to count events or clock pulses. The counter together with the compare/capture channels can be configured to time stamp input events, allowing capture of frequency and pulse-width. It can also perform waveform generation such as frequency generation and pulse-width modulation.

Waveform extensions are featured for motor control, ballast, LED, H-bridge, power converters, and other types of power control applications. They allow for low- and high-side output with optional dead-time insertion. Waveform extensions can also generate a synchronized bit pattern across the waveform output pins. The fault options enable fault protection for safe and deterministic handling, disabling and/or shut down of external drivers.

[Figure 32-1](#) shows all features in TCC.

**Note:** The TCC configurations, such as channel numbers and features, may be reduced for some of the TCC instances.

#### Related Links

[TCC Configurations](#)

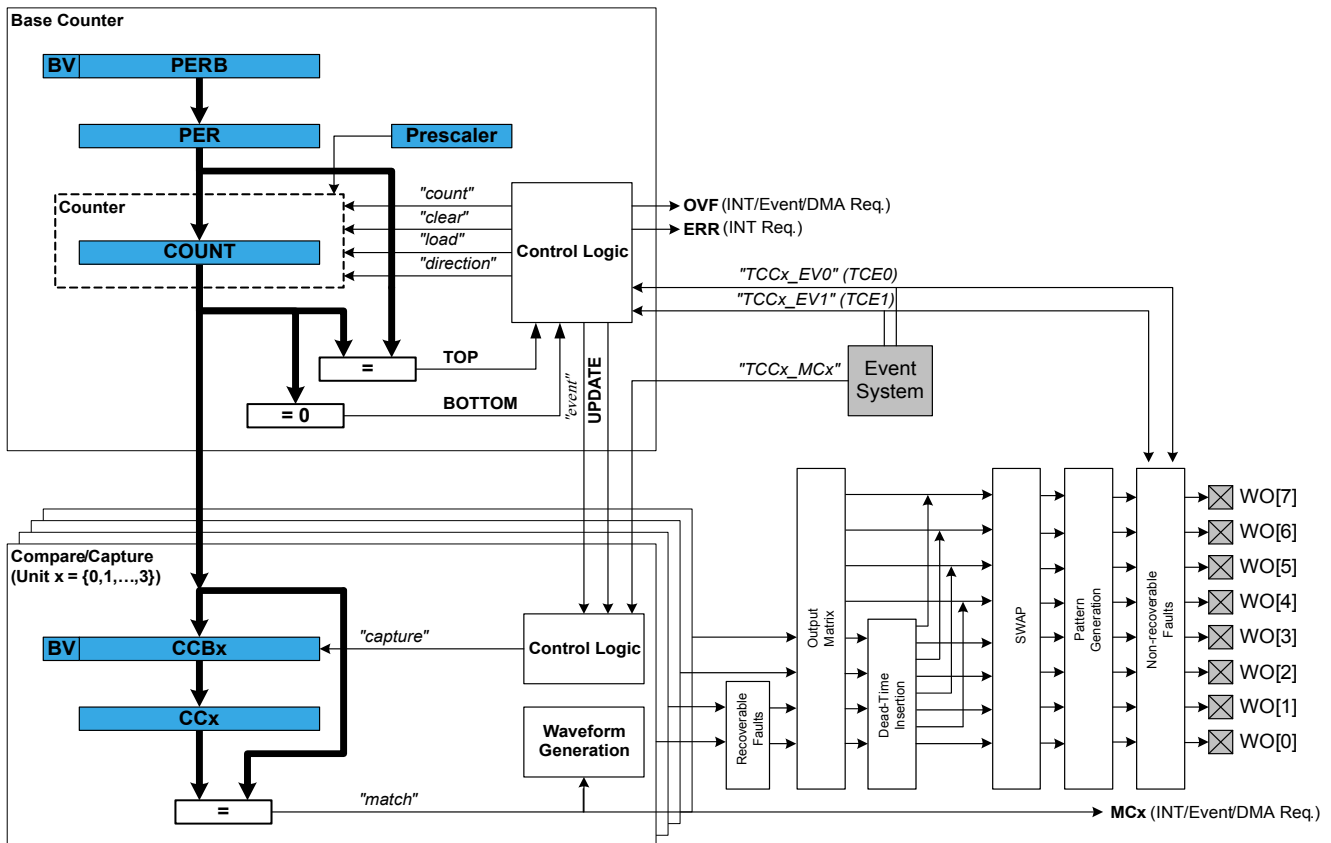
### 32.2 Features

- Up to four compare/capture channels (CC) with:
  - Double buffered period setting
  - Double buffered compare or capture channel
  - Circular buffer on period and compare channel registers
- Waveform generation:
  - Frequency generation
  - Single-slope pulse-width modulation (PWM)
  - Dual-slope pulse-width modulation with half-cycle reload capability
- Input capture:
  - Event capture
  - Frequency capture
  - Pulse-width capture
- Waveform extensions:
  - Configurable distribution of compare channels outputs across port pins
  - Low- and high-side output with programmable dead-time insertion
  - Waveform swap option with double buffer support
  - Pattern generation with double buffer support
  - Dithering support
- Fault protection for safe disabling of drivers:

- Two recoverable fault sources
- Two non-recoverable fault sources
- Debugger can be source of non-recoverable fault
- Input events:
  - Two input events (EVx) for counter
  - One input event (MCx) for each channel
- Output events:
  - Three output events (Count, Re-Trigger and Overflow) available for counter
  - One Compare Match/Input Capture event output for each channel
- Interrupts:
  - Overflow and Re-Trigger interrupt
  - Compare Match/Input Capture interrupt
  - Interrupt on fault detection
- Can be used with DMA and can trigger DMA transactions

### 32.3 Block Diagram

Figure 32-1. Timer/Counter for Control Applications - Block Diagram



### 32.4 Signal Description

Pin Name	Type	Description
TCCx/WO[0]	Digital output	Compare channel 0 waveform output
TCCx/WO[1]	Digital output	Compare channel 1 waveform output
...	...	...
TCCx/WO[WO_NUM-1]	Digital output	Compare channel n waveform output

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

**Related Links**

[I/O Multiplexing and Considerations](#)

### 32.5 Product Dependencies

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

#### 32.5.1 I/O Lines

In order to use the I/O lines of this peripheral, the I/O pins must be configured using the I/O Pin Controller (PORT).

**Related Links**

[PORT: IO Pin Controller](#)

#### 32.5.2 Power Management

This peripheral can continue to operate in any sleep mode where its source clock is running. The interrupts can wake up the device from sleep modes. Events connected to the event system can trigger other operations in the system without exiting sleep modes.

#### 32.5.3 Clocks

The TCC bus clocks (CLK\_TCCx\_APB) can be enabled and disabled in the Power Manager module. The default state of CLK\_TCCx\_APB can be found in the Peripheral Clock Masking section (see the Related Links below).

A generic clock (GCLK\_TCCx) is required to clock the TCC. This clock must be configured and enabled in the generic clock controller before using the TCC. Note that TCC0 and TCC1 share a peripheral clock generator.

The generic clocks (GCLK\_TCCx) are asynchronous to the bus clock (CLK\_TCCx\_APB). Due to this asynchronicity, writing certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) for further details.

**Related Links**

[GCLK - Generic Clock Controller](#)

[Peripheral Clock Masking](#)

#### 32.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). In order to use DMA requests with this peripheral the DMAC must be configured first. Refer to *DMAC – Direct Memory Access Controller* for details.

##### Related Links

[DMAC – Direct Memory Access Controller](#)

#### 32.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. Refer to *Nested Vector Interrupt Controller* for details.

##### Related Links

[Nested Vector Interrupt Controller](#)

#### 32.5.6 Events

The events of this peripheral are connected to the Event System.

##### Related Links

[EVSYS – Event System](#)

#### 32.5.7 Debug Operation

When the CPU is halted in debug mode, this peripheral will halt normal operation. This peripheral can be forced to continue operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

Refer to [DBGCTRL](#) register for details.

#### 32.5.8 Register Access Protection

Registers with write-access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag register (INTFLAG)
- Status register (STATUS)
- Period and Period Buffer registers (PER, PERB)
- Compare/Capture and Compare/Capture Buffer registers (CCx, CCBx)
- Control Waveform register (WAVE)
- Control Waveform Buffer register (WAVEB)
- Pattern Generation Value and Pattern Generation Value Buffer registers (PATT, PATTB)

**Note:** Optional write-protection is indicated by the "PAC Write-Protection" property in the register description.

Write-protection does not apply for accesses through an external debugger.

#### 32.5.9 Analog Connections

Not applicable.

## 32.6 Functional Description

### 32.6.1 Principle of Operation

The following definitions are used throughout the documentation:

**Table 32-1. Timer/Counter for Control Applications - Definitions**

Name	Description
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be the same as Period (PER) or the Compare Channel 0 (CC0) register value depending on the waveform generator mode in <a href="#">Waveform Output Generation Operations</a> .
ZERO	The counter reaches ZERO when it contains all zeroes.
MAX	The counter reaches maximum when it contains all ones.
UPDATE	The timer/counter signals an update when it reaches ZERO or TOP, depending on the direction settings.
Timer	The timer/counter clock control is handled by an internal source.
Counter	The clock control is handled externally (e.g., counting external events).
CC	For compare operations, the CC are referred to as "compare channels." For capture operations, the CC are referred to as "capture channels."

Each TCC instance has up to four compare/capture channels (CCx).

The counter register (COUNT), period registers with buffer (PER and PERB), and compare and capture registers with buffers (CCx and CCBx) are 16- or 24-bit registers, depending on each TCC instance. Each buffer register has a buffer valid (BUFV) flag that indicates when the buffer contains a new value.

Under normal operation, the counter value is continuously compared to the TOP or ZERO value to determine whether the counter has reached TOP or ZERO. In either case, the TCC can generate interrupt requests, request DMA transactions, or generate events for the Event System. In waveform generator mode, these comparisons are used to set the waveform period or pulse width.

A prescaled generic clock (GCLK\_TCCx) and events from the event system can be used to control the counter. The event system is also used as a source to the input capture.

The Recoverable Fault Unit enables event controlled waveforms by acting directly on the generated waveforms of the TCC compare channels output. These events can restart, halt the timer/counter period, shorten the output pulse active time, or disable waveform output as long as the fault condition is present. This can typically be used for current sensing regulation, and zero-crossing and demagnetization re-triggering.

The MCE0 and MCE1 asynchronous event sources are shared with the Recoverable Fault Unit. Only asynchronous events are used internally when fault unit extension is enabled. For further details on how to configure asynchronous events routing, refer to *EVSYS – Event System*.

Recoverable fault sources can be filtered and/or windowed to avoid false triggering, for example from I/O pin glitches, by using digital filtering, input blanking, and qualification options. See also [Recoverable Faults](#).

In order to support applications with different types of motor control, ballast, LED, H-bridge, power converter, and other types of power switching applications, the following independent units are implemented in some of the TCC instances as optional and successive units:

- Recoverable faults and non-recoverable faults
- Output matrix
- Dead-time insertion
- Swap
- Pattern generation

See also [Figure 32-1](#).

The output matrix (OTMX) can distribute and route out the TCC waveform outputs across the port pins in different configurations, each optimized for different application types. The Dead-Time Insertion (DTI) unit splits the four lower OTMX outputs into two non-overlapping signals: the non-inverted low side (LS) and inverted high side (HS) of the waveform output with optional dead-time insertion between LS and HS switching. The SWAP unit can swap the LS and HS pin outputs, and can be used for fast decay motor control.

The pattern generation unit can be used to generate synchronized waveforms with constant logic level on TCC UPDATE conditions. This is useful for easy stepper motor and full bridge control.

The non-recoverable fault module enables event controlled fault protection by acting directly on the generated waveforms of the timer/counter compare channel outputs. When a non-recoverable fault condition is detected, the output waveforms are forced to a preconfigured value that is safe for the application. This is typically used for instant and predictable shut down and disabling high current or voltage drives.

The count event sources (TCE0 and TCE1) are shared with the non-recoverable fault extension. The events can be optionally filtered. If the filter options are not used, the non-recoverable faults provide an immediate asynchronous action on waveform output, even for cases where the clock is not present. For further details on how to configure asynchronous events routing, refer to section *EVSYS – Event System*.

### Related Links

[EVSYS – Event System](#)

## 32.6.2 Basic Operation

### 32.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the TCC is disabled (CTRLA.ENABLE=0):

- Control A (CTRLA) register, except Run Standby (RUNSTDBY), Enable (ENABLE) and Software Reset (SWRST) bits
- Recoverable Fault n Control registers (FCTRLA and FCTRLB)
- Waveform Extension Control register (WEXCTRL)
- Drive Control register (DRVCTRL)
- Event Control register (EVCTRL)

Enable-protected bits in the CTRLA register can be written at the same time as CTRLA.ENABLE is written to '1', but not at the same time as CTRLA.ENABLE is written to '0'. Enable-protection is denoted by the “Enable-Protected” property in the register description.

Before the TCC is enabled, it must be configured as outlined by the following steps:

1. Enable the TCC bus clock (CLK\_TCCx\_APB).
2. If Capture mode is required, enable the channel in capture mode by writing a '1' to the Capture Enable bit in the Control A register (CTRLA.CPTEN).

Optionally, the following configurations can be set before enabling TCC:

1. Select PRESCALER setting in the Control A register (CTRLA.PRESCALER).
2. Select Prescaler Synchronization setting in Control A register (CTRLA.PRESCSYNC).
3. If down-counting operation is desired, write the Counter Direction bit in the Control B Set register (CTRLBSET.DIR) to '1'.
4. Select the Waveform Generation operation in the WAVE register (WAVE.WAVEGEN).
5. Select the Waveform Output Polarity in the WAVE register (WAVE.POL).
6. The waveform output can be inverted for the individual channels using the Waveform Output Invert Enable bit group in the Driver register (DRVCTRL.INVEN).

### 32.6.2.2 Enabling, Disabling, and Resetting

The TCC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The TCC is disabled by writing a zero to CTRLA.ENABLE.

The TCC is reset by writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the TCC, except DBGCTRL, will be reset to their initial state, and the TCC will be disabled. Refer to Control A (CTRLA) register for details.

The TCC should be disabled before the TCC is reset to avoid undefined behavior.

### 32.6.2.3 Prescaler Selection

The GCLK\_TCCx clock is fed into the internal prescaler.

The prescaler consists of a counter that counts up to the selected prescaler value, whereupon the output of the prescaler toggles.

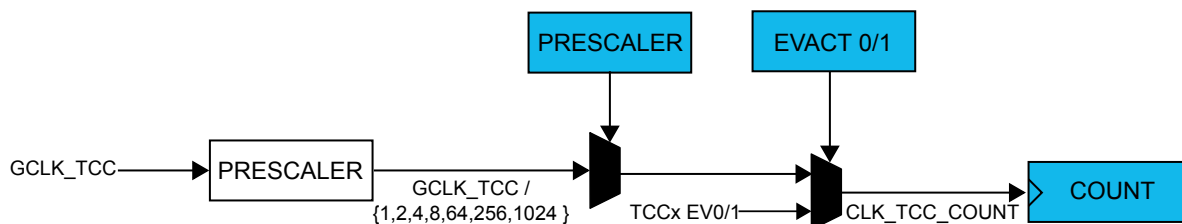
If the prescaler value is higher than one, the counter update condition can be optionally executed on the next GCLK\_TCC clock pulse or the next prescaled clock pulse. For further details, refer to the Prescaler (CTRLA.PRESCALER) and Counter Synchronization (CTRLA.PRESYNC) descriptions.

Prescaler outputs 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).

**Note:** When counting events, the prescaler is bypassed.

The joint stream of prescaler ticks and event action ticks is called CLK\_TCC\_COUNT.

**Figure 32-2. Prescaler**



### 32.6.2.4 Counter Operation

Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each TCC clock input (CLK\_TCC\_COUNT). A counter clear or reload mark the end of current counter cycle and the start of a new one.

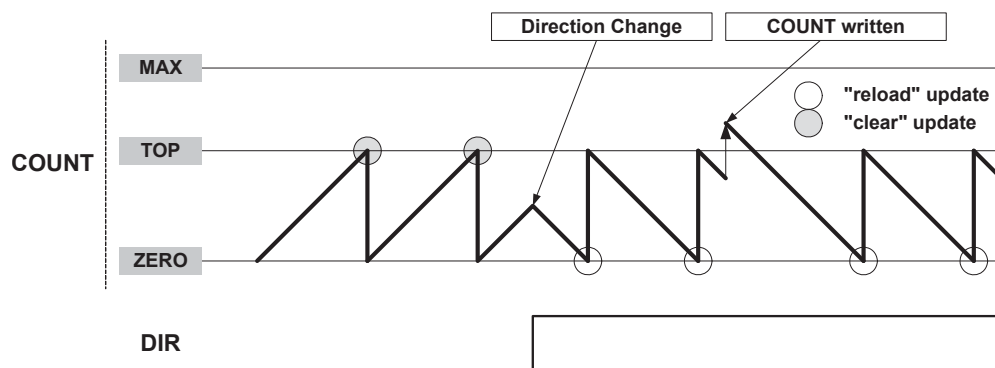


The counting direction is set by the Direction bit in the Control B register (CTRLB.DIR). If the bit is zero, it's counting up and one if counting down.

The counter will count up or down for each tick (clock or event) until it reaches TOP or ZERO. When it's counting up and TOP is reached, the counter will be set to zero at the next tick (overflow) and the Overflow Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF) will be set. When down-counting, the counter is reloaded with the TOP value when ZERO is reached (underflow), and INTFLAG.OVF is set.

INTFLAG.OVF can be used to trigger an interrupt, a DMA request, or an event. An overflow/underflow occurrence (i.e. a compare match with TOP/ZERO) will stop counting if the One-Shot bit in the Control B register is set (CTRLBSET.ONESHOT). The One-Shot feature is explained in the [Additional Features](#) section.

**Figure 32-3. Counter Operation**



It is possible to change the counter value (by writing directly in the COUNT register) even when the counter is running. The COUNT value will always be ZERO or TOP, depending on direction set by CTRLBSET.DIR or CTRLBCLR.DIR, when starting the TCC, unless a different value has been written to it, or the TCC has been stopped at a value other than ZERO. The write access has higher priority than count, clear, or reload. The direction of the counter can also be changed during normal operation. See also [Figure 32-3](#).

### Stop Command

A stop command can be issued from software by using TCC Command bits in Control B Set register (CTRLBSET.CMD=0x2, STOP).

### Pause Event Action

A pause command can be issued when the stop event action is configured in the Input Event Action 1 bits in Event Control register (EVCTRL.EVACT1=0x3, STOP).

### Re-Trigger Command and Event Action

A re-trigger command can be issued from software by using TCC Command bits in Control B Set register (CTRLBSET.CMD=0x1, RETRIGGER), or from event when the re-trigger event action is configured in the Input Event 0/1 Action bits in Event Control register (EVCTRL.EVACTn=0x1, RETRIGGER).

When the command is detected during counting operation, the counter will be reloaded or cleared, depending on the counting direction (CTRLBSET.DIR or CTRLBCLR.DIR). The Re-Trigger bit in the Interrupt Flag Status and Clear register will be set (INTFLAG.TRG). It is also possible to generate an event by writing a '1' to the Re-Trigger Event Output Enable bit in the Event Control register (EVCTRL.TRGEO). If the re-trigger command is detected when the counter is stopped, the counter will resume counting operation from the value in COUNT.

**Note:**

When a re-trigger event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACTn=0x1, RETRIGGER), enabling the counter will not start the counter. The counter will start on the next incoming event and restart on corresponding following event.

**Start Event Action**

The start action can be selected in the Event Control register (EVCTRL.EVACT0=0x3, START) and can start the counting operation when previously stopped. The event has no effect if the counter is already counting. When the module is enabled, the counter operation starts when the event is received or when a re-trigger software command is applied.

**Note:**

When a start event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACT0=0x3, START), enabling the counter will not start the counter. The counter will start on the next incoming event, but it will not restart on subsequent events.

**Count Event Action**

The TCC can count events. When an event is received, the counter increases or decreases the value, depending on direction settings (CTRLBSET.DIR or CTRLBCLR.DIR).

The count event action is selected by the Event Action 0 bit group in the Event Control register (EVCTRL.EVACT0=0x5, COUNT).

**Direction Event Action**

The direction event action can be selected in the Event Control register (EVCTRL.EVACT1=0x2, DIR). When this event is used, the asynchronous event path specified in the event system must be configured or selected. The direction event action can be used to control the direction of the counter operation, depending on external events level. When received, the event level overrides the Direction settings (CTRLBSET.DIR or CTRLBCLR.DIR) and the direction bit value is updated accordingly.

**Increment Event Action**

The increment event action can be selected in the Event Control register (EVCTRL.EVACT0=0x4, INC) and can change the counter state when an event is received. When the TCE0 event (TCCx\_EV0) is received, the counter increments, whatever the direction setting (CTRLBSET.DIR or CTRLBCLR.DIR) is.

**Decrement Event Action**

The decrement event action can be selected in the Event Control register (EVCTRL.EVACT1=0x4, DEC) and can change the counter state when an event is received. When the TCE1 (TCCx\_EV1) event is received, the counter decrements, whatever the direction setting (CTRLBSET.DIR or CTRLBCLR.DIR) is.

**Non-recoverable Fault Event Action**

Non-recoverable fault actions can be selected in the Event Control register (EVCTRL.EVACTn=0x7, FAULT). When received, the counter will be stopped and the output of the compare channels is overridden according to the Driver Control register settings (DRVCTRL.NREx and DRVCTRL.NRVx). TCE0 and TCE1 must be configured as asynchronous events.

**Event Action Off**

If the event action is disabled (EVCTRL.EVACTn=0x0, OFF), enabling the counter will also start the counter.

**Related Links**

### One-Shot Operation

#### 32.6.2.5 Compare Operations

By default, the Compare/Capture channel is configured for compare operations. To perform capture operations, it must be re-configured.

When using the TCC with the Compare/Capture Value registers (CCx) for compare operations, the counter value is continuously compared to the values in the CCx registers. This can be used for timer or for waveform operation.

The Channel x Compare/Capture Buffer Value (CCBx) registers provide double buffer capability. The double buffering synchronizes the update of the CCx register with the buffer value at the UPDATE condition or a force update command (CTRLBSET.CMD=0x3, UPDATE). For further details, refer to [Double Buffering](#). The synchronization prevents the occurrence of odd-length, non-symmetrical pulses and ensures glitch-free output.

#### Waveform Output Generation Operations

The compare channels can be used for waveform generation on output port pins. To make the waveform available on the connected pin, the following requirements must be fulfilled:

1. Choose a waveform generation mode in the Waveform Generation Operation bit in Waveform register (WAVE.WAVEGEN).
2. Optionally invert the waveform output WO[x] by writing the corresponding Waveform Output x Inversion bit in the Driver Control register (DRVCTRL.INVENx).
3. Configure the pins with the I/O Pin Controller. Refer to *PORT - I/O Pin Controller* for details.

The counter value is continuously compared with each CCx value. On a comparison match, the Match or Capture Channel x bit in the Interrupt Flag Status and Clear register (INTFLAG.MCx) will be set on the next zero-to-one transition of CLK\_TCC\_COUNT (see Normal Frequency Operation). An interrupt and/or event can be generated on the same condition if Match/Capture occurs, i.e. INTENSET.MCx and/or EVCTRL.MCEOx is '1'. Both interrupt and event can be generated simultaneously. The same condition generates a DMA request.

There are seven waveform configurations for the Waveform Generation Operation bit group in the Waveform register (WAVE.WAVEGEN). This will influence how the waveform is generated and impose restrictions on the top value. The configurations are:

- Normal Frequency (NFRQ)
- Match Frequency (MFRQ)
- Normal Pulse-Width Modulation (NPWM)
- Dual-slope, interrupt/event at TOP (DSTOP)
- Dual-slope, interrupt/event at ZERO (DSBOTTOM)
- Dual-slope, interrupt/event at Top and ZERO (DSBOTH)
- Dual-slope, critical interrupt/event at ZERO (DSCRITICAL)

When using MFRQ configuration, the TOP value is defined by the CC0 register value. For the other waveform operations, the TOP value is defined by the Period (PER) register value.

For dual-slope waveform operations, the update time occurs when the counter reaches ZERO. For the other waveforms generation modes, the update time occurs on counter wraparound, on overflow, underflow, or re-trigger.

The table below shows the update counter and overflow event/interrupt generation conditions in different operation modes.

**Table 32-2. Counter Update and Overflow Event/interrupt Conditions**

Name	Operation	TOP	Update	Output Waveform		OVFIF/Event	
				On Match	On Update	Up	Down
NFRQ	Normal Frequency	PER	TOP/ ZERO	Toggle	Stable	TOP	ZERO
MFRQ	Match Frequency	CC0	TOP/ ZERO	Toggle	Stable	TOP	ZERO
NPWM	Single-slope PWM	PER	TOP/ ZERO	See section 'Output Polarity' below		TOP	ZERO
DSCRITICAL	Dual-slope PWM	PER	ZERO			-	ZERO
DSBOTTOM	Dual-slope PWM	PER	ZERO			-	ZERO
DSBOTH	Dual-slope PWM	PER	TOP <sup>(1)</sup> & ZERO			TOP	ZERO
DSTOP	Dual-slope PWM	PER	ZERO			TOP	-

1. The UPDATE condition on TOP only will occur when circular buffer is enabled for the channel.

### Related Links

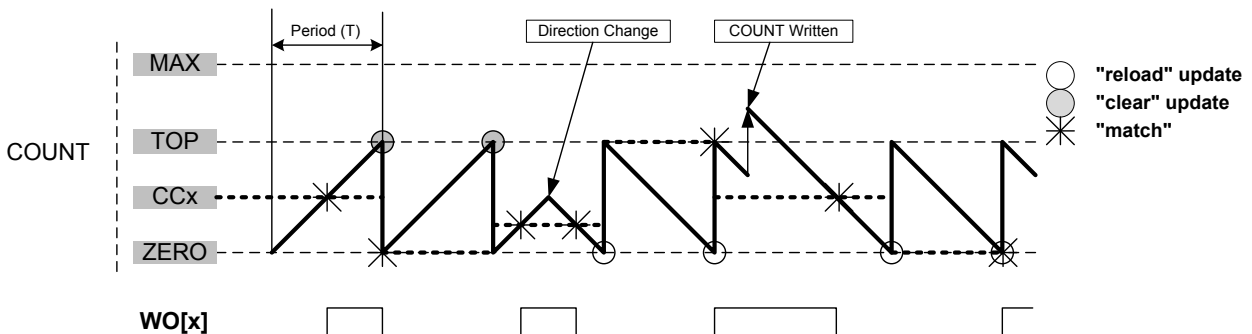
[Circular Buffer](#)

[PORT: IO Pin Controller](#)

### Normal Frequency (NFRQ)

For Normal Frequency generation, the period time (T) is controlled by the period register (PER). The waveform generation output (WO[x]) is toggled on each compare match between COUNT and CCx, and the corresponding Match or Capture Channel x Interrupt Flag (EVCTRL.MCEOx) will be set.

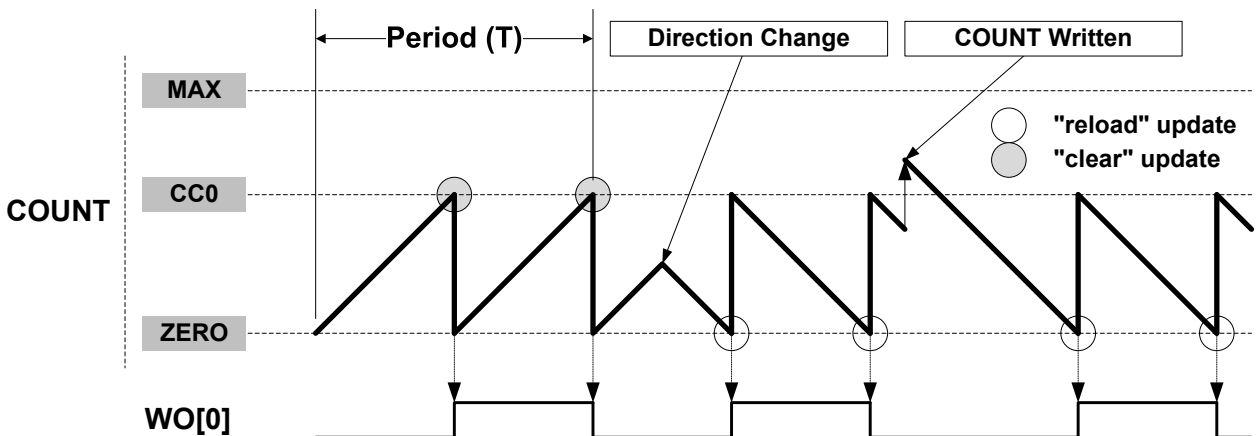
**Figure 32-4. Normal Frequency Operation**



### Match Frequency (MFRQ)

For Match Frequency generation, the period time (T) is controlled by CC0 register instead of PER. WO[0] toggles on each update condition.

Figure 32-5. Match Frequency Operation

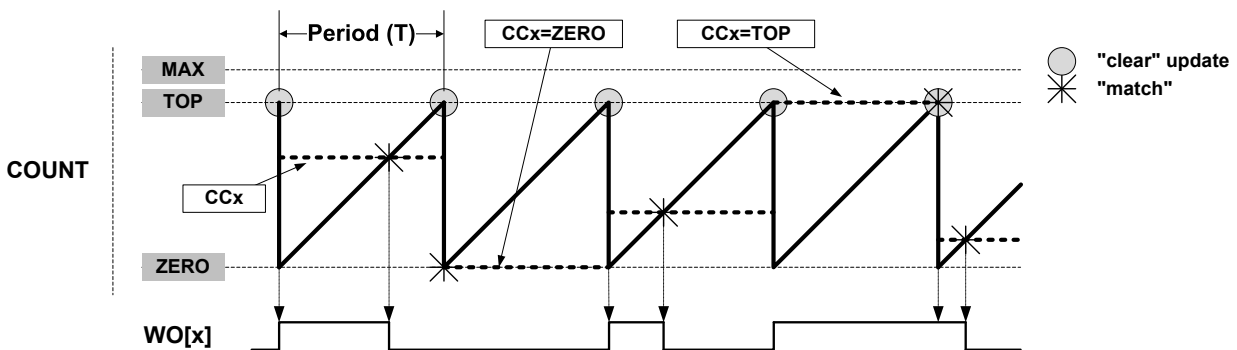


**Normal Pulse-Width Modulation (NPWM)**  
 NPWM uses single-slope PWM generation.

**Single-Slope PWM Operation**

For single-slope PWM generation, the period time (T) is controlled by Top value, and CCx controls the duty cycle of the generated waveform output. When up-counting, the WO[x] is set at start or compare match between the COUNT and TOP values, and cleared on compare match between COUNT and CCx register values. When down-counting, the WO[x] is cleared at start or compare match between the COUNT and ZERO values, and set on compare match between COUNT and CCx register values.

Figure 32-6. Single-Slope PWM Operation



The following equation calculates the exact resolution for a single-slope PWM ( $R_{PWM\_SS}$ ) waveform:

$$R_{PWM\_SS} = \frac{\log(TOP+1)}{\log(2)}$$

The PWM frequency depends on the Period register value (PER) and the peripheral clock frequency ( $f_{GCLK\_TCC}$ ), and can be calculated by the following equation:

$$f_{PWM\_SS} = \frac{f_{GCLK\_TCC}}{N(TOP+1)}$$

Where N represents the prescaler divider used (1, 2, 4, 8, 16, 64, 256, 1024).

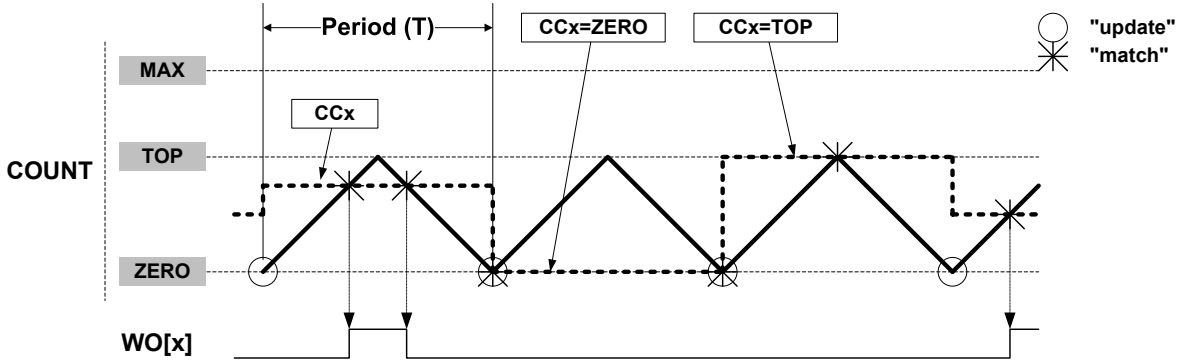
**Dual-Slope PWM Generation**

For dual-slope PWM generation, the period setting (TOP) is controlled by PER, while CCx control the duty cycle of the generated waveform output. The figure below shows how the counter repeatedly counts from ZERO to PER and then from PER to ZERO. The waveform generator output is set on compare

match when up-counting, and cleared on compare match when down-counting. An interrupt and/or event is generated on TOP (when counting upwards) and/or ZERO (when counting up or down).

In DSBOTH operation, the circular buffer must be enabled to enable the update condition on TOP.

**Figure 32-7. Dual-Slope Pulse Width Modulation**



Using dual-slope PWM results in a lower maximum operation frequency compared to single-slope PWM generation. The period (TOP) defines the PWM resolution. The minimum resolution is 1 bit (TOP=0x00000001).

The following equation calculates the exact resolution for dual-slope PWM ( $R_{P_{PWM\_DS}}$ ):

$$R_{P_{PWM\_DS}} = \frac{\log(PER+1)}{\log(2)}$$

The PWM frequency  $f_{P_{PWM\_DS}}$  depends on the period setting (TOP) and the peripheral clock frequency  $f_{GCLK\_TCC}$ , and can be calculated by the following equation:

$$f_{P_{PWM\_DS}} = \frac{f_{GCLK\_TCC}}{2N \cdot PER}$$

$N$  represents the prescaler divider used. The waveform generated will have a maximum frequency of half of the TCC clock frequency ( $f_{GCLK\_TCC}$ ) when TOP is set to 0x00000001 and no prescaling is used.

The pulse width ( $P_{P_{PWM\_DS}}$ ) depends on the compare channel (CCx) register value and the peripheral clock frequency ( $f_{GCLK\_TCC}$ ), and can be calculated by the following equation:

$$P_{P_{PWM\_DS}} = \frac{2N \cdot (TOP - CCx)}{f_{GCLK\_TCC}}$$

$N$  represents the prescaler divider used.

**Note:** In DSTOP, DSBOTTOM and DSBOTH operation, when TOP is lower than MAX/2, the CCx MSB bit defines the ramp on which the CCx Match interrupt or event is generated. (Rising if CCx[MSB] = 0, falling if CCx[MSB] = 1.)

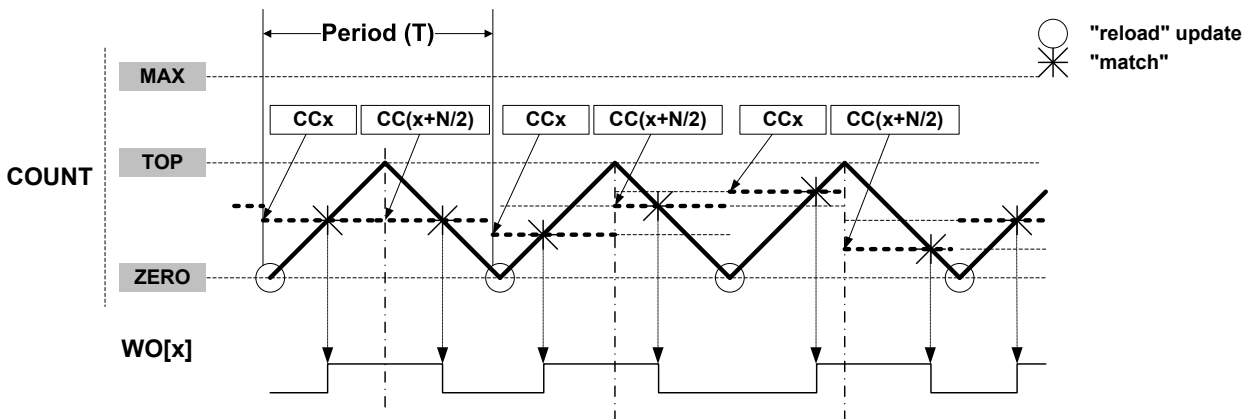
**Related Links**

[Circular Buffer](#)

**Dual-Slope Critical PWM Generation**

Critical mode generation allows generation of non-aligned centered pulses. In this mode, the period time is controlled by PER while CCx control the generated waveform output edge during up-counting and CC(x+CC\_NUM/2) control the generated waveform output edge during down-counting.

Figure 32-8. Dual-Slope Critical Pulse Width Modulation (N=CC\_NUM)



### Output Polarity

The polarity (WAVE.POLx) is available in all waveform output generation. In single-slope and dual-slope PWM operation, it is possible to invert the pulse edge alignment individually on start or end of a PWM cycle for each compare channels. The table below shows the waveform output set/clear conditions, depending on the settings of timer/counter, direction, and polarity.

Table 32-3. Waveform Generation Set/Clear Conditions

Waveform Generation operation	DIR	POLx	Waveform Generation Output Update	
			Set	Clear
Single-Slope PWM	0	0	Timer/counter matches TOP	Timer/counter matches CCx
		1	Timer/counter matches CC	Timer/counter matches TOP
	1	0	Timer/counter matches CC	Timer/counter matches ZERO
		1	Timer/counter matches ZERO	Timer/counter matches CC
Dual-Slope PWM	x	0	Timer/counter matches CC when counting up	Timer/counter matches CC when counting down
		1	Timer/counter matches CC when counting down	Timer/counter matches CC when counting up

In Normal and Match Frequency, the WAVE.POLx value represents the initial state of the waveform output.

### 32.6.2.6 Double Buffering

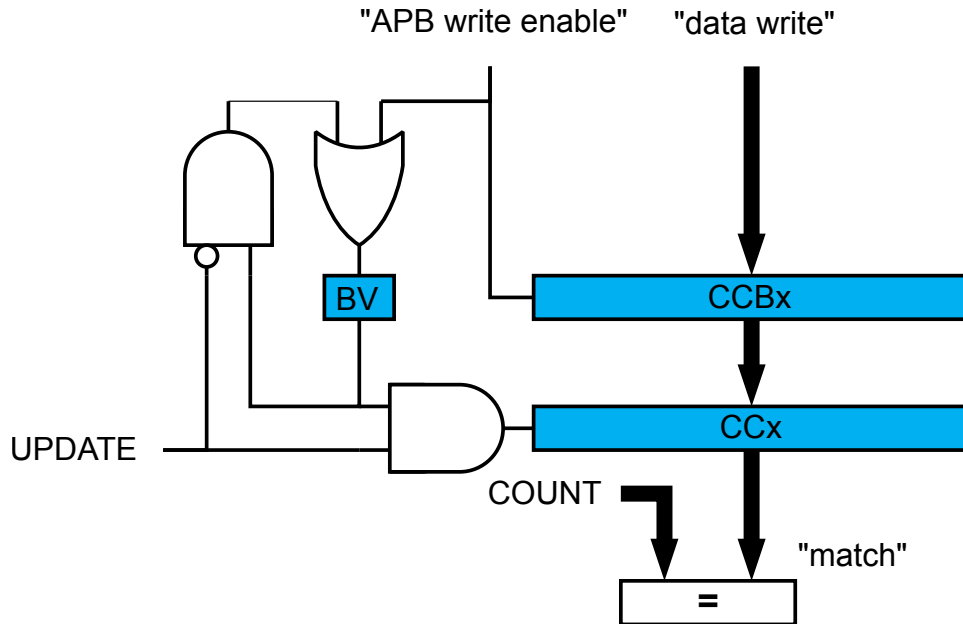
The Pattern (PATT), Waveform (WAVE), Period (PER) and Compare Channels (CCx) registers are all double buffered. Each buffer register has a buffer valid (PATTBV, WAVEBV, PERBV and CCBVx) bit in the STATUS register, which indicates that the buffer register contains a valid value that can be copied into the corresponding register. .

When the buffer valid flag bit in the STATUS register is '1' and the Lock Update bit in the CTRLB register is set to '0', (writing CTRLBCLR.LUPD to '1'), double buffering is enabled: the data from buffer registers will be copied into the corresponding register under hardware UPDATE conditions, then the buffer valid flags bit in the STATUS register are automatically cleared by hardware.

**Note:** Software update command (CTRLBSET.CMD=0x3) act independently of LUPD value.

A compare register is double buffered as in the following figure.

Figure 32-9. Compare Channel Double Buffering



Both the registers (PATT/WAVE/PER/CCx) and corresponding buffer registers (PATTB/WAVEBV/PERB/CCBx) are available in the I/O register map, and the double buffering feature is not mandatory. The double buffering is disabled by writing a '1' to CTRLSET.LUPD.

**Note:** In NFRQ, MFRQ or PWM down-counting counter mode (CTRLBSET.DIR=1), when double buffering is enabled (CTRLBCLR.LUPD=1), PERB register is continuously copied into the PER independently of update conditions.

**Changing the Period**

The counter period can be changed by writing a new Top value to the Period register (PER or CC0, depending on the waveform generation mode), any period update on registers (PER or CCx) is effective after the synchronization delay, whatever double buffering enabling is.

Figure 32-10. Unbuffered Single-Slope Up-Counting Operation

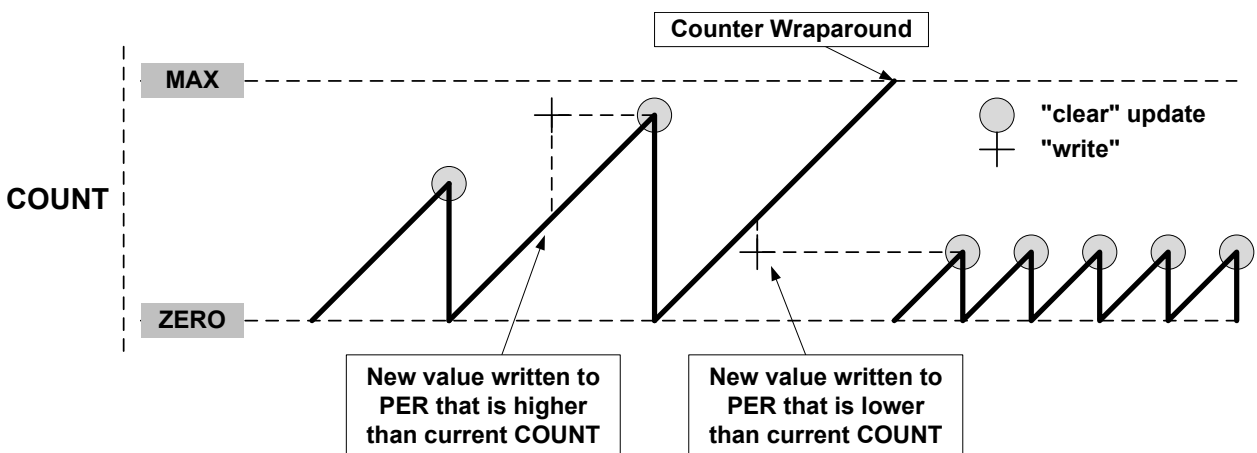
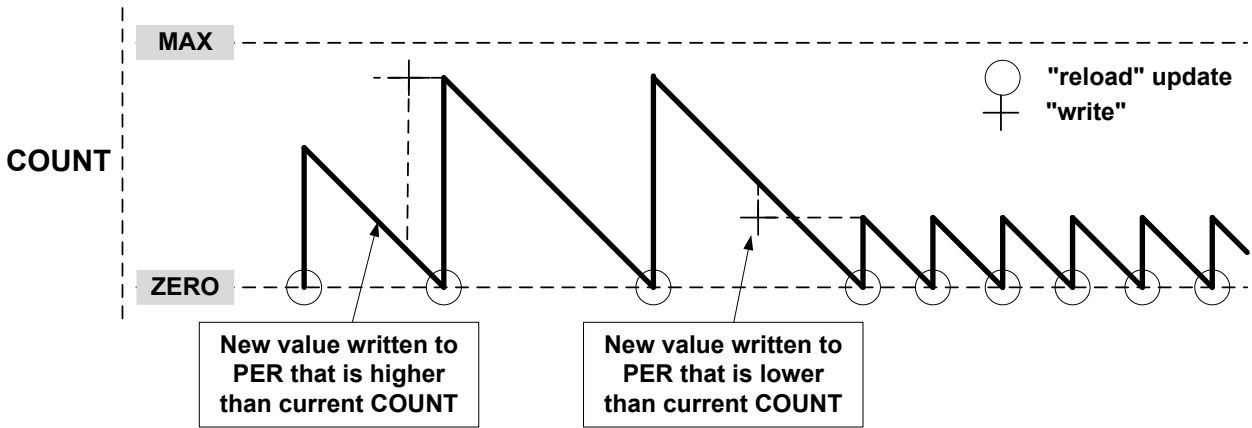


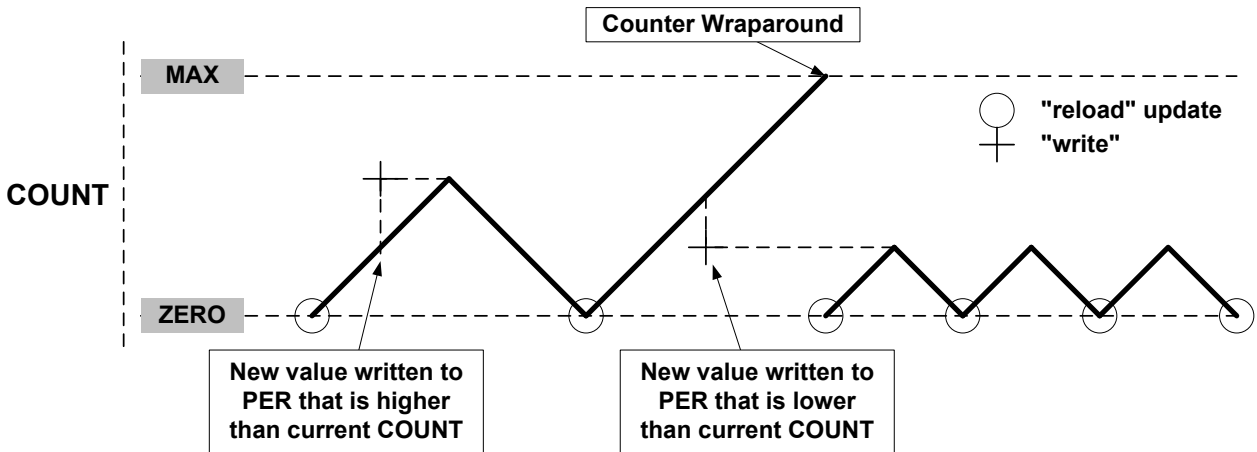


Figure 32-11. Unbuffered Single-Slope Down-Counting Operation



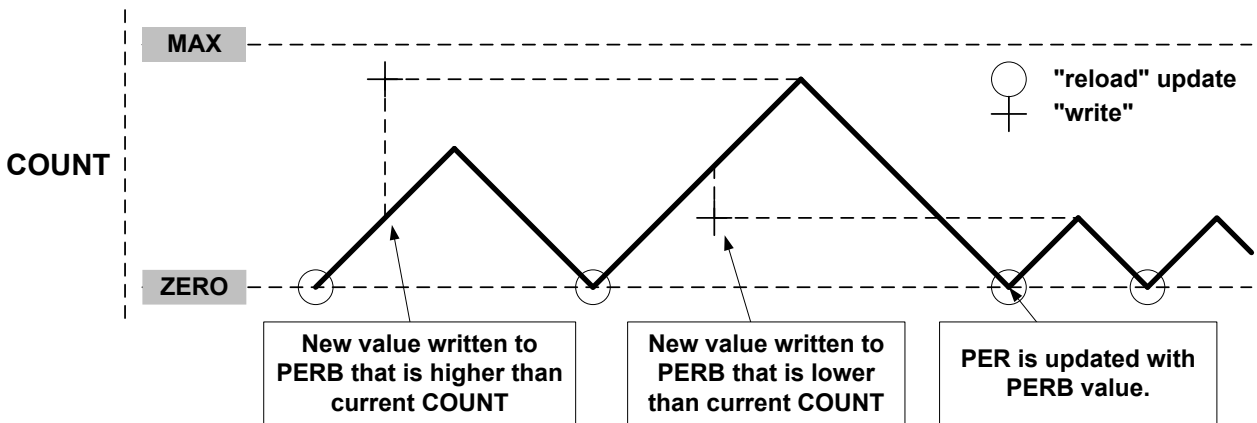
A counter wraparound can occur in any operation mode when up-counting without buffering, see [Figure 32-10](#). COUNT and TOP are continuously compared, so when a new value that is lower than the current COUNT is written to TOP, COUNT will wrap before a compare match.

Figure 32-12. Unbuffered Dual-Slope Operation



When double buffering is used, the buffer can be written at any time and the counter will still maintain correct operation. The period register is always updated on the update condition, as shown in [Figure 32-13](#). This prevents wraparound and the generation of odd waveforms.

Figure 32-13. Changing the Period Using Buffering



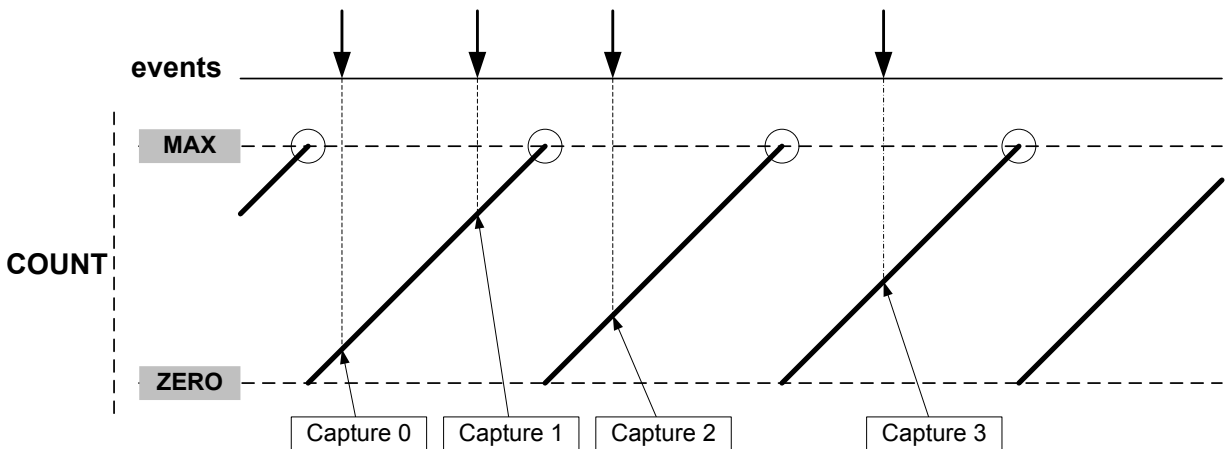
32.6.2.7 Capture Operations

To enable and use capture operations, the Match or Capture Channel x Event Input Enable bit in the Event Control register (EVCTRL.MCEIx) must be written to '1'. The capture channels to be used must also be enabled in the Capture Channel x Enable bit in the Control A register (CTRLA.CPTENx) before capturing can be performed.

Event Capture Action

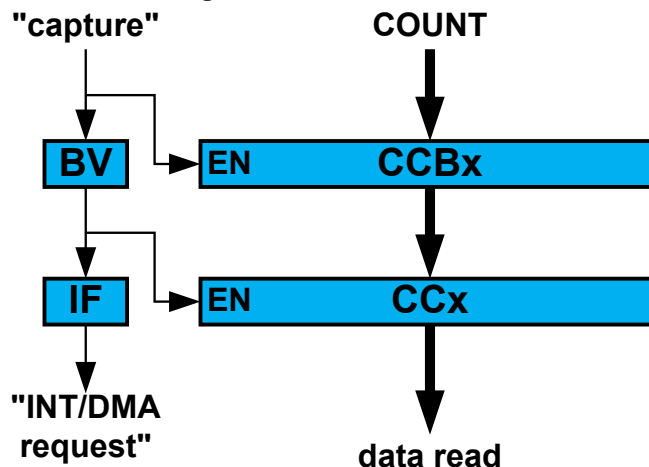
The compare/capture channels can be used as input capture channels to capture events from the Event System, and give them a timestamp. The following figure shows four capture events for one capture channel.

Figure 32-14. Input Capture Timing



For input capture, the buffer register and the corresponding CCx act like a FIFO. When CCx is empty or read, any content in CCBx is transferred to CCx. The buffer valid flag is passed to set the CCx interrupt flag (IF) and generate the optional interrupt, event or DMA request. CCBx register value can't be read, all captured data must be read from CCx register.

Figure 32-15. Capture Double Buffering



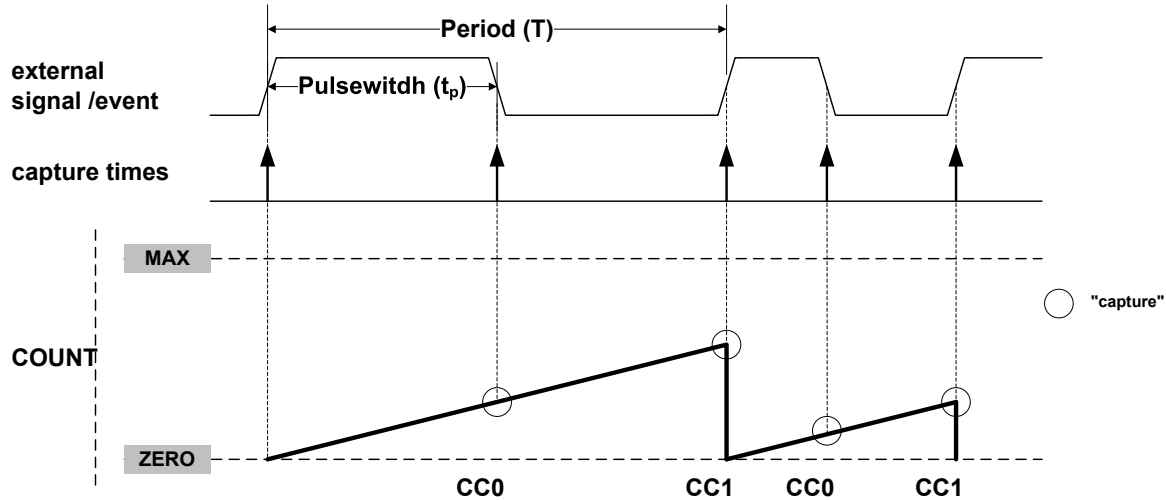
The TCC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Buffer Valid flag (STATUS.CCBV) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

Period and Pulse-Width (PPW) Capture Action

The TCC can perform two input captures and restart the counter on one of the edges. This enables the TCC to measure the pulse-width and period and to characterize the frequency  $f$  and  $dutyCycle$  of an input signal:

$$f = \frac{1}{T} \quad , \quad dutyCycle = \frac{t_p}{T}$$

Figure 32-16. PWP Capture



Selecting PWP or PPW in the Timer/Counter Event Input 1 Action bit group in the Event Control register (EVCTRL.EVACT1) enables the TCC to perform one capture action on the rising edge and the other one on the falling edge. When using PPW (period and pulse-width) event action, period  $T$  will be captured into CC0 and the pulse-width  $t_p$  into CC1. The PWP (Pulse-width and Period) event action offers the same functionality, but  $T$  will be captured into CC1 and  $t_p$  into CC0.

The Timer/Counter Event x Invert Enable bit in Event Control register (EVCTRL.TCEINVx) is used for event source x to select whether the wraparound should occur on the rising edge or the falling edge. If EVCTRL.TCEINVx=1, the wraparound will happen on the falling edge.

The corresponding capture is done only if the channel is enabled in capture mode (CTRLA.CPTENx=1). If not, the capture action will be ignored and the channel will be enabled in compare mode of operation. When only one of these channel is required, the other channel can be used for other purposes.

The TCC can detect capture overflow of the input capture channels: When a new capture event is detected while the INTFLAG.MCx is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

**Note:** When up-counting (CTRLBSET.DIR=0), counter values lower than 1 cannot be captured in Capture Minimum mode (FCTRLn.CAPTURE=CAPTMIN). To capture the full range including value 0, the TCC must be configured in down-counting mode (CTRLBSET.DIR=0).

**Note:** In dual-slope PWM operation, and when TOP is lower than MAX/2, the CCx MSB captures the CTRLB.DIR state to identify the ramp on which the capture has been done. For rising ramps CCx[MSB] is zero, for falling ramps CCx[MSB]=1.

**32.6.3 Additional Features**

**32.6.3.1 One-Shot Operation**

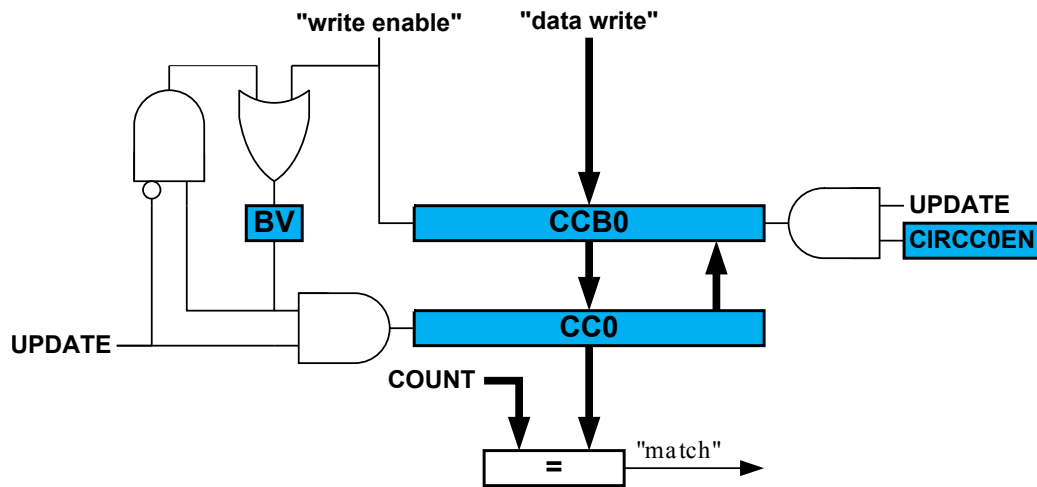
When one-shot is enabled, the counter automatically stops on the next counter overflow or underflow condition. When the counter is stopped, the Stop bit in the Status register (STATUS.STOP) is set and the waveform outputs are set to the value defined by DRVCTRL.NREx and DRVCTRL.NRVx.

One-shot operation can be enabled by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) and disabled by writing a '1' to CTRLBCLR.ONESHOT. When enabled, the TCC will count until an overflow or underflow occurs and stop counting. The one-shot operation can be restarted by a re-trigger software command, a re-trigger event or a start event. When the counter restarts its operation, STATUS.STOP is automatically cleared.

**32.6.3.2 Circular Buffer**

The Period register (PER) and the compare channels register (CC0 to CC3) support circular buffer operation. When circular buffer operation is enabled, the PER or CCx values are copied into the corresponding buffer registers at each update condition. Circular buffering is dedicated to RAMP2, RAMP2A, and DSBOTH operations.

**Figure 32-17. Circular Buffer on Channel 0**



**32.6.3.3 Dithering Operation**

The TCC supports dithering on Pulse-width or Period on a 16, 32 or 64 PWM cycles frame.

Dithering consists in adding some extra clock cycles in a frame of several PWM cycles, and can improve the accuracy of the *average* output pulse width and period. The extra clock cycles are added on some of the compare match signals, one at a time, through a "blue noise" process that minimizes the flickering on the resulting dither patterns.

Dithering is enabled by writing the corresponding configuration in the Enhanced Resolution bits in CTRLA register (CTRLA.RESOLUTION):

- DITH4 enable dithering every 16 PWM frames
- DITH5 enable dithering every 32 PWM frames
- DITH6 enable dithering every 64 PWM frames

The DITHERCY bits of COUNT, PER and CCx define the number of extra cycles to add into the frame (DITHERCY bits from the respective COUNT, PER or CCx registers). The remaining bits of COUNT, PER, CCx define the compare value itself.

The pseudo code, giving the extra cycles insertion regarding the cycle is:

```
int extra_cycle(resolution, dithercy, cycle){
    int MASK;
    int value
    switch (resolution){
        DITH4: MASK = 0x0f;
        DITH5: MASK = 0x1f;
        DITH6: MASK = 0x3f;
    }
    value = cycle * dithercy;
    if ((MASK & value) + dithercy > MASK)
        return 1;
    return 0;
}
```

### Dithering on Period

Writing DITHERCY in PER will lead to an average PWM period configured by the following formulas.

DITH4 mode:

$$PwmPeriod = \left( \frac{DITHERCY}{16} + PER \right) \left( \frac{1}{f_{GCLK\_TCC}} \right)$$

**Note:** If DITH4 mode is enabled, the last 4 significant bits from PER/CCx or COUNT register correspond to the DITHERCY value, rest of the bits corresponds to PER/CCx or COUNT value.

DITH5 mode:

$$PwmPeriod = \left( \frac{DITHERCY}{32} + PER \right) \left( \frac{1}{f_{GCLK\_TCC}} \right)$$

DITH6 mode:

$$PwmPeriod = \left( \frac{DITHERCY}{64} + PER \right) \left( \frac{1}{f_{GCLK\_TCC}} \right)$$

### Dithering on Pulse Width

Writing DITHERCY in CCx will lead to an average PWM pulse width configured by the following formula.

DITH4 mode:

$$PwmPulseWidth = \left( \frac{DITHERCY}{16} + CCx \right) \left( \frac{1}{f_{GCLK\_TCC}} \right)$$

DITH5 mode:

$$PwmPulseWidth = \left( \frac{DITHERCY}{32} + CCx \right) \left( \frac{1}{f_{GCLK\_TCC}} \right)$$

DITH6 mode:

$$PwmPulseWidth = \left( \frac{DITHERCY}{64} + CCx \right) \left( \frac{1}{f_{GCLK\_TCC}} \right)$$

**Note:** The PWM period will remain static in this case.

#### 32.6.3.4 Ramp Operations

Three ramp operation modes are supported. All of them require the timer/counter running in single-slope PWM generation. The ramp mode is selected by writing to the Ramp Mode bits in the Waveform Control register (WAVE.RAMP).

### RAMP1 Operation

This is the default PWM operation, described in [Single-Slope PWM Generation](#).

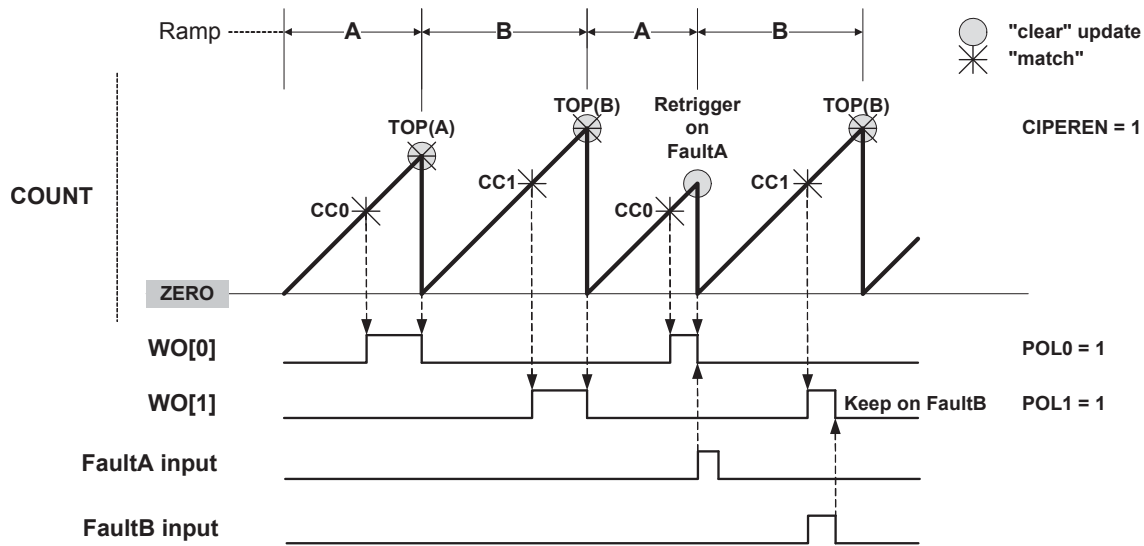
### RAMP2 Operation

These operation modes are dedicated for power factor correction (PFC), Half-Bridge and Push-Pull SMPS topologies, where two consecutive timer/counter cycles are interleaved, see [Figure 32-18](#). In cycle A, odd channel output is disabled, and in cycle B, even channel output is disabled. The ramp index changes after each update, but can be software modified using the Ramp index command bits in Control B Set register (CTRLBSET.IDXCMD).

### Standard RAMP2 (RAMP2) Operation

Ramp A and B periods are controlled by the PER register value. The PER value can be different on each ramp by the Circular Period buffer option in the Wave register (WAVE.CIPEREN=1). This mode uses a two-channel TCC to generate two output signals, or one output signal with another CC channel enabled in capture mode.

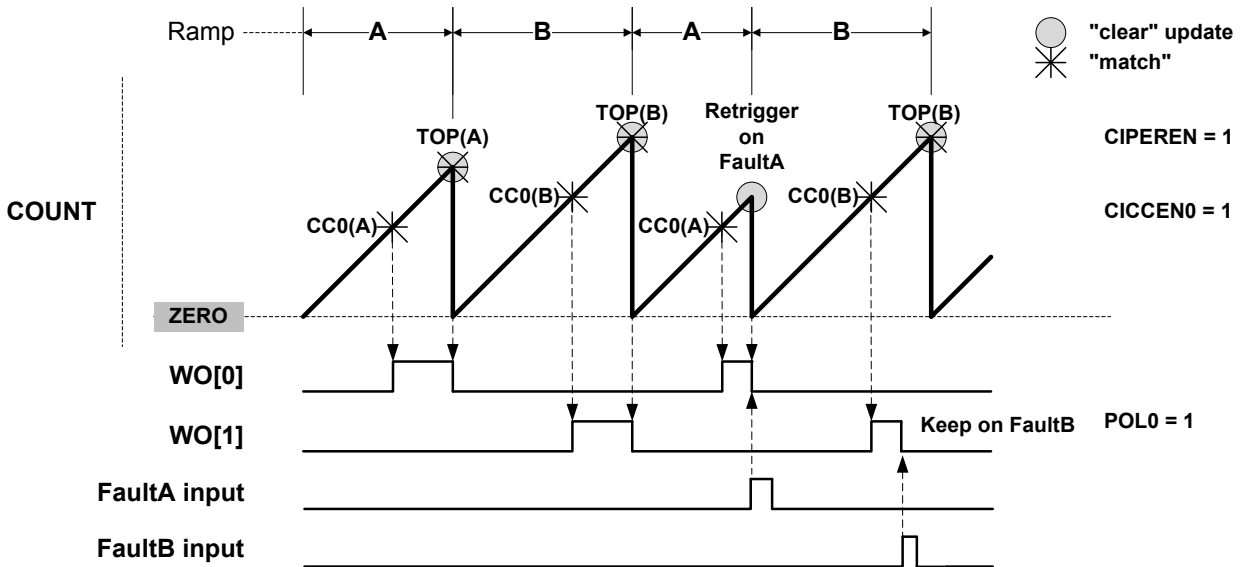
**Figure 32-18. RAMP2 Standard Operation**



### Alternate RAMP2 (RAMP2A) Operation

Alternate RAMP2 operation is similar to RAMP2, but CC0 controls both WO[0] and WO[1] waveforms when the corresponding circular buffer option is enabled (CIPEREN=1). The waveform polarity is the same on both outputs. Channel 1 can be used in capture mode.

Figure 32-19. RAMP2 Alternate Operation



### 32.6.3.5 Recoverable Faults

Recoverable faults can restart or halt the timer/counter. Two faults, called Fault A and Fault B, can trigger recoverable fault actions on the compare channels CC0 and CC1 of the TCC. The compare channels' outputs can be clamped to inactive state either as long as the fault condition is present, or from the first valid fault condition detection on until the end of the timer/counter cycle.

#### Fault Inputs

The first two channel input events (TCCxMC0 and TCCxMC1) can be used as Fault A and Fault B inputs, respectively. Event system channels connected to these fault inputs must be configured as asynchronous. The TCC must work in a PWM mode.

#### Fault Filtering

There are three filters available for each input Fault A and Fault B. They are configured by the corresponding Recoverable Fault n Configuration registers (FCTRLA and FCTRLB). The three filters can either be used independently or in any combination.

**Input Filtering** By default, the event detection is asynchronous. When the event occurs, the fault system will immediately and asynchronously perform the selected fault action on the compare channel output, also in device power modes where the clock is not available. To avoid false fault detection on external events (e.g. due to a glitch on an I/O port) a digital filter can be enabled and configured by the Fault B Filter Value bits in the Fault n Configuration registers (FCTRLn.FILTERVAL). If the event width is less than FILTERVAL (in clock cycles), the event will be discarded. A valid event will be delayed by FILTERVAL clock cycles.

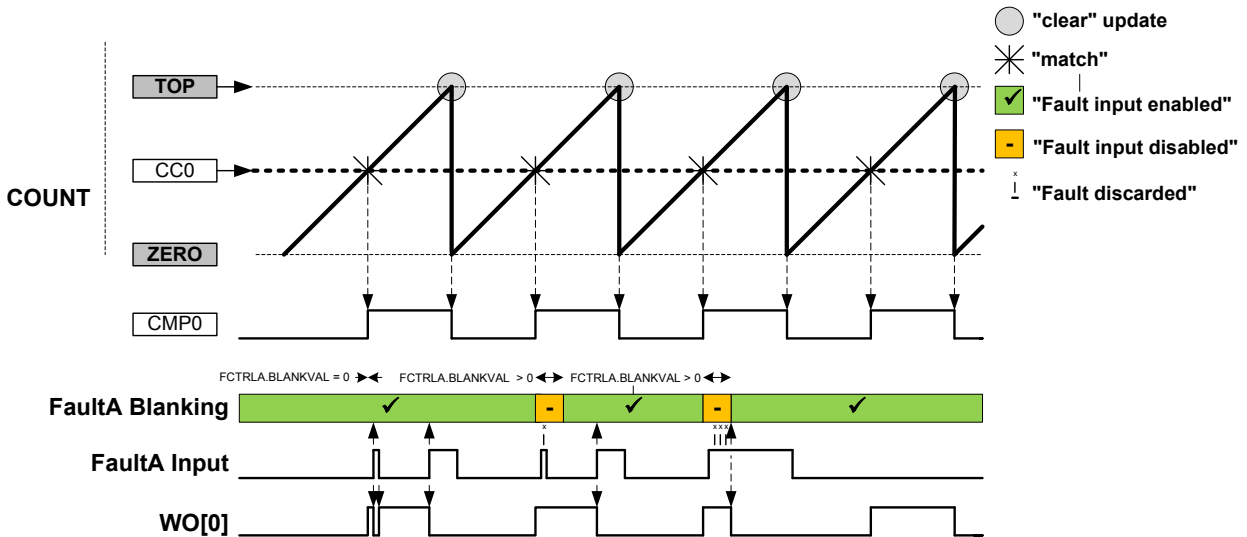
**Fault Blanking** This ignores any fault input for a certain time just after a selected waveform output edge. This can be used to prevent false fault triggering due to signal bouncing, as shown in the figure below. Blanking can be enabled by writing an edge triggering configuration to the Fault n Blanking Mode bits in the Recoverable Fault n Configuration register (FCTRLn.BLANK). The desired duration of the blanking must be written to the Fault n Blanking Time bits (FCTRLn.BLANKVAL). The blanking time  $t_b$  is calculated by

$$t_b = \frac{1 + \text{BLANKVAL}}{f_{\text{GCLK\_TCCx\_PRESC}}}$$

Here,  $f_{\text{GCLK\_TCCx\_PRESC}}$  is the frequency of the prescaled peripheral clock frequency  $f_{\text{GCLK\_TCCx}}$ .

The maximum blanking time ( $\text{FCTRLn.BLANKVAL} = 255$ ) at  $f_{\text{GCLK\_TCCx}} = 96\text{MHz}$  is  $2.67\mu\text{s}$  (no prescaler) or  $170\mu\text{s}$  (prescaling). For  $f_{\text{GCLK\_TCCx}} = 1\text{MHz}$ , the maximum blanking time is either  $170\mu\text{s}$  (no prescaling) or  $10.9\text{ms}$  (prescaling enabled).

**Figure 32-20. Fault Blanking in RAMP1 Operation with Inverted Polarity**



**Fault Qualification** This is enabled by writing a '1' to the Fault n Qualification bit in the Recoverable Fault n Configuration register ( $\text{FCTRLn.QUAL}$ ). When the recoverable fault qualification is enabled ( $\text{FCTRLn.QUAL} = 1$ ), the fault input is disabled all the time the corresponding channel output has an inactive level, as shown in the figures below.

**Figure 32-21. Fault Qualification in RAMP1 Operation**

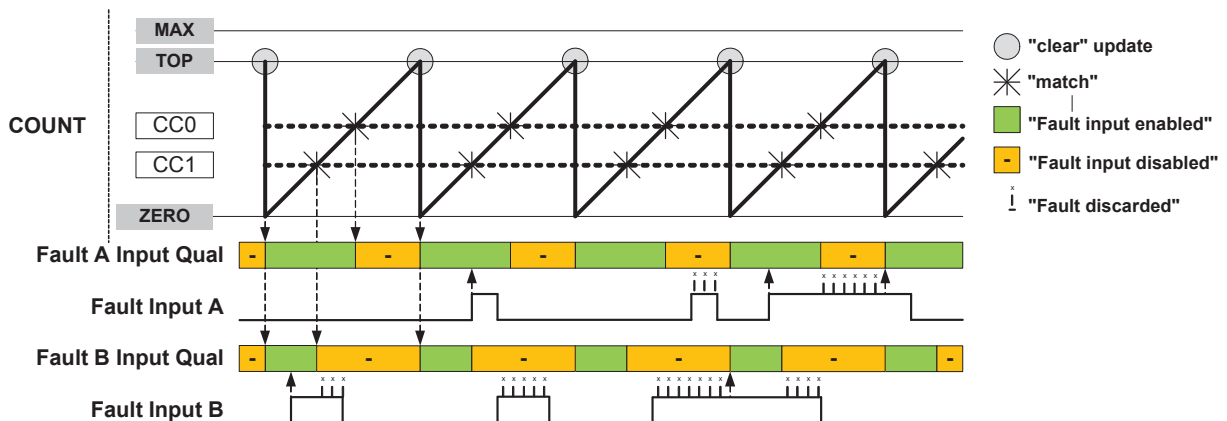
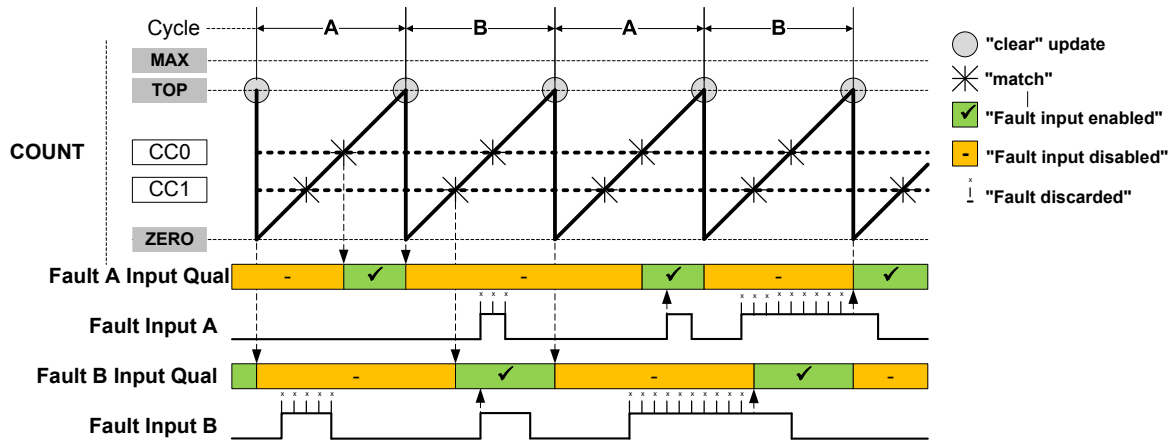




Figure 32-22. Fault Qualification in RAMP2 Operation with Inverted Polarity

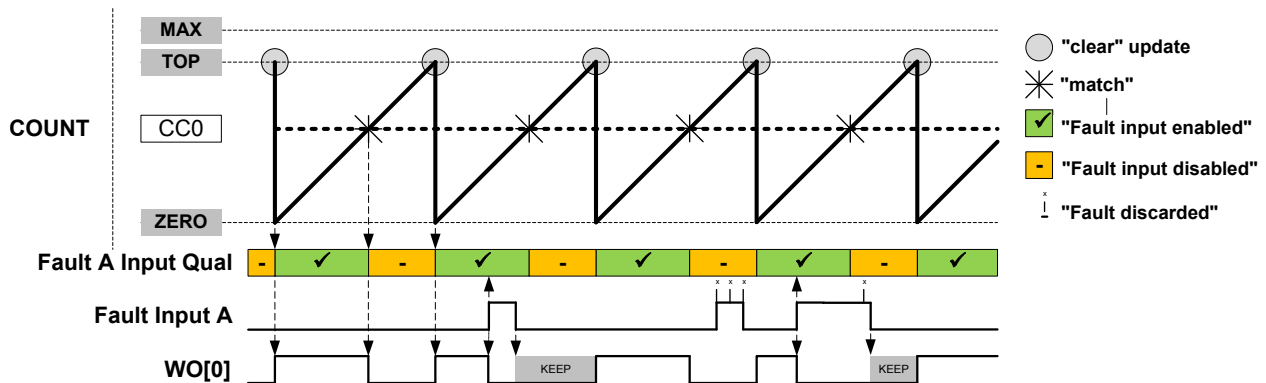


**Fault Actions**

Different fault actions can be configured individually for Fault A and Fault B. Most fault actions are not mutually exclusive; hence two or more actions can be enabled at the same time to achieve a result that is a combination of fault actions.

**Keep Action** This is enabled by writing the Fault n Keeper bit in the Recoverable Fault n Configuration register (FCTRLn.KEEP) to '1'. When enabled, the corresponding channel output will be clamped to zero as long as the fault condition is present. The clamp will be released on the start of the first cycle after the fault condition is no longer present, see next Figure.

Figure 32-23. Waveform Generation with Fault Qualification and Keep Action



**Restart Action** This is enabled by writing the Fault n Restart bit in Recoverable Fault n Configuration register (FCTRLn.RESTART) to '1'. When enabled, the timer/counter will be restarted as soon as the corresponding fault condition is present. The ongoing cycle is stopped and the timer/counter starts a new cycle, see Figure 32-24. In Ramp 1 mode, when the new cycle starts, the compare outputs will be clamped to inactive level as long as the fault condition is present.

**Note:** For RAMP2 operation, when a new timer/counter cycle starts the cycle index will change automatically, see Figure 32-25. Fault A and Fault B are qualified only during the cycle A and cycle B respectively: Fault A is disabled during cycle B, and Fault B is disabled during cycle A.

Figure 32-24. Waveform Generation in RAMP1 mode with Restart Action

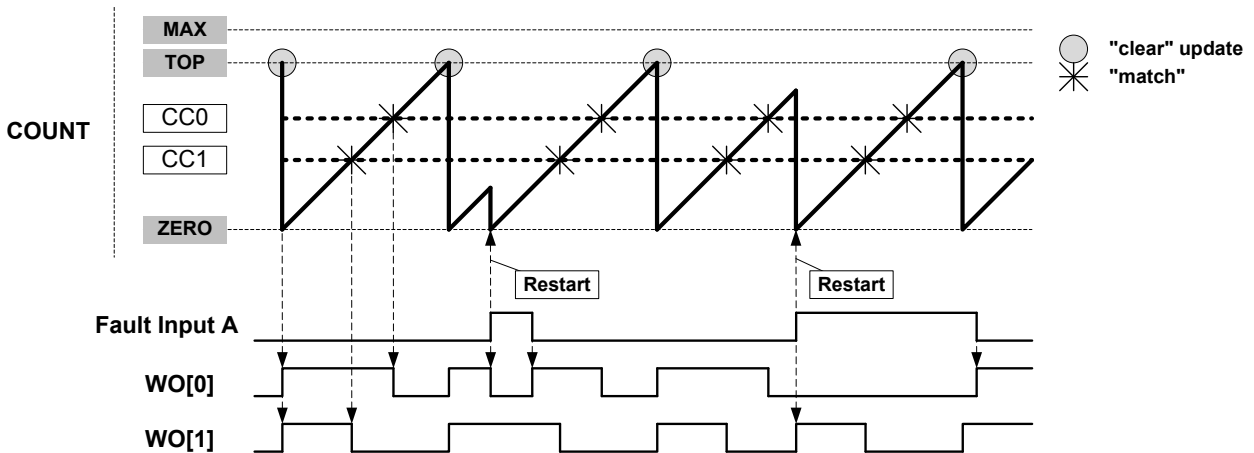
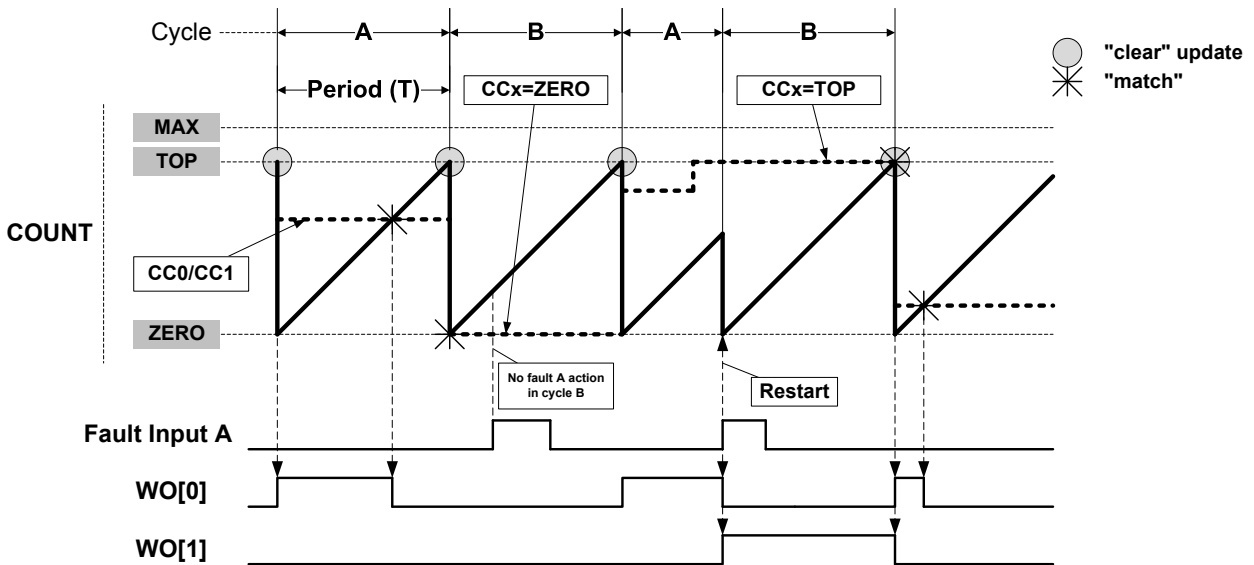


Figure 32-25. Waveform Generation in RAMP2 mode with Restart Action



**Capture Action** Several capture actions can be selected by writing the Fault n Capture Action bits in the Fault n Control register (FCTRLn.CAPTURE). When one of the capture operations is selected, the counter value is captured when the fault occurs. These capture operations are available:

- CAPT - the equivalent to a standard capture operation, for further details refer to [Capture Operations](#)
- CAPTMIN - gets the minimum time stamped value: on each new local minimum captured value, an event or interrupt is issued.
- CAPTMAX - gets the maximum time stamped value: on each new local maximum captured value, an event or interrupt (IT) is issued, see [Figure 32-26](#).
- LOCMIN - notifies by event or interrupt when a local minimum captured value is detected.
- LOCMAX - notifies by event or interrupt when a local maximum captured value is detected.
- DERIV0 - notifies by event or interrupt when a local extreme captured value is detected, see [Figure 32-27](#).

### CCx Content:

In CAPTMIN and CAPTMAX operations, CCx keeps the respective extremum captured values, see [Figure 32-26](#). In LOCMIN, LOCMAX or DERIV0 operation, CCx follows the counter value at fault time, see [Figure 32-27](#).

Before enabling CAPTMIN or CAPTMAX mode of capture, the user must initialize the corresponding CCx register value to a value different from zero (for CAPTMIN) top (for CAPTMAX). If the CCx register initial value is zero (for CAPTMIN) top (for CAPTMAX), no captures will be performed using the corresponding channel.

### MCx Behaviour:

In LOCMIN and LOCMAX operation, capture is performed on each capture event. The MCx interrupt flag is set only when the captured value is upper or equal (for LOCMIN) or lower or equal (for LOCMAX) to the previous captured value. So interrupt flag is set when a new relative local Minimum (for CAPTMIN) or Maximum (for CAPTMAX) value has been detected. DERIV0 is equivalent to an OR function of (LOCMIN, LOCMAX).

In CAPT operation, capture is performed on each capture event. The MCx interrupt flag is set on each new capture.

In CAPTMIN and CAPTMAX operation, capture is performed only when on capture event time, the counter value is lower (for CAPTMIN) or upper (for CAPMAX) than the last captured value. The MCx interrupt flag is set only when on capture event time, the counter value is upper or equal (for CAPTMIN) or lower or equal (for CAPTMAX) to the value captured on the previous event. So interrupt flag is set when a new absolute local Minimum (for CAPTMIN) or Maximum (for CAPTMAX) value has been detected.

### Interrupt Generation

In CAPT mode, an interrupt is generated on each filtered Fault n and each dedicated CCx channel capture counter value. In other modes, an interrupt is only generated on an extreme captured value.

**Figure 32-26. Capture Action “CAPTMAX”**

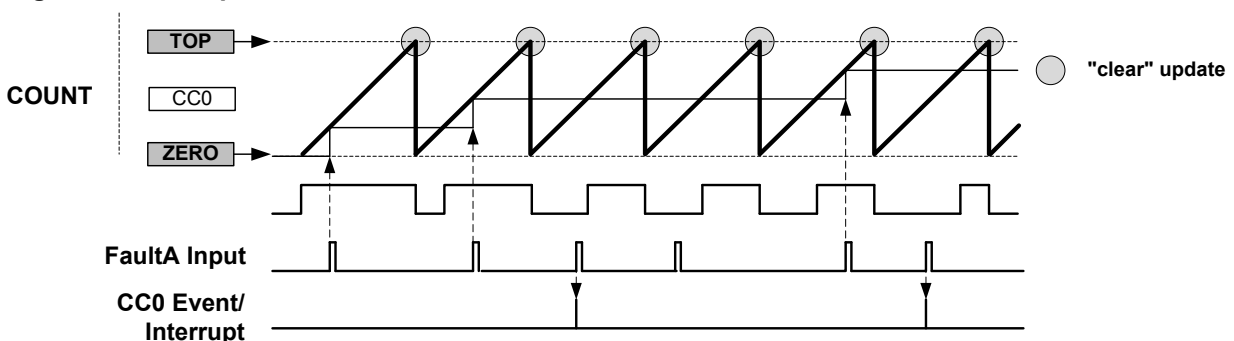
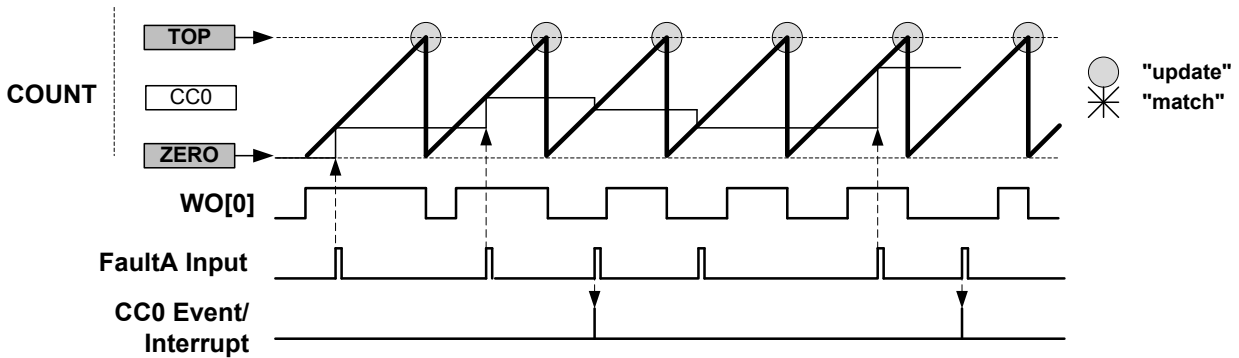


Figure 32-27. Capture Action “DERIV0”



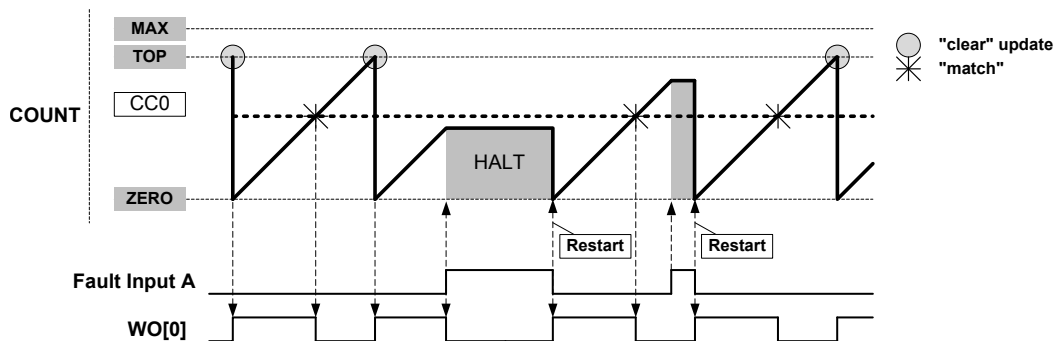
**Hardware Halt Action** This is configured by writing 0x1 to the Fault n Halt mode bits in the Recoverable Fault n Configuration register (FCTRLn.HALT). When enabled, the timer/counter is halted and the cycle is extended as long as the corresponding fault is present.

The next figure ('Waveform Generation with Halt and Restart Actions') shows an example where both restart action and hardware halt action are enabled for Fault A. The compare channel 0 output is clamped to inactive level as long as the timer/counter is halted. The timer/counter resumes the counting operation as soon as the fault condition is no longer present. As the restart action is enabled in this example, the timer/counter is restarted after the fault condition is no longer present.

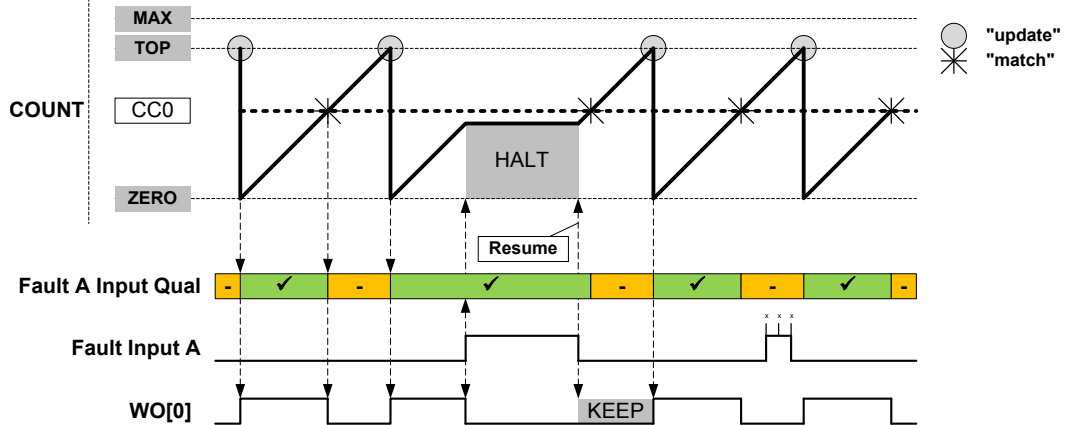
The figure after that ('Waveform Generation with Fault Qualification, Halt, and Restart Actions') shows a similar example, but with additionally enabled fault qualification. Here, counting is resumed after the fault condition is no longer present.

Note that in RAMP2 and RAMP2A operations, when a new timer/counter cycle starts, the cycle index will automatically change.

Figure 32-28. Waveform Generation with Halt and Restart Actions



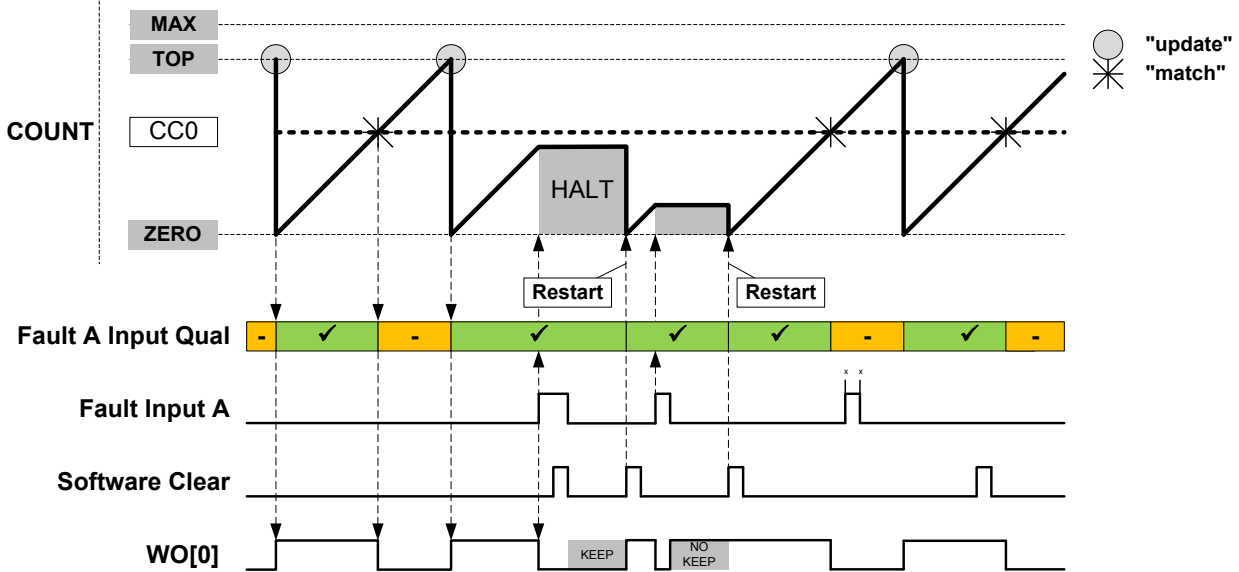
**Figure 32-29. Waveform Generation with Fault Qualification, Halt, and Restart Actions**



**Software Halt Action**

This is configured by writing 0x2 to the Fault n Halt mode bits in the Recoverable Fault n configuration register (FCTRLn.HALT). Software halt action is similar to hardware halt action, but in order to restart the timer/counter, the corresponding fault condition must not be present anymore, and the corresponding FAULT n bit in the STATUS register must be cleared by software.

**Figure 32-30. Waveform Generation with Software Halt, Fault Qualification, Keep and Restart Actions**



**32.6.3.6 Non-Recoverable Faults**

The non-recoverable fault action will force all the compare outputs to a pre-defined level programmed into the Driver Control register (DRVCTRL.NRE and DRVCTRL.NRV). The non-recoverable fault input (EV0 and EV1) actions are enabled in Event Control register (EVCTRL.EVACT0 and EVCTRL.EVACT1).

To avoid false fault detection on external events (e.g. a glitch on an I/O port) a digital filter can be enabled using Non-Recoverable Fault Input x Filter Value bits in the Driver Control register (DRVCTRL.FILTERVALn). Therefore, the event detection is synchronous, and event action is delayed by the selected digital filter value clock cycles.

When the Fault Detection on Debug Break Detection bit in Debug Control register (DGBCTRL.FDDBD) is written to '1', a non-recoverable Debug Faults State and an interrupt (DFS) is generated when the system goes in debug operation.

32.6.3.7 Waveform Extension

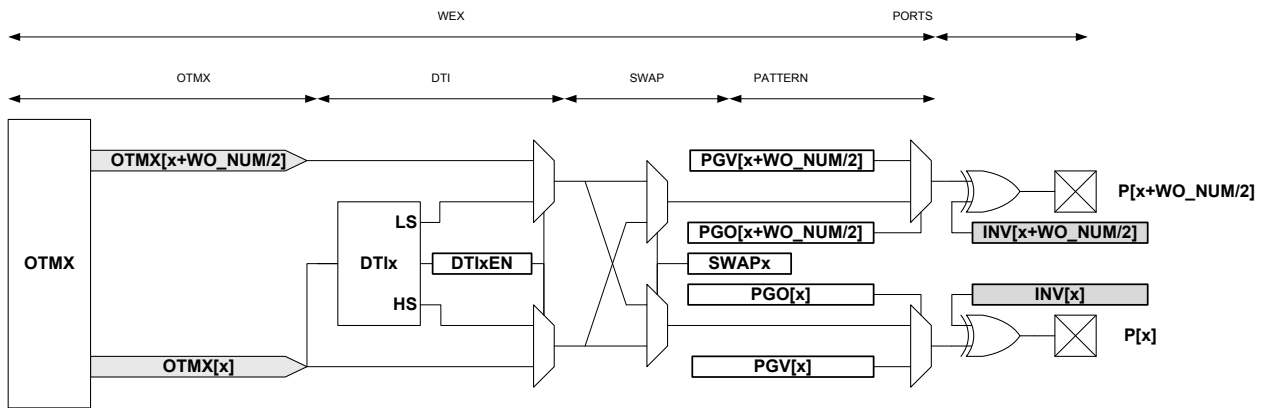
Figure 32-31 shows a schematic diagram of actions of the four optional units that follow the recoverable fault stage on a port pin pair: Output Matrix (OTMX), Dead-Time Insertion (DTI), SWAP and Pattern Generation. The DTI and SWAP units can be seen as a four port pair slices:

- Slice 0 DTI0 / SWAP0 acting on port pins (WO[0], WO[WO\_NUM/2 +0])
- Slice 1 DTI1 / SWAP1 acting on port pins (WO[1], WO[WO\_NUM/2 +1])

And more generally:

- Slice n DTIx / SWAPx acting on port pins (WO[x], WO[WO\_NUM/2 +x])

Figure 32-31. Waveform Extension Stage Details



The output matrix (OTMX) unit distributes compare channels, according to the selectable configurations in Table 32-4.

Table 32-4. Output Matrix Channel Pin Routing Configuration

Value	OTMX[x]							
0x0	CC3	CC2	CC1	CC0	CC3	CC2	CC1	CC0
0x1	CC1	CC0	CC1	CC0	CC1	CC0	CC1	CC0
0x2	CC0	CC0	CC0	CC0	CC0	CC0	CC0	CC0
0x3	CC1	CC1	CC1	CC1	CC1	CC1	CC1	CC0

Notes on Table 32-4:

- Configuration 0x0 is the default configuration. The channel location is the default one, and channels are distributed on outputs modulo the number of channels. Channel 0 is routed to the Output matrix output OTMX[0], and Channel 1 to OTMX[1]. If there are more outputs than channels, then channel 0 is duplicated to the Output matrix output OTMX[CC\_NUM], channel 1 to OTMX[CC\_NUM+1] and so on.
- Configuration 0x1 distributes the channels on output modulo half the number of channels. This assigns twice the number of output locations to the lower channels than the default configuration. This can be used, for example, to control the four transistors of a full bridge using only two compare channels.

Using pattern generation, some of these four outputs can be overwritten by a constant level, enabling flexible drive of a full bridge in all quadrant configurations.

- Configuration 0x2 distributes compare channel 0 (CC0) to all port pins. With pattern generation, this configuration can control a stepper motor.
- Configuration 0x3 distributes the compare channel CC0 to the first output, and the channel CC1 to all other outputs. Together with pattern generation and the fault extension, this configuration can control up to seven LED strings, with a boost stage.

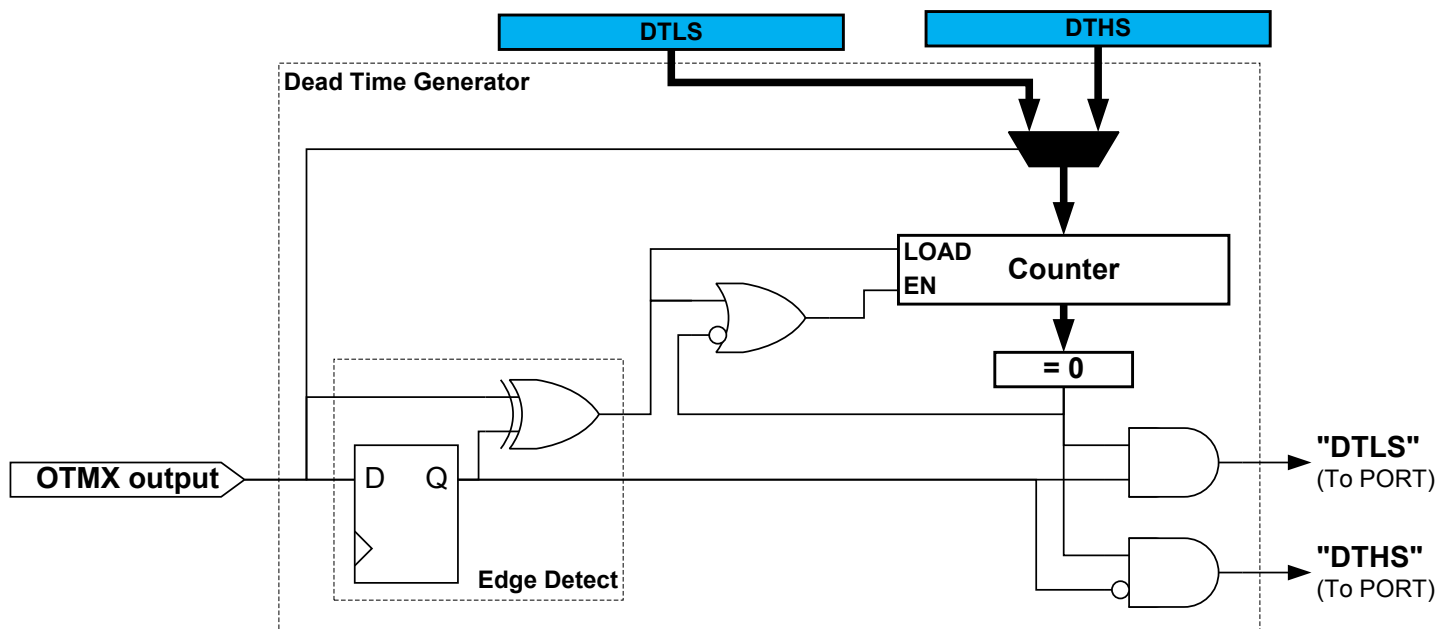
**Table 32-5. Example: four compare channels on four outputs**

Value	OTMX[3]	OTMX[2]	OTMX[1]	OTMX[0]
0x0	CC3	CC2	CC1	CC0
0x1	CC1	CC0	CC1	CC0
0x2	CC0	CC0	CC0	CC0
0x3	CC1	CC1	CC1	CC0

The dead-time insertion (DTI) unit generates OFF time with the non-inverted low side (LS) and inverted high side (HS) of the wave generator output forced at low level. This OFF time is called dead time. Dead-time insertion ensures that the LS and HS will never switch simultaneously.

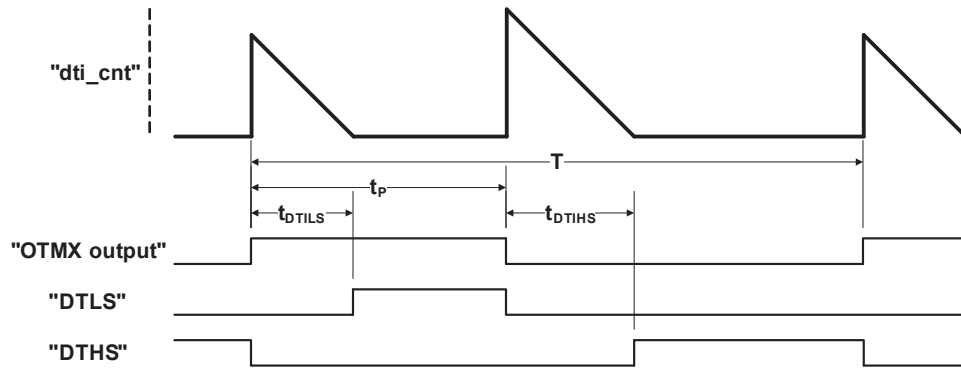
The DTI stage consists of four equal dead-time insertion generators; one for each of the first four compare channels. Figure 32-32 shows the block diagram of one DTI generator. The four channels have a common register which controls the dead time, which is independent of high side and low side setting.

**Figure 32-32. Dead-Time Generator Block Diagram**



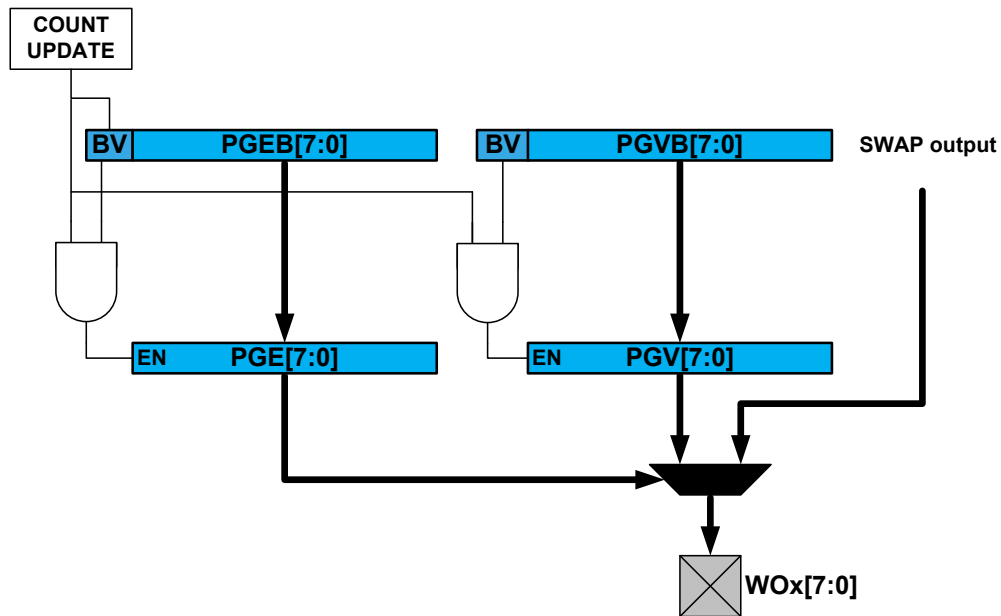
As shown in Figure 32-33, the 8-bit dead-time counter is decremented by one for each peripheral clock cycle until it reaches zero. A non-zero counter value will force both the low side and high side outputs into their OFF state. When the output matrix (OTMX) output changes, the dead-time counter is reloaded according to the edge of the input. When the output changes from low to high (positive edge) it initiates a counter reload of the DTLs register. When the output changes from high to low (negative edge) it reloads the DTHS register.

Figure 32-33. Dead-Time Generator Timing Diagram



The pattern generator unit produces a synchronized bit pattern across the port pins it is connected to. The pattern generation features are primarily intended for handling the commutation sequence in brushless DC motors (BLDC), stepper motors, and full bridge control. See also [Figure 32-34](#).

Figure 32-34. Pattern Generator Block Diagram



As with other double-buffered timer/counter registers, the register update is synchronized to the UPDATE condition set by the timer/counter waveform generation operation. If synchronization is not required by the application, the software can simply access directly the PATT.PGE, PATT.PGV bits registers.

32.6.4 DMA, Interrupts, and Events

Table 32-6. Module Requests for TCC

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Overflow / Underflow	Yes	Yes		Yes <sup>(1)</sup>	On DMA acknowledge
Channel Compare Match or Capture	Yes	Yes	Yes <sup>(2)</sup>	Yes <sup>(3)</sup>	For circular buffering: on DMA acknowledge



Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
					For capture channel: when CCx register is read
Retrigger	Yes	Yes			
Count	Yes	Yes			
Capture Overflow Error	Yes				
Debug Fault State	Yes				
Recoverable Faults	Yes				
Non-Recoverable Faults	Yes				
TCCx Event 0 input			Yes <sup>(4)</sup>		
TCCx Event 1 input			Yes <sup>(5)</sup>		

### Notes:

1. DMA request set on overflow, underflow or re-trigger conditions.
2. Can perform capture or generate recoverable fault on an event input.
3. In capture or circular modes.
4. On event input, either action can be executed:
  - re-trigger counter
  - control counter direction
  - stop the counter
  - decrement the counter
  - perform period and pulse width capture
  - generate non-recoverable fault
5. On event input, either action can be executed:
  - re-trigger counter
  - increment or decrement counter depending on direction
  - start the counter
  - increment or decrement counter based on direction
  - increment counter regardless of direction
  - generate non-recoverable fault

### 32.6.4.1 DMA Operation

The TCC can generate the following DMA requests:

**Counter overflow (OVF)** If the Ones-shot Trigger mode in the control A register (CTRLA.DMAOS) is written to '0', the TCC generates a DMA request on each cycle when an update condition (overflow, underflow or re-trigger) is detected.  
When an update condition (overflow, underflow or re-trigger) is detected while CTRLA.DMAOS=1, the TCC generates a DMA trigger on the cycle following the DMA One-Shot Command written to the Control B register (CTRLBSET.CMD=DMAOS).

In both cases, the request is cleared by hardware on DMA acknowledge.

**Channel Match (MCx)** A DMA request is set only on a compare match if CTRLA.DMAOS=0. The request is cleared by hardware on DMA acknowledge.

When CTRLA.DMAOS=1, the DMA requests are not generated.

**Channel Capture (MCx)** For a capture channel, the request is set when valid data is present in the CCx register, and cleared once the CCx register is read.

In this operation mode, the CTRLA.DMAOS bit value is ignored.

**DMA Operation with Circular Buffer**

When circular buffer operation is enabled, the buffer registers must be written in a correct order and synchronized to the update times of the timer. The DMA triggers of the TCC provide a way to ensure a safe and correct update of circular buffers.

**Note:** Circular buffer are intended to be used with RAMP2, RAMP2A and DSBOTH operation only.

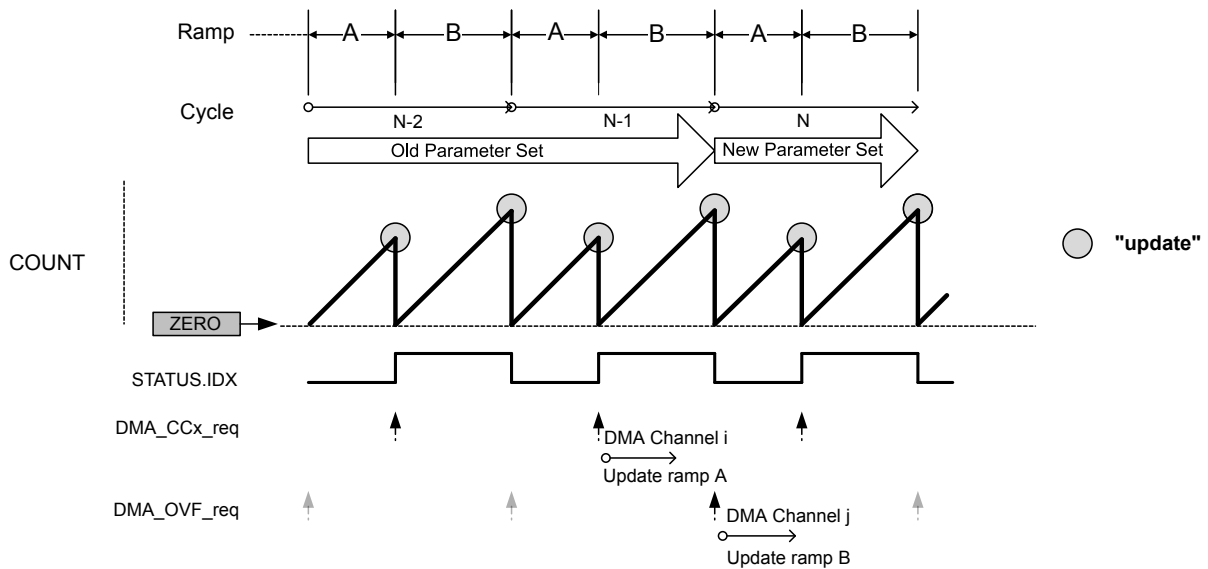
*DMA Operation with Circular Buffer in RAMP2 and RAMP2A Mode*

When a CCx channel is selected as a circular buffer, the related DMA request is not set on a compare match detection, but on start of ramp B.

If at least one circular buffer is enabled, the DMA overflow request is conditioned to the start of ramp A with an effective DMA transfer on previous ramp B (DMA acknowledge).

The update of all circular buffer values for ramp A can be done through a DMA channel triggered on a MC trigger. The update of all circular buffer values for ramp B, can be done through a second DMA channel triggered by the overflow DMA request.

**Figure 32-35. DMA Triggers in RAMP and RAMP2 Operation Mode and Circular Buffer Enabled**



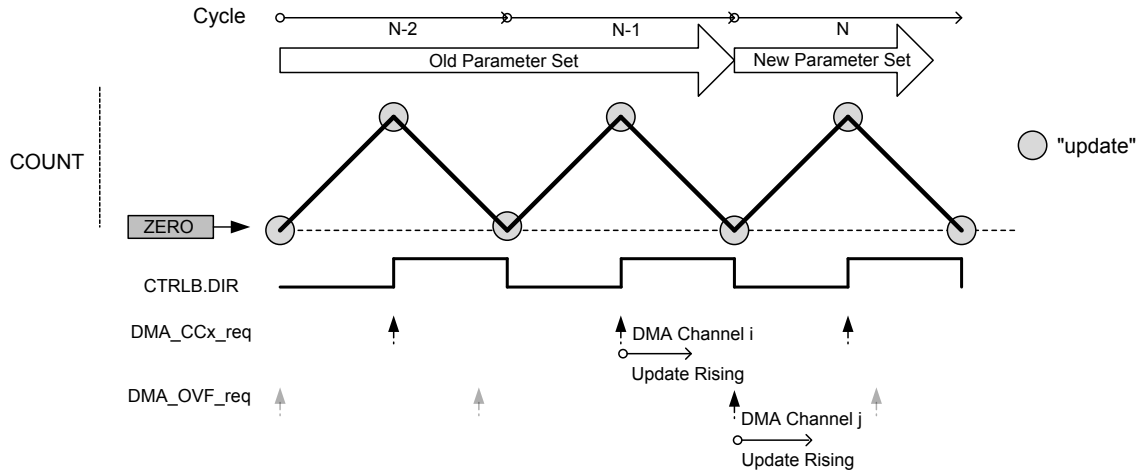
**DMA Operation with Circular Buffer in DSBOTH Mode**

When a CC channel is selected as a circular buffer, the related DMA request is not set on a compare match detection, but on start of down-counting phase.

If at least one circular buffer is enabled, the DMA overflow request is conditioned to the start of up-counting phase with an effective DMA transfer on previous down-counting phase (DMA acknowledge).

When up-counting, all circular buffer values can be updated through a DMA channel triggered by MC trigger. When down-counting, all circular buffer values can be updated through a second DMA channel, triggered by the OVF DMA request.

**Figure 32-36. DMA Triggers in DSBOOTH Operation Mode and Circular Buffer Enabled**



### 32.6.4.2 Interrupts

The TCC has the following interrupt sources:

- Overflow/Underflow (OVF)
- Retrigger (TRG)
- Count (CNT) - refer also to description of [EVCTRL.CNTSEL](#).
- Capture Overflow Error (ERR)
- Debug Fault State (DFS)
- Recoverable Faults (FAULTn)
- Non-recoverable Faults (FAULTx)
- Compare Match or Capture Channels (MCx)

These interrupts are asynchronous wake-up sources. See Sleep Mode Entry and Exit Table in PM/Sleep Mode Controller section for details.

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 '1' to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by writing a '1' 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 TCC is reset. See [INTFLAG](#) for details on how to clear interrupt flags. The TCC 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: Interrupts must be globally enabled for interrupt requests to be generated. Refer to *Nested Vector Interrupt Controller* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

[Sleep Mode Controller](#)

[IDLE Mode](#)

[STANDBY Mode](#)

### 32.6.4.3 Events

The TCC can generate the following output events:

- Overflow/Underflow (OVF)
- Trigger (TRG)
- Counter (CNT) For further details, refer to [EVCTRL.CNTSEL](#) description.
- Compare Match or Capture on compare/capture channels: MCx

Writing a '1' ('0') to an Event Output bit in the Event Control Register (EVCTRL.xxEO) enables (disables) the corresponding output event. Refer also to *EVSYS – Event System*.

The TCC can take the following actions on a channel input event (MCx):

- Capture event
- Generate a recoverable or non-recoverable fault

The TCC can take the following actions on counter Event 1 (TCCx EV1):

- Counter re-trigger
- Counter direction control
- Stop the counter
- Decrement the counter on event
- Period and pulse width capture
- Non-recoverable fault

The TCC can take the following actions on counter Event 0 (TCCx EV0):

- Counter re-trigger
- Count on event (increment or decrement, depending on counter direction)
- Counter start - start counting on the event rising edge. Further events will not restart the counter; the counter will keep on counting using prescaled GCLK\_TCCx, until it reaches TOP or ZERO, depending on the direction.
- Counter increment on event. This will increment the counter, irrespective of the counter direction.
- Count during active state of an asynchronous event (increment or decrement, depending on counter direction). In this case, the counter will be incremented or decremented on each cycle of the prescaled clock, as long as the event is active.
- Non-recoverable fault

The counter Event Actions are available in the Event Control registers (EVCTRL.EVACT0 and EVCTRL.EVACT1). For further details, refer to [EVCTRL](#).

Writing a '1' ('0') to an Event Input bit in the Event Control register (EVCTRL.MCEIx or EVCTRL.TCEIx) enables (disables) the corresponding action on input event.

**Note:** When several events are connected to the TCC, the enabled action will apply for each of the incoming events. Refer to *EVSYS – Event System* for details on how to configure the event system.

#### Related Links

[EVSYS – Event System](#)

### 32.6.5 Sleep Mode Operation

The TCC 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 '1'. The MODULE can in any sleep mode wake up the device using interrupts or perform actions through the Event System.

### 32.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset and Enable bits in Control A register (CTRLA.SWRST and CTRLA.ENABLE)

The following registers are synchronized when written:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Status register (STATUS)
- Pattern and Pattern Buffer registers (PATT and PATTB)
- Waveform register (WAVE)
- Count Value register (COUNT)
- Period Value and Period Buffer Value registers (PER and PERB)
- Compare/Capture Channel x and Channel x Compare/Capture Buffer Value registers (CCx and CCBx)

The following registers are synchronized when read:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Count Value register (COUNT): synchronization is done on demand through READSYNC command (CTRLBSET.CMD)
- Pattern and Pattern Buffer registers (PATT and PATTB)
- Waveform register (WAVE)
- Period Value and Period Buffer Value registers (PER and PERB)
- Compare/Capture Channel x and Channel x Compare/Capture Buffer Value registers (CCx and CCBx)

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

Required read-synchronization is denoted by the "Read-Synchronized" property in the register description.

#### Related Links

[Register Synchronization](#)

### 32.7 Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		RESOLUTION[1:0]					ENABLE	SWRST
		15:8		ALOCK	PRESCYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
		23:16								
		31:24					CPTEN3	CPTEN2	CPTEN1	CPTEN0
0x04	CTRLBCLR	7:0		CMD[2:0]		IDXCMD[1:0]	ONESHOT	LUPD	DIR	
0x05	CTRLBSET	7:0		CMD[2:0]		IDXCMD[1:0]	ONESHOT	LUPD	DIR	
0x06 ... 0x07	Reserved									
0x08	SYNCBUSY	7:0	PER	WAVE	PATT	COUNT	STATUS	CTRLB	ENABLE	SWRST
		15:8					CC3	CC2	CC1	CC0
		23:16		CCB3	CCB2	CCB1	CCB0	PERB	WAVEB	PATTB
		31:24								
0x0C	FCTRLA	7:0	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]	
		15:8		CAPTURE[2:0]			CHSEL[1:0]		HALT[1:0]	
		23:16		BLANKVAL[7:0]						
		31:24		FILTERVAL[3:0]						
0x10	FCTRLB	7:0	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]	
		15:8		CAPTURE[2:0]			CHSEL[1:0]		HALT[1:0]	
		23:16		BLANKVAL[7:0]						
		31:24		FILTERVAL[3:0]						
0x14	WEXCTRL	7:0							OTMX[1:0]	
		15:8					DTIENx	DTIENx	DTIENx	DTIENx
		23:16		DTLS[7:0]						
		31:24		DTHS[7:0]						
0x18	DRVCTRL	7:0	NREx	NREx	NREx	NREx	NREx	NREx	NREx	NREx
		15:8	NRVx	NRVx	NRVx	NRVx	NRVx	NRVx	NRVx	NRVx
		23:16	INVENx	INVENx	INVENx	INVENx	INVENx	INVENx	INVENx	INVENx
		31:24	FILTERVAL1[3:0]				FILTERVAL0[3:0]			
0x1C ... 0x1D	Reserved									
0x1E	DBGCTRL	7:0					FDDBD		DBGRUN	
0x1F	Reserved									
0x20	EVCTRL	7:0	CNTSEL[1:0]		EVACT1[2:0]			EVACT0[2:0]		
		15:8	TCEIx	TCEIx	TCINVx	TCINVx		CNTEO	TRGEO	OVFEO
		23:16					MCEIx	MCEIx	MCEIx	MCEIx
		31:24					MCEOx	MCEOx	MCEOx	MCEOx
0x24	INTENCLR	7:0					ERR	CNT	TRG	OVF
		15:8	FAULTx	FAULTx	FAULTB	FAULTA	DFS			
		23:16					MCx	MCx	MCx	MCx
0x27	Reserved									
0x28	INTENSET	7:0					ERR	CNT	TRG	OVF

Offset	Name	Bit Pos.								
		15:8	FAULTx	FAULTx	FAULTB	FAULTA	DFS			
		23:16					MCx	MCx	MCx	MCx
0x2B	Reserved									
		7:0					ERR	CNT	TRG	OVF
		15:8	FAULTx	FAULTx	FAULTB	FAULTA	DFS			
		23:16					MCx	MCx	MCx	MCx
0x2C	INTFLAG									
0x2F	Reserved									
		7:0	PERBV	WAVEBV	PATTBV		DFS		IDX	STOP
		15:8	FAULTx	FAULTx	FAULTB	FAULTA	FAULT1IN	FAULT0IN	FAULTBIN	FAULTAIN
		23:16					CCBVx	CCBVx	CCBVx	CCBVx
		31:24					CMPx	CMPx	CMPx	CMPx
		7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
		23:16	COUNT[23:16]							
		31:24								
		7:0	PGE0[7:0]							
		15:8	PGV0[7:0]							
0x38	PATT									
0x3A ... 0x3B	Reserved									
		7:0	CIPEREN					WAVEGEN[2:0]		
		15:8					CICCEN3	CICCEN2	CICCEN1	CICCEN0
		23:16					POL3	POL2	POL1	POL0
		31:24					SWAP3	SWAP2	SWAP1	SWAP0
		7:0	PER[1:0]		DITHER[5:0]					
		15:8	PER[9:2]							
		23:16	PER[17:10]							
		31:24								
		7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
		7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
		7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
0x54 ...	Reserved									

Offset	Name	Bit Pos.							
0x63									
0x64	PATTB	7:0	PGEB0[7:0]						
		15:8	PGVB0[7:0]						
0x66 ... 0x67	Reserved								
0x68	WAVEB	7:0	CIPERENB	RAMPB[1:0]			WAVEGENB[2:0]		
		15:8				CICCENB3	CICCENB2	CICCENB1	CICCENB0
		23:16				POLB3	POLB2	POLB1	POLB0
		31:24				SWAPB 3	SWAPB 2	SWAPB 1	SWAPB 0
0x6C	PERB	7:0	PERB[1:0]		DITHERB[5:0]				
		15:8	PERB[9:2]						
		23:16	PERB[17:10]						
		31:24							
0x70	CCB0	7:0	CCB[1:0]		DITHERB[5:0]				
		15:8	CCB[9:2]						
		23:16	CCB[17:10]						
		31:24							
0x74	CCB1	7:0	CCB[1:0]		DITHERB[5:0]				
		15:8	CCB[9:2]						
		23:16	CCB[17:10]						
		31:24							
0x78	CCB2	7:0	CCB[1:0]		DITHERB[5:0]				
		15:8	CCB[9:2]						
		23:16	CCB[17:10]						
		31:24							
0x7C	CCB3	7:0	CCB[1:0]		DITHERB[5:0]				
		15:8	CCB[9:2]						
		23:16	CCB[17:10]						
		31:24							

### 32.8 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 require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" and/or "Write-Synchronized" property in each individual register description.

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.



### 32.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized (ENABLE, SWRST)

Bit	31	30	29	28	27	26	25	24
					CPTEN3	CPTEN2	CPTEN1	CPTEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
		ALOCK	PRESCYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
		RESOLUTION[1:0]					ENABLE	SWRST
Access		R/W	R/W				R/W	R/W
Reset		0	0				0	0

#### Bits 24, 25, 26, 27 – CPTEN Capture Channel x Enable

These bits are used to select the capture or compare operation on channel x.

Writing a '1' to CPTENx enables capture on channel x.

Writing a '0' to CPTENx disables capture on channel x.

#### Bit 14 – ALOCK Auto Lock

This bit is not synchronized.

Value	Description
0	The Lock Update bit in the Control B register (CTRLB.LUPD) is not affected by overflow/underflow, and re-trigger events
1	CTRLB.LUPD is set to '1' on each overflow/underflow or re-trigger event.

#### Bits 13:12 – PRESCYNC[1:0] Prescaler and Counter Synchronization

These bits select if on re-trigger event, the Counter is cleared or reloaded on either the next GCLK\_TCCx clock, or on the next prescaled GCLK\_TCCx clock. It is also possible to reset the prescaler on re-trigger event.

These bits are not synchronized.

Value	Name	Description	
		Counter Reloaded	Prescaler
0x0	GCLK	Reload or reset Counter on next GCLK	-
0x1	PRESC	Reload or reset Counter on next prescaler clock	-
0x2	RESYNC	Reload or reset Counter on next GCLK	Reset prescaler counter
0x3	Reserved		

### Bit 11 – RUNSTDBY Run in Standby

This bit is used to keep the TCC running in standby mode.

This bit is not synchronized.

Value	Description
0	The TCC is halted in standby.
1	The TCC continues to run in standby.

### Bits 10:8 – PRESCALER[2:0] Prescaler

These bits select the Counter prescaler factor.

These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TCC
0x1	DIV2	Prescaler: GCLK_TCC/2
0x2	DIV4	Prescaler: GCLK_TCC/4
0x3	DIV8	Prescaler: GCLK_TCC/8
0x4	DIV16	Prescaler: GCLK_TCC/16
0x5	DIV64	Prescaler: GCLK_TCC/64
0x6	DIV256	Prescaler: GCLK_TCC/256
0x7	DIV1024	Prescaler: GCLK_TCC/1024

### Bits 6:5 – RESOLUTION[1:0] Dithering Resolution

These bits increase the TCC resolution by enabling the dithering options.

These bits are not synchronized.

**Table 32-7. Dithering**

Value	Name	Description
0x0	NONE	The dithering is disabled.
0x1	DITH4	Dithering is done every 16 PWM frames.  PER[3:0] and CCx[3:0] contain dithering pattern selection.
0x2	DITH5	Dithering is done every 32 PWM frames.

Value	Name	Description
		PER[4:0] and CCx[4:0] contain dithering pattern selection.
0x3	DITH6	Dithering is done every 64 PWM frames. PER[5:0] and CCx[5:0] contain dithering pattern selection.

### Bit 1 – ENABLE Enable

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 bit in the SYNCBUSY register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

### Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.

Writing a '1' to this bit resets all registers in the TCC (except DBGCTRL) to their initial state, and the TCC will be disabled.

Writing a '1' 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 SYNCBUSY.SWRST will both be cleared when the reset is complete.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

**32.8.2 Control B Clear**

**Name:** CTRLBCLR  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

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.

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]			IDXCMD[1:0]		ONESHOT	LUPD	DIR
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 7:5 – CMD[2:0] TCC Command**

These bits can be used for software control of re-triggering and stop commands of the TCC. When a command has been executed, the CMD bit field will read back zero. The commands are executed on the next prescaled GCLK\_TCC clock cycle.

Writing zero to this bit group has no effect.

Writing a '1' to any of these bits will clear the pending command.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Clear start, restart or retrigger
0x2	STOP	Force stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force COUNT read synchronization

**Bits 4:3 – IDXCMD[1:0] Ramp Index Command**

These bits can be used to force cycle A and cycle B changes in RAMP2 and RAMP2A operation. On timer/counter update condition, the command is executed, the IDX flag in STATUS register is updated and the IDXCMD command is cleared.

Writing zero to these bits has no effect.

Writing a '1' to any of these bits will clear the pending command.

Value	Name	Description
0x0	DISABLE	DISABLE Command disabled: IDX toggles between cycles A and B
0x1	SET	Set IDX: cycle B will be forced in the next cycle
0x2	CLEAR	Clear IDX: cycle A will be forced in next cycle
0x3	HOLD	Hold IDX: the next cycle will be the same as the current cycle.

**Bit 2 – ONESHOT One-Shot**

This bit controls one-shot operation of the TCC. When one-shot operation is enabled, the TCC will stop counting on the next overflow/underflow condition or on a stop command.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will disable the one-shot operation.

Value	Description
0	The TCC will update the counter value on overflow/underflow condition and continue operation.
1	The TCC will stop counting on the next underflow/overflow condition.

### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TCC buffered registers.

When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

This bit has no effect when input capture operation is enabled.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will enable updating.

Value	Description
0	The CCBx, PERB, PGVB, PGOB, and SWAPBx buffer registers values <i>are</i> copied into the corresponding CCx, PER, PGV, PGO and SWAPx registers on hardware update condition.
1	The CCBx, PERB, PGVB, PGOB, and SWAPBx buffer registers values are <i>not</i> copied into the corresponding CCx, PER, PGV, PGO and SWAPx registers on hardware update condition.

### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

32.8.3 Control B Set

**Name:** CTRLBSET  
**Offset:** 0x05  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

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.

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]			IDXCMD[1:0]		ONESHOT	LUPD	DIR
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 7:5 – CMD[2:0]** TCC Command

These bits can be used for software control of re-triggering and stop commands of the TCC. When a command has been executed, the CMD bit field will be read back as zero. The commands are executed on the next prescaled GCLK\_TCC clock cycle.

Writing zero to this bit group has no effect

Writing a valid value to this bit group will set the associated command.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force start, restart or retrigger
0x2	STOP	Force stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT

**Bits 4:3 – IDXCMD[1:0]** Ramp Index Command

These bits can be used to force cycle A and cycle B changes in RAMP2 and RAMP2A operation. On timer/counter update condition, the command is executed, the IDX flag in STATUS register is updated and the IDXCMD command is cleared.

Writing a zero to these bits has no effect.

Writing a valid value to these bits will set a command.

Value	Name	Description
0x0	DISABLE	Command disabled: IDX toggles between cycles A and B
0x1	SET	Set IDX: cycle B will be forced in the next cycle
0x2	CLEAR	Clear IDX: cycle A will be forced in next cycle
0x3	HOLD	Hold IDX: the next cycle will be the same as the current cycle.

**Bit 2 – ONESHOT** One-Shot

This bit controls one-shot operation of the TCC. When in one-shot operation, the TCC will stop counting on the next overflow/underflow condition or a stop command.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will enable the one-shot operation.

Value	Description
0	The TCC will count continuously.
1	The TCC will stop counting on the next underflow/overflow condition.

### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TCC buffered registers.

When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

This bit has no effect when input capture operation is enabled.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will lock updating.

Value	Description
0	The CCBx, PERB, PGVB, PGOB, and SWAPBx buffer registers values <i>are</i> copied into the corresponding CCx, PER, PGV, PGO and SWAPx registers on hardware update condition.
1	The CCBx, PERB, PGVB, PGOB, and SWAPBx buffer registers values are <i>not</i> copied into CCx, PER, PGV, PGO and SWAPx registers on hardware update condition.

### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

### 32.8.4 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
		CCB3	CCB2	CCB1	CCB0	PERB	WAVEB	PATTB
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
					CC3	CC2	CC1	CC0
Access					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	PER	WAVE	PATT	COUNT	STATUS	CTRLB	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 19, 20, 21, 22 – CCB Compare/Capture Buffer Channel x Synchronization Busy

This bit is cleared when the synchronization of Compare/Capture Buffer Channel x register between the clock domains is complete.

This bit is set when the synchronization of Compare/Capture Buffer Channel x register between clock domains is started.

CCBx bit is available only for existing Compare/Capture Channels. For details on CC channels number, refer to each TCC feature list.

#### Bit 18 – PERB PER Buffer Synchronization Busy

This bit is cleared when the synchronization of PERB register between the clock domains is complete.

This bit is set when the synchronization of PERB register between clock domains is started.

#### Bit 17 – WAVEB WAVE Buffer Synchronization Busy

This bit is cleared when the synchronization of WAVEB register between the clock domains is complete.

This bit is set when the synchronization of WAVEB register between clock domains is started.

#### Bit 16 – PATTB PATT Buffer Synchronization Busy

This bit is cleared when the synchronization of PATTB register between the clock domains is complete.

This bit is set when the synchronization of PATTB register between clock domains is started.



**Bits 8, 9, 10, 11 – CC** Compare/Capture Channel x Synchronization Busy

This bit is cleared when the synchronization of Compare/Capture Channel x register between the clock domains is complete.

This bit is set when the synchronization of Compare/Capture Channel x register between clock domains is started.

CCx bit is available only for existing Compare/Capture Channels. For details on CC channels number, refer to each TCC feature list.

This bit is set when the synchronization of CCx register between clock domains is started.

**Bit 7 – PER** PER Synchronization Busy

This bit is cleared when the synchronization of PER register between the clock domains is complete.

This bit is set when the synchronization of PER register between clock domains is started.

**Bit 6 – WAVE** WAVE Synchronization Busy

This bit is cleared when the synchronization of WAVE register between the clock domains is complete.

This bit is set when the synchronization of WAVE register between clock domains is started.

**Bit 5 – PATT** PATT Synchronization Busy

This bit is cleared when the synchronization of PATTERN register between the clock domains is complete.

This bit is set when the synchronization of PATTERN register between clock domains is started.

**Bit 4 – COUNT** COUNT Synchronization Busy

This bit is cleared when the synchronization of COUNT register between the clock domains is complete.

This bit is set when the synchronization of COUNT register between clock domains is started.

**Bit 3 – STATUS** STATUS Synchronization Busy

This bit is cleared when the synchronization of STATUS register between the clock domains is complete.

This bit is set when the synchronization of STATUS register between clock domains is started.

**Bit 2 – CTRLB** CTRLB Synchronization Busy

This bit is cleared when the synchronization of CTRLB register between the clock domains is complete.

This bit is set when the synchronization of CTRLB register between clock domains is started.

**Bit 1 – ENABLE** ENABLE Synchronization Busy

This bit is cleared when the synchronization of ENABLE bit between the clock domains is complete.

This bit is set when the synchronization of ENABLE bit between clock domains is started.

**Bit 0 – SWRST** SWRST Synchronization Busy

This bit is cleared when the synchronization of SWRST bit between the clock domains is complete.

This bit is set when the synchronization of SWRST bit between clock domains is started.

32.8.5 Fault Control A and B

**Name:** FCTRL  
**Offset:** 0x0C + n\*0x04 [n=0..1]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	FILTERVAL[3:0]							
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
	BLANKVAL[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
		CAPTURE[2:0]			CHSEL[1:0]		HALT[1:0]	
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]	
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

**Bits 27:24 – FILTERVAL[3:0]** Recoverable Fault n Filter Value

These bits define the filter value applied on MCEx (x=0,1) event input line. The value must be set to zero when MCEx event is used as synchronous event.

**Bits 23:16 – BLANKVAL[7:0]** Recoverable Fault n Blanking Value

These bits determine the duration of the blanking of the fault input source. Activation and edge selection of the blank filtering are done by the BLANK bits (FCTRLn.BLANK).

When enabled, the fault input source is internally disabled for BLANKVAL\* prescaled GCLK\_TCC periods after the detection of the waveform edge.

**Bits 14:12 – CAPTURE[2:0]** Recoverable Fault n Capture Action

These bits select the capture and Fault n interrupt/event conditions.

**Table 32-8. Fault n Capture Action**

Value	Name	Description
0x0	DISABLE	Capture on valid recoverable Fault n is disabled
0x1	CAPT	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0].  INTFLAG.FAULTn flag rises on each new captured value.

Value	Name	Description
0x2	CAPTMIN	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0], if COUNT value is lower than the last stored capture value (CC).  INTFLAG.FAULTn flag rises on each local minimum detection.
0x3	CAPTMAX	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0], if COUNT value is higher than the last stored capture value (CC).  INTFLAG.FAULTn flag rises on each local maximum detection.
0x4	LOCMIN	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0].  INTFLAG.FAULTn flag rises on each local minimum value detection.
0x5	LOCMAX	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0].  INTFLAG.FAULTn flag rises on each local maximum detection.
0x6	DERIV0	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0].  INTFLAG.FAULTn flag rises on each local maximum or minimum detection.

### Bits 11:10 – CHSEL[1:0] Recoverable Fault n Capture Channel

These bits select the channel for capture operation triggered by recoverable Fault n.

Value	Name	Description
0x0	CC0	Capture value stored into CC0
0x1	CC1	Capture value stored into CC1
0x2	CC2	Capture value stored into CC2
0x3	CC3	Capture value stored into CC3

### Bits 9:8 – HALT[1:0] Recoverable Fault n Halt Operation

These bits select the halt action for recoverable Fault n.

Value	Name	Description
0x0	DISABLE	Halt action disabled
0x1	HW	Hardware halt action
0x2	SW	Software halt action
0x3	NR	Non-recoverable fault

### Bit 7 – RESTART Recoverable Fault n Restart

Setting this bit enables restart action for Fault n.

Value	Description
0	Fault n restart action is disabled.
1	Fault n restart action is enabled.

### Bits 6:5 – BLANK[1:0] Recoverable Fault n Blanking Operation

These bits, select the blanking start point for recoverable Fault n.

Value	Name	Description
0x0	START	Blanking applied from start of the Ramp period
0x1	RISE	Blanking applied from rising edge of the waveform output
0x2	FALL	Blanking applied from falling edge of the waveform output
0x3	BOTH	Blanking applied from each toggle of the waveform output

### Bit 4 – QUAL Recoverable Fault n Qualification

Setting this bit enables the recoverable Fault n input qualification.

Value	Description
0	The recoverable Fault n input is not disabled on CMPx value condition.
1	The recoverable Fault n input is disabled when output signal is at inactive level (CMPx == 0).

### Bit 3 – KEEP Recoverable Fault n Keep

Setting this bit enables the Fault n keep action.

Value	Description
0	The Fault n state is released as soon as the recoverable Fault n is released.
1	The Fault n state is released at the end of TCC cycle.

### Bits 1:0 – SRC[1:0] Recoverable Fault n Source

These bits select the TCC event input for recoverable Fault n.

Event system channel connected to MCEx event input, must be configured to route the event asynchronously, when used as a recoverable Fault n input.

Value	Name	Description
0x0	DISABLE	Fault input disabled
0x1	ENABLE	MCEx (x=0,1) event input
0x2	INVERT	Inverted MCEx (x=0,1) event input
0x3	ALTFault	Alternate fault (A or B) state at the end of the previous period.

### 32.8.6 Waveform Extension Control

**Name:** WEXCTRL  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	DTHS[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
	DTLS[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
					DTIENx	DTIENx	DTIENx	DTIENx
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
							OTMX[1:0]	
Access							R/W	R/W
Reset							0	0

**Bits 31:24 – DTHS[7:0]** Dead-Time High Side Outputs Value

This register holds the number of GCLK\_TCC clock cycles for the dead-time high side.

**Bits 23:16 – DTLS[7:0]** Dead-time Low Side Outputs Value

This register holds the number of GCLK\_TCC clock cycles for the dead-time low side.

**Bits 11,10,9,8 – DTIENx** Dead-time Insertion Generator x Enable

Setting any of these bits enables the dead-time insertion generator for the corresponding output matrix. This will override the output matrix [x] and [x+WO\_NUM/2], with the low side and high side waveform respectively.

Value	Description
0	No dead-time insertion override.
1	Dead time insertion override on signal outputs[x] and [x+WO_NUM/2], from matrix outputs[x] signal.

**Bits 1:0 – OTMX[1:0]** Output Matrix

These bits define the matrix routing of the TCC waveform generation outputs to the port pins, according to [Table 32-4](#).

### 32.8.7 Driver Control

**Name:** DRVCTRL  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	FILTERVAL1[3:0]				FILTERVAL0[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	23	22	21	20	19	18	17	16
	INVENx	INVENx	INVENx	INVENx	INVENx	INVENx	INVENx	INVENx
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
	NRVx	NRVx	NRVx	NRVx	NRVx	NRVx	NRVx	NRVx
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
	NREx	NREx	NREx	NREx	NREx	NREx	NREx	NREx
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 31:28 – FILTERVAL1[3:0] Non-Recoverable Fault Input 1 Filter Value

These bits define the filter value applied on TCE1 event input line. When the TCE1 event input line is configured as a synchronous event, this value must be 0x0.

#### Bits 27:24 – FILTERVAL0[3:0] Non-Recoverable Fault Input 0 Filter Value

These bits define the filter value applied on TCE0 event input line. When the TCE0 event input line is configured as a synchronous event, this value must be 0x0.

#### Bits 23,22,21,20,19,18,17,16 – INVENx Waveform Output x Inversion

These bits are used to select inversion on the output of channel x.

Writing a '1' to INVENx inverts output from WO[x].

Writing a '0' to INVENx disables inversion of output from WO[x].

#### Bits 15,14,13,12,11,10,9,8 – NRVx NRVx Non-Recoverable State x Output Value

These bits define the value of the enabled override outputs, under non-recoverable fault condition.

#### Bits 7,6,5,4,3,2,1,0 – NREx Non-Recoverable State x Output Enable

These bits enable the override of individual outputs by NRVx value, under non-recoverable fault condition.

---

---

Value	Description
0	Non-recoverable fault tri-state the output.
1	Non-recoverable faults set the output to NRVx level.

### 32.8.8 Debug control

**Name:** DBGCTRL  
**Offset:** 0x1E  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### Bit 2 – FDDBD Fault Detection on Debug Break Detection

This bit is not affected by software reset and should not be changed by software while the TCC is enabled.

By default this bit is zero, and the on-chip debug (OCD) fault protection is disabled. When this bit is written to '1', OCD break request from the OCD system will trigger non-recoverable fault. When this bit is set, OCD fault protection is enabled and OCD break request from the OCD system will trigger a non-recoverable fault.

Value	Description
0	No faults are generated when TCC is halted in debug mode.
1	A non recoverable fault is generated and FAULTD flag is set when TCC is halted in debug mode.

#### Bit 0 – DBGRUN Debug Running State

This bit is not affected by software reset and should not be changed by software while the TCC is enabled.

Value	Description
0	The TCC is halted when the device is halted in debug mode.
1	The TCC continues normal operation when the device is halted in debug mode.



### 32.8.9 Event Control

**Name:** EVCTRL  
**Offset:** 0x20  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
					MCEOx	MCEOx	MCEOx	MCEOx
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					MCEIx	MCEIx	MCEIx	MCEIx
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	TCEIx	TCEIx	TCINVx	TCINVx		CNTEO	TRGEO	OVFEO
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0
Bit	7	6	5	4	3	2	1	0
	CNTSEL[1:0]		EVACT1[2:0]			EVACT0[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 27,26,25,24 – MCEOx Match or Capture Channel x Event Output Enable

These bits control if the Match/capture event on channel x is enabled and will be generated for every match or capture.

Value	Description
0	Match/capture x event is disabled and will not be generated.
1	Match/capture x event is enabled and will be generated for every compare/capture on channel x.

#### Bits 19,18,17,16 – MCEIx Match or Capture Channel x Event Input Enable

These bits indicate if the Match/capture x incoming event is enabled

These bits are used to enable match or capture input events to the CCx channel of TCC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

#### Bits 15,14 – TCEIx Timer/Counter Event Input x Enable

This bit is used to enable input event x to the TCC.

Value	Description
0	Incoming event x is disabled.
1	Incoming event x is enabled.

### Bits 13,12 – TCINVx Timer/Counter Event x Invert Enable

This bit inverts the event x input.

Value	Description
0	Input event source x is not inverted.
1	Input event source x is inverted.

### Bit 10 – CNTEO Timer/Counter Event Output Enable

This bit is used to enable the counter cycle event. When enabled, an event will be generated on begin or end of counter cycle depending of CNTSEL[1:0] settings.

Value	Description
0	Counter cycle output event is disabled and will not be generated.
1	Counter cycle output event is enabled and will be generated depend of CNTSEL[1:0] value.

### Bit 9 – TRGEO Retrigger Event Output Enable

This bit is used to enable the counter retrigger event. When enabled, an event will be generated when the counter retriggers operation.

Value	Description
0	Counter retrigger event is disabled and will not be generated.
1	Counter retrigger event is enabled and will be generated for every counter retrigger.

### Bit 8 – OVFE0 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 reaches the TOP or the ZERO value.

Value	Description
0	Overflow/underflow counter event is disabled and will not be generated.
1	Overflow/underflow counter event is enabled and will be generated for every counter overflow/underflow.

### Bits 7:6 – CNTSEL[1:0] Timer/Counter Interrupt and Event Output Selection

These bits define on which part of the counter cycle the counter event output is generated.

Value	Name	Description
0x0	BEGIN	An interrupt/event is generated at begin of each counter cycle
0x1	END	An interrupt/event is generated at end of each counter cycle
0x2	BETWEEN	An interrupt/event is generated between each counter cycle.
0x3	BOUNDARY	An interrupt/event is generated at begin of first counter cycle, and end of last counter cycle.

### Bits 5:3 – EVACT1[2:0] Timer/Counter Event Input 1 Action

These bits define the action the TCC will perform on TCE1 event input.

Value	Name	Description
0x0	OFF	Event action disabled.
0x1	RETRIGGER	Start restart or re-trigger TC on event

Value	Name	Description
0x2	DIR (asynch)	Direction control
0x3	STOP	Stop TC on event
0x4	DEC	Decrement TC on event
0x5	PPW	Period captured into CC0 Pulse Width on CC1
0x6	PWP	Period captured into CC1 Pulse Width on CC0
0x7	FAULT	Non-recoverable Fault

### Bits 2:0 – EVACT0[2:0] Timer/Counter Event Input 0 Action

These bits define the action the TCC will perform on TCE0 event input 0.

Value	Name	Description
0x0	OFF	Event action disabled.
0x1	RETRIGGER	Start restart or re-trigger TC on event
0x2	COUNTEV	Count on event.
0x3	START	Start TC on event
0x4	INC	Increment TC on EVENT
0x5	COUNT (asynch)	Count on active state of asynchronous event
0x6	-	Reserved
0x7	FAULT	Non-recoverable Fault

### 32.8.10 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x24  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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 (INTENSET) register.

Bit	23	22	21	20	19	18	17	16
					MCx	MCx	MCx	MCx
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULTx	FAULTx	FAULTB	FAULTA	DFS			
Access	R/W	R/W	R/W	R/W	R/W			
Reset	0	0	0	0	0			
Bit	7	6	5	4	3	2	1	0
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### Bits 19,18,17,16 – MCx Match or Capture Channel x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the corresponding Match or Capture Channel x Interrupt Disable/Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

#### Bits 15,14 – FAULTx Non-Recoverable Fault x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

#### Bit 13 – FAULTB Recoverable Fault B Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Recoverable Fault B Interrupt Disable/Enable bit, which disables the Recoverable Fault B interrupt.

Value	Description
0	The Recoverable Fault B interrupt is disabled.
1	The Recoverable Fault B interrupt is enabled.

### Bit 12 – FAULTA Recoverable Fault A Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Recoverable Fault A Interrupt Disable/Enable bit, which disables the Recoverable Fault A interrupt.

Value	Description
0	The Recoverable Fault A interrupt is disabled.
1	The Recoverable Fault A interrupt is enabled.

### Bit 11 – DFS Non-Recoverable Debug Fault Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Debug Fault State Interrupt Disable/Enable bit, which disables the Debug Fault State interrupt.

Value	Description
0	The Debug Fault State interrupt is disabled.
1	The Debug Fault State interrupt is enabled.

### Bit 3 – ERR Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Error Interrupt Disable/Enable bit, which disables the Compare interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

### Bit 2 – CNT Counter Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Counter Interrupt Disable/Enable bit, which disables the Counter interrupt.

Value	Description
0	The Counter interrupt is disabled.
1	The Counter interrupt is enabled.

### Bit 1 – TRG Retrigger Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Retrigger Interrupt Disable/Enable bit, which disables the Retrigger interrupt.

Value	Description
0	The Retrigger interrupt is disabled.
1	The Retrigger interrupt is enabled.

**Bit 0 – OVF** Overflow Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Overflow Interrupt Disable/Enable bit, which disables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

### 32.8.11 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x28  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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	23	22	21	20	19	18	17	16
					MCx	MCx	MCx	MCx
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULTx	FAULTx	FAULTB	FAULTA	DFS			
Access	R/W	R/W	R/W	R/W	R/W			
Reset	0	0	0	0	0			
Bit	7	6	5	4	3	2	1	0
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### Bits 19,18,17,16 – MCx Match or Capture Channel x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the corresponding Match or Capture Channel x Interrupt Disable/Enable bit, which enables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

#### Bits 15,14 – FAULTx Non-Recoverable Fault x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Non-Recoverable Fault x Interrupt Disable/Enable bit, which enables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

#### Bit 13 – FAULTB Recoverable Fault B Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Recoverable Fault B Interrupt Disable/Enable bit, which enables the Recoverable Fault B interrupt.

Value	Description
0	The Recoverable Fault B interrupt is disabled.
1	The Recoverable Fault B interrupt is enabled.

### Bit 12 – FAULTA Recoverable Fault A Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Recoverable Fault A Interrupt Disable/Enable bit, which enables the Recoverable Fault A interrupt.

Value	Description
0	The Recoverable Fault A interrupt is disabled.
1	The Recoverable Fault A interrupt is enabled.

### Bit 11 – DFS Non-Recoverable Debug Fault Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Debug Fault State Interrupt Disable/Enable bit, which enables the Debug Fault State interrupt.

Value	Description
0	The Debug Fault State interrupt is disabled.
1	The Debug Fault State interrupt is enabled.

### Bit 3 – ERR Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Error Interrupt Disable/Enable bit, which enables the Compare interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

### Bit 2 – CNT Counter Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Retrigger Interrupt Disable/Enable bit, which enables the Counter interrupt.

Value	Description
0	The Counter interrupt is disabled.
1	The Counter interrupt is enabled.

### Bit 1 – TRG Retrigger Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Retrigger Interrupt Disable/Enable bit, which enables the Retrigger interrupt.

Value	Description
0	The Retrigger interrupt is disabled.
1	The Retrigger interrupt is enabled.



**Bit 0 – OVF** Overflow Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Overflow Interrupt Disable/Enable bit, which enables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

### 32.8.12 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x2C  
**Reset:** 0x00000000  
**Property:** -

Bit	23	22	21	20	19	18	17	16
					MCx	MCx	MCx	MCx
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULTx	FAULTx	FAULTB	FAULTA	DFS			
Access	R/W	R/W	R/W	R/W	R/W			
Reset	0	0	0	0	0			
Bit	7	6	5	4	3	2	1	0
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### Bits 19,18,17,16 – MCx Match or Capture Channel x Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a match with the compare condition or once CCx register contain a valid capture value.

Writing a '0' to one of these bits has no effect.

Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag

In Capture operation, this flag is automatically cleared when CCx register is read.

#### Bits 15,14 – FAULTx Non-Recoverable Fault x Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a Non-Recoverable Fault x occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Non-Recoverable Fault x interrupt flag.

#### Bit 13 – FAULTB Recoverable Fault B Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a Recoverable Fault B occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Recoverable Fault B interrupt flag.

#### Bit 12 – FAULTA Recoverable Fault A Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a Recoverable Fault B occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Recoverable Fault B interrupt flag.

**Bit 11 – DFS** Non-Recoverable Debug Fault State Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after an Debug Fault State occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Debug Fault State interrupt flag.

**Bit 3 – ERR** Error Interrupt Flag

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 '0' to this bit has no effect.

Writing a '1' to this bit clears the error interrupt flag.

**Bit 2 – CNT** Counter Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a counter event occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the CNT interrupt flag.

**Bit 1 – TRG** Retrigger Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a counter retrigger occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the re-trigger interrupt flag.

**Bit 0 – OVF** Overflow Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after an overflow condition occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Overflow interrupt flag.

### 32.8.13 Status

**Name:** STATUS  
**Offset:** 0x30  
**Reset:** 0x00000001  
**Property:** -

Bit	31	30	29	28	27	26	25	24
					CMPx	CMPx	CMPx	CMPx
Access					R	R	R	R
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					CCBVx	CCBVx	CCBVx	CCBVx
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULTx	FAULTx	FAULTB	FAULTA	FAULT1IN	FAULT0IN	FAULTBIN	FAULTAIN
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	PERBV	WAVEBV	PATTBV		DFS		IDX	STOP
Access	R/W	R/W	R/W		R/W		R	R
Reset	0	0	0		0		0	1

#### Bits 27,26,25,24 – CMPx Channel x Compare Value

This bit reflects the channel x output compare value.

Value	Description
0	Channel compare output value is 0.
1	Channel compare output value is 1.

#### Bits 19,18,17,16 – CCBVx Channel x Compare or Capture Buffer Valid

For a compare channel, this bit is set when a new value is written to the corresponding CCBx register. The bit is cleared either by writing a '1' to the corresponding location when CTRLB.LUPD is set, or automatically on an UPDATE condition.

For a capture channel, the bit is set when a valid capture value is stored in the CCBx register. The bit is automatically cleared when the CCx register is read.

#### Bits 15,14 – FAULTx Non-recoverable Fault x State

This bit is set by hardware as soon as non-recoverable Fault x condition occurs.

This bit is cleared by writing a one to this bit and when the corresponding FAULTxIN status bit is low.

Once this bit is clear, the timer/counter will restart from the last COUNT value. To restart the timer/counter from BOTTOM, the timer/counter restart command must be executed before clearing the corresponding

STATEx bit. For further details on timer/counter commands, refer to available commands description ([CTRLBSET.CMD](#)).

**Bit 13 – FAULTB** Recoverable Fault B State

This bit is set by hardware as soon as recoverable Fault B condition occurs.

This bit can be clear by hardware when Fault B action is resumed, or by writing a '1' to this bit when the corresponding FAULTBIN bit is low. If software halt command is enabled (FAULTB.HALT=SW), clearing this bit will release the timer/counter.

**Bit 12 – FAULTA** Recoverable Fault A State

This bit is set by hardware as soon as recoverable Fault A condition occurs.

This bit can be clear by hardware when Fault A action is resumed, or by writing a '1' to this bit when the corresponding FAULTAIN bit is low. If software halt command is enabled (FAULTA.HALT=SW), clearing this bit will release the timer/counter.

**Bit 11 – FAULT1IN** Non-Recoverable Fault 1 Input

This bit is set while an active Non-Recoverable Fault 1 input is present.

**Bit 10 – FAULT0IN** Non-Recoverable Fault 0 Input

This bit is set while an active Non-Recoverable Fault 0 input is present.

**Bit 9 – FAULTBIN** Recoverable Fault B Input

This bit is set while an active Recoverable Fault B input is present.

**Bit 8 – FAULTAIN** Recoverable Fault A Input

This bit is set while an active Recoverable Fault A input is present.

**Bit 7 – PERBV** Period Buffer Valid

This bit is set when a new value is written to the PERB register. This bit is automatically cleared by hardware on UPDATE condition when CTRLB.LUPD is set, or by writing a '1' to this bit.

**Bit 6 – WAVEBV** Waveform Control Buffer Valid

This bit is set when a new value is written to the WAVEB register. This bit is automatically cleared by hardware on UPDATE condition when CTRLB.LUPD is set, or by writing a '1' to this bit.

**Bit 5 – PATTBV** Pattern Generator Value Buffer Valid

This bit is set when a new value is written to the PATTB register. This bit is automatically cleared by hardware on UPDATE condition when CTRLB.LUPD is set, or by writing a '1' to this bit.

**Bit 3 – DFS** Debug Fault State

This bit is set by hardware in debug mode when DDBGCTRL.FDDBD bit is set. The bit is cleared by writing a '1' to this bit and when the TCC is not in debug mode.

When the bit is set, the counter is halted and the waveforms state depend on DRVCTRL.NRE and DRVCTRL.NRV registers.

**Bit 1 – IDX** Ramp Index

In RAMP2 and RAMP2A operation, the bit is cleared during the cycle A and set during the cycle B. In RAMP1 operation, the bit always reads zero. For details on ramp operations, refer to [Ramp Operations](#).

**Bit 0 – STOP** Stop

This bit is set when the TCC is disabled either on a STOP command or on an UPDATE condition when One-Shot operation mode is enabled (CTRLBSET.ONESHOT=1).

This bit is clear on the next incoming counter increment or decrement.

Value	Description
0	Counter is running.
1	Counter is stopped.

### 32.8.14 Counter Value

**Name:** COUNT  
**Offset:** 0x34  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

**Note:** Prior to any read access, this register must be synchronized by user by writing the according TCC Command value to the Control B Set register (CTRLBSET.CMD=READSYNC).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
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 23:0 – COUNT[23:0] Counter Value

These bits hold the value of the counter register.

**Note:** When the TCC is configured as 16-bit timer/counter, the excess bits are read zero.

**Note:** This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0 (depicted)
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6

### 32.8.15 Pattern

**Name:** PATT  
**Offset:** 0x38  
**Reset:** 0x0000  
**Property:** Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	PGV0[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
	PGE0[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 8:15, 16:23, 24:31, 32:39, 40:47, 48:55, 56:63, 64:71 – PGV** Pattern Generation Output Value  
 This register holds the values of pattern for each waveform output.

**Bits 0:7, 8:15, 16:23, 24:31, 32:39, 40:47, 48:55, 56:63 – PGE** Pattern Generation Output Enable  
 This register holds the enable status of pattern generation for each waveform output. A bit written to '1' will override the corresponding SWAP output with the corresponding PGVn value.



### 32.8.16 Waveform

**Name:** WAVE  
**Offset:** 0x3C  
**Reset:** 0x00000000  
**Property:** Write-Synchronized

Bit	31	30	29	28	27	26	25	24
					SWAP3	SWAP2	SWAP1	SWAP0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					POL3	POL2	POL1	POL0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
					CICCEN3	CICCEN2	CICCEN1	CICCEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	CIPEREN					WAVEGEN[2:0]		
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

**Bits 24, 25, 26, 27 – SWAP** Swap DTI Output Pair x  
 Setting these bits enables output swap of DTI outputs [x] and [x+WO\_NUM/2]. Note the DTIxEN settings will not affect the swap operation.

**Bits 16, 17, 18, 19 – POL** Channel Polarity x  
 Setting these bits enables the output polarity in single-slope and dual-slope PWM operations.

Value	Name	Description
0	(single-slope PWM waveform generation)	Compare output is initialized to ~DIR and set to DIR when TCC counter matches CCx value
1	(single-slope PWM waveform generation)	Compare output is initialized to DIR and set to ~DIR when TCC counter matches CCx value.
0	(dual-slope PWM waveform generation)	Compare output is set to ~DIR when TCC counter matches CCx value
1	(dual-slope PWM waveform generation)	Compare output is set to DIR when TCC counter matches CCx value.

**Bits 8, 9, 10, 11 – CICCEN** Circular CC Enable x  
 Setting this bits enables the compare circular buffer option on channel. When the bit is set, CCx register value is copied-back into the CCx register on UPDATE condition.

### Bit 7 – CIPEREN Circular Period Enable

Setting this bits enable the period circular buffer option. When the bit is set, the PER register value is copied-back into the PERB register on UPDATE condition.

These bits select Ramp operation (RAMP). These bits are not synchronized.

Value	Name	Description
0x0	RAMP1	RAMP1 operation
0x1	RAMP2A	Alternative RAMP2 operation
0x2	RAMP2	RAMP2 operation
0x3	-	Reserved
0x4		

### Bits 2:0 – WAVEGEN[2:0] Waveform Generation Operation

These bits select the waveform generation operation. The settings impact the top value and control if frequency or PWM waveform generation should be used. These bits are not synchronized.

Value	Name	Description						
		Operation	Top	Update	Waveform Output On Match	Waveform Output On Update	OVFIF/Event Up Down	
0x0	NFRQ	Normal Frequency	PER	TOP/Zero	Toggle	Stable	TOP	Zero
0x1	MFRQ	Match Frequency	CC0	TOP/Zero	Toggle	Stable	TOP	Zero
0x2	NPWM	Normal PWM	PER	TOP/Zero	Set	Clear	TOP	Zero
0x3								
0x4	DSCRITICAL	Dual-slope PWM	PER	Zero	~DIR	Stable	–	Zero
0x5	DSBOTTOM	Dual-slope PWM	PER	Zero	~DIR	Stable	–	Zero
0x6	DSBOTH	Dual-slope PWM	PER	TOP & Zero	~DIR	Stable	TOP	Zero
0x7	DSTOP	Dual-slope PWM	PER	Zero	~DIR	Stable	TOP	–

### 32.8.17 Period Value

**Name:** PER  
**Offset:** 0x40  
**Reset:** 0xFFFFFFFF  
**Property:** Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access	PER[17:10]							
Reset	1	1	1	1	1	1	1	1
Bit	15	14	13	12	11	10	9	8
Access	PER[9:2]							
Reset	1	1	1	1	1	1	1	1
Bit	7	6	5	4	3	2	1	0
Access	PER[1:0]		DITHER[5:0]					
Reset	1	1	1	1	1	1	1	1

#### Bits 23:6 – PER[17:0] Period Value

These bits hold the value of the period buffer register.

**Note:** When the TCC is configured as 16-bit timer/counter, the excess bits are read zero.

**Note:** This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6 (depicted)

#### Bits 5:0 – DITHER[5:0] Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM pulse period every 64 PWM frames.

**Note:** This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

<b>CTRLA.RESOLUTION</b>	<b>Bits [n:0]</b>
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

**32.8.18 Compare/Capture Channel x**

**Name:** CC  
**Offset:** 0x44 + n\*0x04 [n=0..3]  
**Reset:** 0x00000000  
**Property:** Write-Synchronized, Read-Synchronized

The CCx register represents the 16-, 24- bit value, CCx. The register has two functions, depending of the mode of operation.

For capture operation, this register represents the second buffer level and access point for the CPU and DMA.

For compare operation, this register is continuously compared to the counter value. Normally, the output form the comparator is then used for generating waveforms.

CCx register is updated with the buffer value from their corresponding CCBx register when an UPDATE condition occurs.

In addition, in match frequency operation, the CC0 register controls the counter period.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	CC[17:10]							
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[9:2]							
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
	CC[1:0]		DITHER[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 23:6 – CC[17:0]** Channel x Compare/Capture Value

These bits hold the value of the Channel x compare/capture register.

**Note:** When the TCC is configured as 16-bit timer/counter, the excess bits are read zero.

**Note:** This bit field occupies the m MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6 (depicted)

### Bits 5:0 – DITHER[5:0] Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM pulse width every 64 PWM frames.

**Note:** This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [n:0]
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

### 32.8.19 Pattern Buffer

**Name:** PATTB  
**Offset:** 0x64  
**Reset:** 0x0000  
**Property:** Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
PGVB0[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
PGE0[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 8:15, 16:23, 24:31, 32:39, 40:47, 48:55, 56:63, 64:71 – PGVB** Pattern Generation Output Value Buffer

This register is the buffer for the PGV register. If double buffering is used, valid content in this register is copied to the PGV register on an UPDATE condition.

**Bits 0:7, 8:15, 16:23, 24:31, 32:39, 40:47, 48:55, 56:63 – PGE0** Pattern Generation Output Enable Buffer

This register is the buffer of the PGE register. If double buffering is used, valid content in this register is copied into the PGE register at an UPDATE condition.

### 32.8.20 Waveform Buffer

**Name:** WAVEB  
**Offset:** 0x68  
**Reset:** 0x00000000  
**Property:** Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
					SWAPB 3	SWAPB 2	SWAPB 1	SWAPB 0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					POLB3	POLB2	POLB1	POLB0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
					CICCENB3	CICCENB2	CICCENB1	CICCENB0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	CIPERENB		RAMPB[1:0]			WAVEGENB[2:0]		
Access	R/W		R/W	R/W		R/W	R/W	R/W
Reset	0		0	0		0	0	0

**Bits 24, 25, 26, 27 – SWAPB** Swap DTI output pair x Buffer

These register bits are the buffer bits for the SWAP register bits. If double buffering is used, valid content in these bits is copied to the corresponding SWAPx bits on an UPDATE condition.

**Bits 16, 17, 18, 19 – POLB** Channel Polarity x Buffer

These register bits are the buffer bits for POLx register bits. If double buffering is used, valid content in these bits is copied to the corresponding POBx bits on an UPDATE condition.

**Bits 8, 9, 10, 11 – CICCENB** Circular CCx Buffer Enable

These register bits are the buffer bits for CICCENx register bits. If double buffering is used, valid content in these bits is copied to the corresponding CICCENx bits on a UPDATE condition.

**Bit 7 – CIPERENB** Circular Period Enable Buffer

This register bit is the buffer bit for CIPEREN register bit. If double buffering is used, valid content in this bit is copied to the corresponding CIPEREN bit on a UPDATE condition.

**Bits 5:4 – RAMPB[1:0]** Ramp Operation Buffer

These register bits are the buffer bits for RAMP register bits. If double buffering is used, valid content in these bits is copied to the corresponding RAMP bits on a UPDATE condition.



**Bits 2:0 – WAVEGENB[2:0]** Waveform Generation Operation Buffer

These register bits are the buffer bits for WAVEGEN register bits. If double buffering is used, valid content in these bits is copied to the corresponding WAVEGEN bits on a UPDATE condition.

### 32.8.21 Period Buffer Value

**Name:** PERB  
**Offset:** 0x6C  
**Reset:** 0xFFFFFFFF  
**Property:** Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	PERB[17:10]							
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
Bit	15	14	13	12	11	10	9	8
	PERB[9:2]							
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
Bit	7	6	5	4	3	2	1	0
	PERB[1:0]		DITHERB[5:0]					
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 23:6 – PERB[17:0] Period Buffer Value

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition.

**Note:** When the TCC is configured as 16-bit timer/counter, the excess bits are read zero.

**Note:** This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6 (depicted)

#### Bits 5:0 – DITHERB[5:0] Dithering Buffer Cycle Number

These bits represent the PER.DITHER bits buffer. When the double buffering is enabled, the value of this bit field is copied to the PER.DITHER bits on an UPDATE condition.

**Note:** This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

<b>CTRLA.RESOLUTION</b>	<b>Bits [n:0]</b>
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

### 32.8.22 Channel x Compare/Capture Buffer Value

**Name:** CCB  
**Offset:** 0x70 + n\*0x04 [n=0..3]  
**Reset:** 0x00000000  
**Property:** Write-Synchronized, Read-Synchronized

CCBx is copied into CCx at TCC update time

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	CCB[17:10]							
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
	CCB[9:2]							
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
	CCB[1:0]		DITHERB[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 23:6 – CCB[17:0] Channel x Compare/Capture Buffer Value

These bits hold the value of the Channel x Compare/Capture Buffer Value register. The register serves as the buffer for the associated compare or capture registers (CCx). Accessing this register using the CPU or DMA will affect the corresponding CCBVx status bit.

**Note:** When the TCC is configured as 16-bit timer/counter, the excess bits are read zero.

**Note:** This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6 (depicted)

#### Bits 5:0 – DITHERB[5:0] Dithering Buffer Cycle Number

These bits represent the CCx.DITHER bits buffer. When the double buffering is enable, DITHERBUF bits value is copied to the CCx.DITHER bits on an UPDATE condition.

**Note:** This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [n:0]
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

## 33. ADC – Analog-to-Digital Converter

### 33.1 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.

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. It is possible to use DMA to move ADC results directly to memory or peripherals when conversions are done.

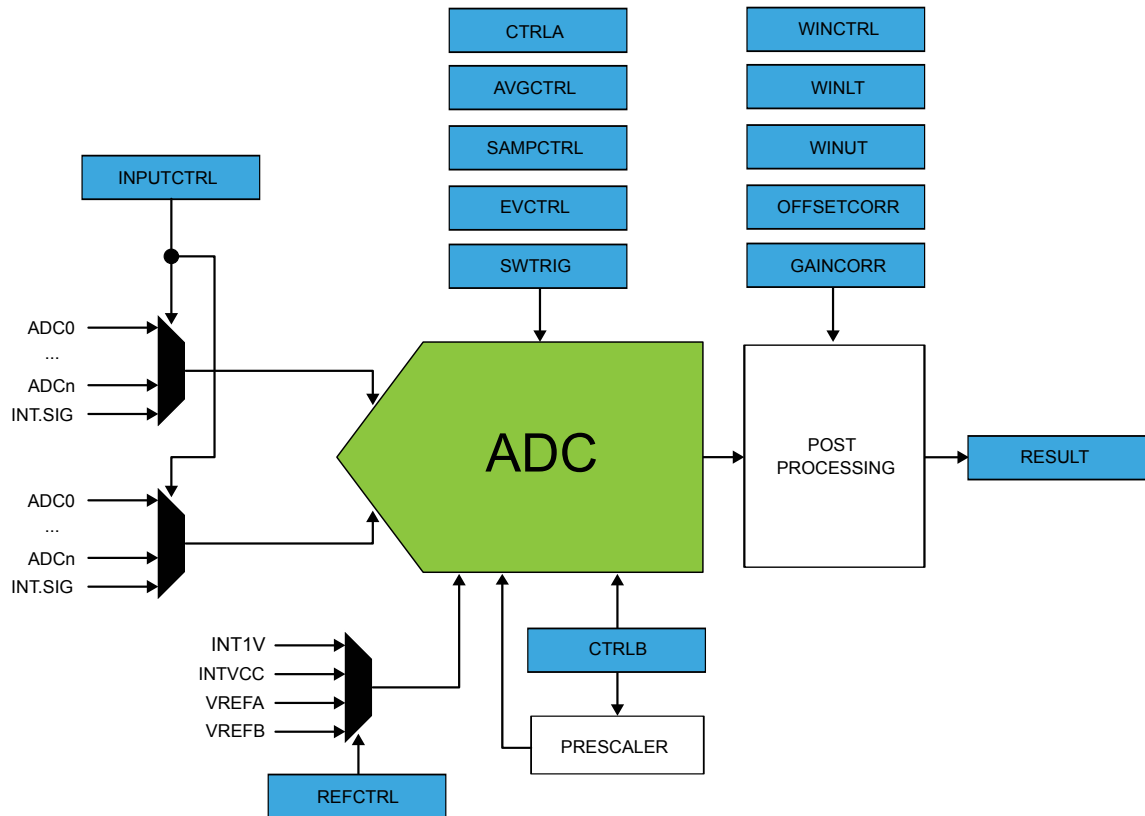
### 33.2 Features

- 8-, 10- or 12-bit resolution
- Up to 350,000 samples per second (350ksps)
- Differential and single-ended inputs
  - Up to analog input
  - positive and negative, including internal and external
- internal inputs
  - Bandgap
  - DAC
  - 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)
- Optional DMA transfer of conversion result
- Hardware gain and offset compensation

- Averaging and oversampling with decimation to support, up to 16-bit result
- Selectable sampling time

### 33.3 Block Diagram

Figure 33-1. ADC Block Diagram



### 33.4 Signal Description

Signal Name	Type	Description
VREFA	Analog input	External reference voltage A
VREFB	Analog input	External reference voltage B
ADC[19..0] <sup>(1)</sup>	Analog input	Analog input channels

**Note:** Refer to *Configuration Summary* for details on exact number of analog input channels.

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

#### Related Links

[I/O Multiplexing and Considerations](#)

### **33.5 Product Dependencies**

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

#### **33.5.1 I/O Lines**

Using the ADC's I/O lines requires the I/O pins to be configured using the port configuration (PORT).

**Related Links**

[PORT - I/O Pin Controller](#)

#### **33.5.2 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. Events connected to the event system can trigger other operations in the system without exiting sleep modes.

**Related Links**

[PM – Power Manager](#)

#### **33.5.3 Clocks**

The ADC bus clock (CLK\_APB\_ADCx) can be enabled in the Main Clock, which also defines the default state.

The ADC requires a generic clock (GCLK\_ADC). This clock must be configured and enabled in the Generic Clock Controller (GCLK) before using the ADC.

A generic clock is asynchronous to the bus clock. Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to *Synchronization* for further details.

**Related Links**

[Peripheral Clock Masking](#)

[GCLK - Generic Clock Controller](#)

#### **33.5.4 DMA**

The DMA request line is connected to the DMA Controller (DMAC). Using the ADC DMA requests requires the DMA Controller to be configured first.

**Related Links**

[DMAC – Direct Memory Access Controller](#)

#### **33.5.5 Interrupts**

The interrupt request line is connected to the interrupt controller. Using the ADC interrupt requires the interrupt controller to be configured first.

**Related Links**

[Nested Vector Interrupt Controller](#)

#### **33.5.6 Events**

The events are connected to the Event System.

**Related Links**

[EVSYS – Event System](#)



### 33.5.7 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.

### 33.5.8 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 (INTFLAG) register

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

PAC write-protection does not apply to accesses through an external debugger.

#### Related Links

[PAC - Peripheral Access Controller](#)

### 33.5.9 Analog Connections

I/O-pins AIN0 to AIN19 as well as the VREFA/VREFB reference voltage pin are analog inputs to the ADC.

### 33.5.10 Calibration

The BIAS and LINEARITY calibration values 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.

#### Related Links

[NVM Software Calibration Area Mapping](#)

## 33.6 Functional Description

### 33.6.1 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 or external (connected I/O pins). The user can also configure whether the conversion should be single-ended or differential.

### 33.6.2 Basic Operation

#### 33.6.2.1 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* 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).

#### Related Links

[SYSCTRL – System Controller](#)

### 33.6.2.2 Enabling, Disabling and Reset

The ADC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The ADC is disabled by writing CTRLA.ENABLE=0. The ADC is reset by writing a '1' 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.

The ADC must be disabled before it is reset.

### 33.6.2.3 Operation

In the most basic configuration, the ADC samples values from the configured internal or external sources (INPUTCTRL register). The rate of the conversion depends on the combination of the GCLK\_ADCx frequency and the clock prescaler.

To convert analog values to digital values, the ADC needs to be initialized first, as described in [Initialization](#). Data conversion can be started either manually by setting the Start bit in the Software Trigger register (SWTRIG.START=1), or automatically by configuring an automatic trigger to initiate the conversions. A free-running mode can be used to continuously convert an input channel. When using free-running mode the first conversion must be started, while subsequent conversions will start automatically at the end of previous conversions.

The automatic trigger can be configured to trigger on many different conditions.

The result of the conversion is stored in the Result register (RESULT) 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 soon as it is 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). When the RESRDY interrupt flag is set, the new result has been synchronized to the RESULT register.

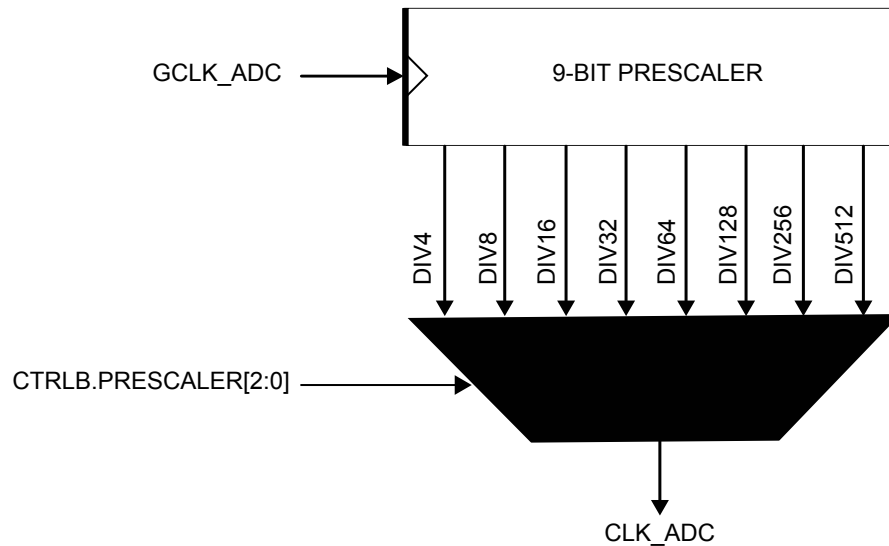
To enable one of the available interrupts sources, the corresponding bit in the Interrupt Enable Set register (INTENSET) must be written to '1'.

### 33.6.3 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 33-2. ADC Prescaler**



The propagation delay of an ADC measurement depends on the selected mode and is given by:

- Single-shot mode:

$$\text{PropagationDelay} = \frac{1 + \frac{\text{Resolution}}{2} + \text{DelayGain}}{f_{\text{CLK+ - ADC}}}$$

- Free-running mode:

$$\text{PropagationDelay} = \frac{\frac{\text{Resolution}}{2} + \text{DelayGain}}{f_{\text{CLK+ - ADC}}}$$

**Table 33-1. 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

### 33.6.4 ADC Resolution

The ADC supports 8-bit, 10-bit or 12-bit resolution. Resolution can be changed by writing the Resolution bit group in the Control B register (CTRLB.RESSEL). By default, the ADC resolution is set to 12 bits.

### 33.6.5 Differential and Single-Ended Conversions

The ADC has two conversion options: differential and single-ended:

- If the positive input may go below the negative input, the **differential** mode should be used in order to get correct results.
- If the positive input is always positive, the **single-ended** conversion should be used in order to have full 12-bit resolution in the conversion.

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 the Control B (CTRLB) register 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 differential mode is enabled by setting DIFFMODE bit in the Control B register (CTRLB.DIFFMODE). Both conversion types could be run in single mode or in free-running mode. When the free-running mode is selected, an ADC input will continuously sample the input and performs a new conversion. The INTFLAG.RESRDY bit will be set at the end of each conversion.

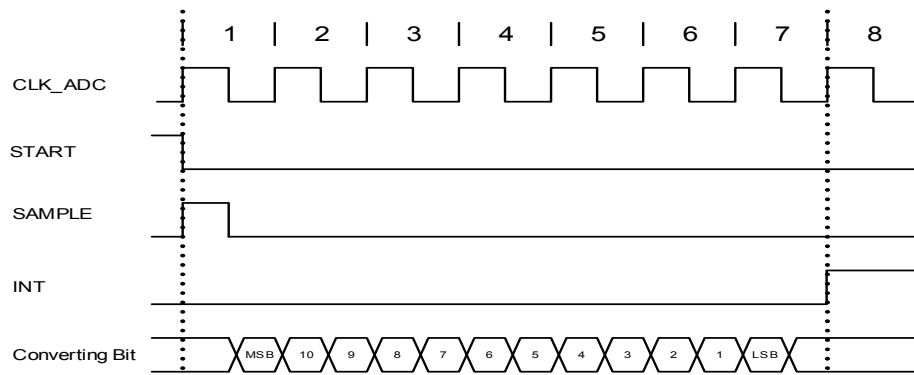
#### Related Links

[CTRLB](#)

#### 33.6.5.1 Conversion Timing

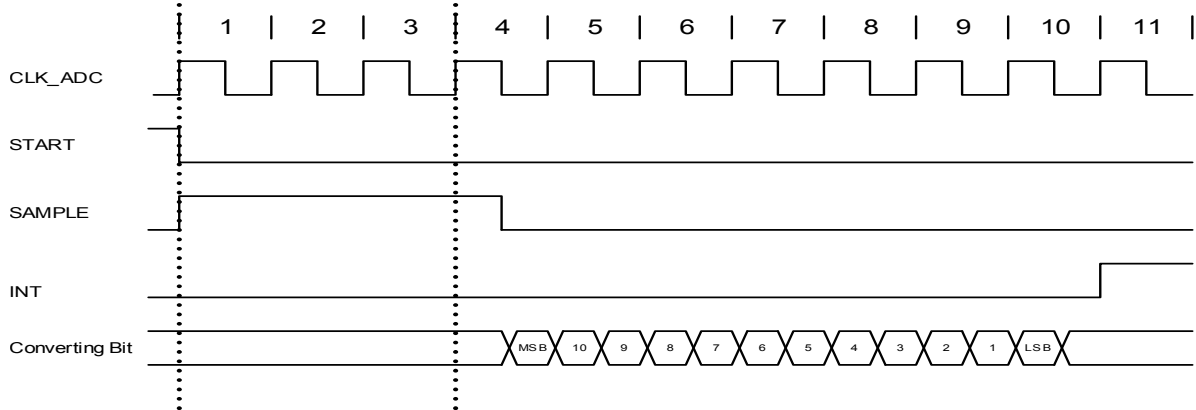
The following figure shows the ADC timing for one single conversion. A conversion starts after the software or event start are synchronized with the GCLK\_ADC clock. The input channel is sampled in the first half CLK\_ADC period.

**Figure 33-3. ADC Timing for One Conversion in Differential Mode without Gain**

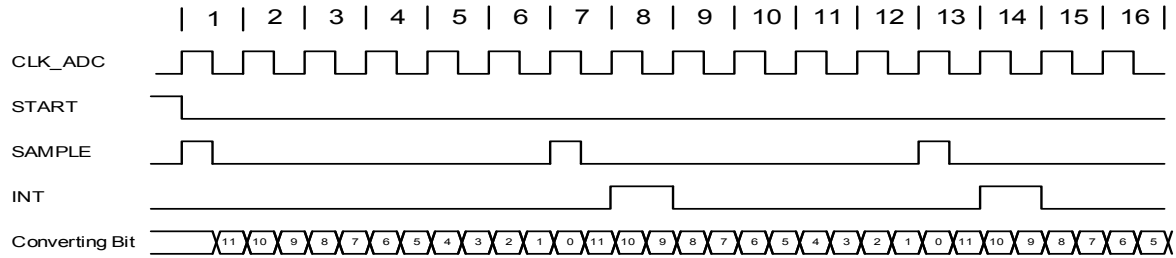


The sampling time can be increased by using the Sampling Time Length bit group in the Sampling Time Control register (SAMPCTRL.SAMPLEN). As example, the next figure is showing the timing conversion.

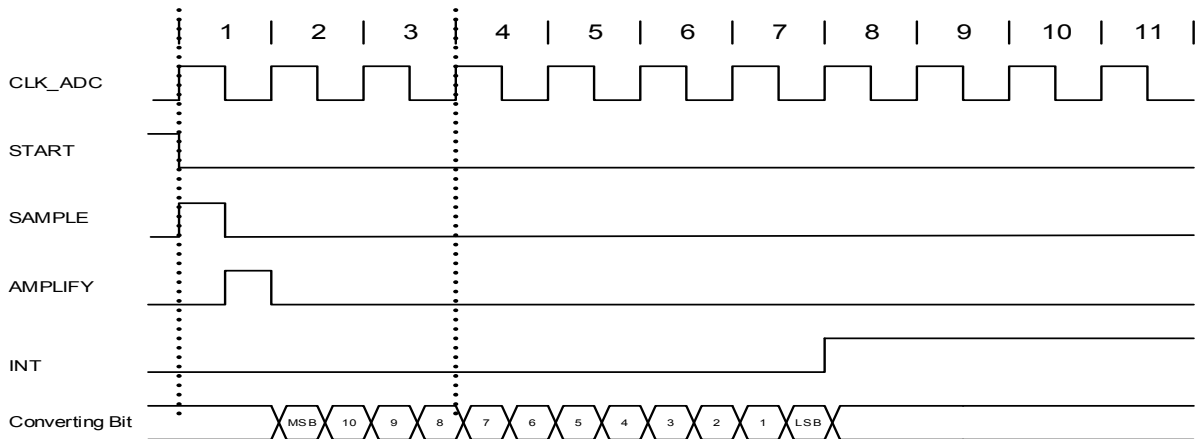
**Figure 33-4. ADC Timing for One Conversion in Differential Mode without Gain, but with Increased Sampling Time**



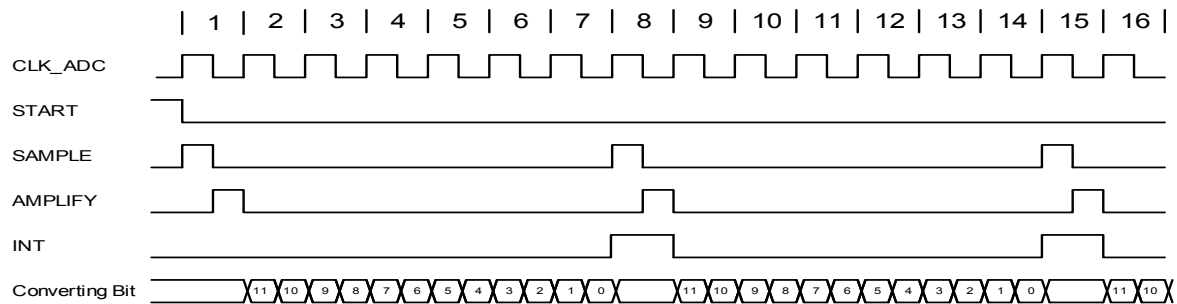
**Figure 33-5. ADC Timing for Free Running in Differential Mode without Gain**



**Figure 33-6. ADC Timing for One Conversion in Single-Ended Mode without Gain**



**Figure 33-7. ADC Timing for Free Running in Single-Ended Mode without Gain**



### 33.6.6 Accumulation

The result from multiple consecutive conversions can be accumulated. The number of samples to be accumulated is specified by the Number of Samples to be Collected field in the Average Control register (AVGCTRL.SAMPLENUM). When accumulating more than 16 samples, the result will be too large to match the 16-bit RESULT register size. To avoid overflow, the result is right shifted automatically to fit within the available register size. The number of automatic right shifts is specified in the table below.

**Note:** To perform the accumulation of two or more samples, the Conversion Result Resolution field in the Control B register (CTRLB.RESSEL) must be set.

**Table 33-2. 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

### 33.6.7 Averaging

Averaging is a feature that increases the sample accuracy, at the cost of a reduced sampling rate. This feature is suitable when operating in noisy conditions.

Averaging is done by accumulating  $m$  samples, as described in [Accumulation](#), and dividing 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.

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).

**Note:** To perform the averaging of two or more samples, the Conversion Result Resolution field in the Control B register (CTRLB.RESSEL) must be set to '1'.

Averaging AVGCTRL.SAMPLENUM samples will reduce the un-averaged sampling rate by a factor

$$\frac{1}{\text{AVGCTRL.SAMPLENUM}}$$

When the averaged result is available, the INTFLAG.RESRDY bit will be set.

**Table 33-3. 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

### 33.6.8 Oversampling and Decimation

By using oversampling and decimation, the ADC resolution can be increased from 12 bits up to 16 bits, for the cost of reduced effective sampling rate.

To increase the resolution by  $n$  bits,  $4^n$  samples must be accumulated. The result must then be right-shifted 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 33-4. 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

### 33.6.9 Window Monitor

The window monitor feature allows the conversion result in the RESULT register to be compared to predefined threshold values. The window mode is selected by setting the Window Monitor Mode bits in the Window Monitor Control register (WINCTRL.WINMODE[2:0]). Threshold values must be written in 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. 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 e.g. in 8-bit mode, 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.

### 33.6.10 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 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).

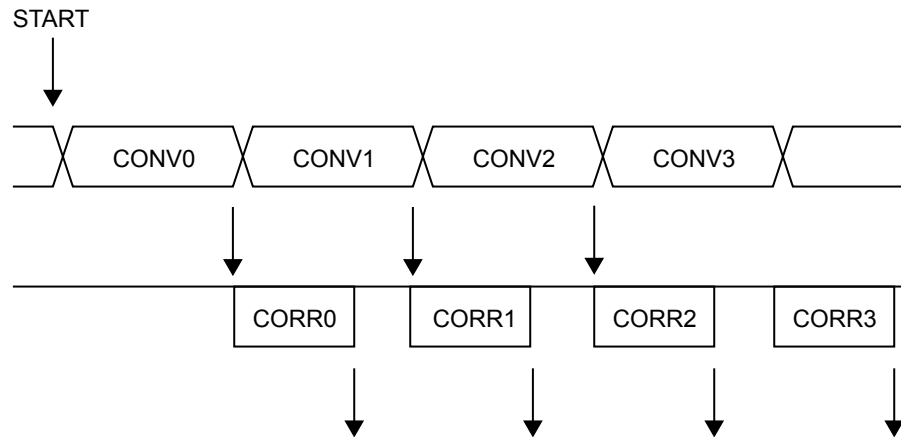
To correct these two errors, the Digital Correction Logic Enabled bit in the Control B register (CTRLB.CORREN) must be set to "1".

Offset and gain error compensation results are both calculated according to:

$$\text{Result} = (\text{Conversion value} + \text{OFFSETCORR}) \cdot \text{GAINCORR}$$

The correction will introduce a latency of 13 CLK\_ADC clock cycles. In free running mode this latency is introduced on the first conversion only, since its duration is always less than the propagation delay. In single conversion mode this latency is introduced for each conversion.

**Figure 33-8. ADC Timing Correction Enabled**



### 33.6.11 DMA Operation

The ADC generates the following DMA request:

- Result Conversion Ready (RESRDY): the request is set when a conversion result is available and cleared when the RESULT register is read. When the averaging operation is enabled, the DMA request is set when the averaging is completed and result is available.

### 33.6.12 Interrupts

The ADC has the following interrupt sources:

- Result Conversion Ready: RESRDY



- Window Monitor: WINMON
- Overrun: OVERRUN

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 ADC 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* for details. The user must read the INTFLAG register to determine which interrupt condition is present.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 33.6.13 Events

The ADC can generate the following output events:

- Result Ready (RESRDY): Generated when the conversion is complete and the result is available.
- Window Monitor (WINMON): Generated when the window monitor condition match.

Setting an Event Output bit in the Event Control Register (EVCTRL.xxEO=1) enables the corresponding output event. Clearing this bit disables the corresponding output event. Refer to the *Event System* chapter for details on configuring the event system.

The peripheral can take the following actions on an input event:

- Start conversion (START): Start a conversion.
- Conversion flush (FLUSH): Flush the conversion.

Setting an Event Input bit in the Event Control register (EVCTRL.xxEI=1) enables the corresponding action on input event. Clearing this bit disables the corresponding action on input event.

**Note:** If several events are connected to the ADC, the enabled action will be taken on any of the incoming events. The events must be correctly routed in the Event System.

#### Related Links

[EVSYS – Event System](#)

### 33.6.14 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 CTRLA.RUNSTDBY=0, the ADC is disabled during sleep, but maintains its current configuration. When CTRLA.RUNSTDBY=1, the ADC continues to operate during sleep. Note that when CTRLA.RUNSTDBY=0, 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 CTRLA.RUNSTDBY=1, any enabled ADC interrupt source can wake up the CPU. While the CPU is sleeping, ADC conversion can only be triggered by events.

### 33.6.15 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

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=1, 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 are synchronized when written:

- Software Reset bit in the Control A register (CTRLA.SWRST)
- Enable bit in the Control A register (CTRLA.ENABLE)

The following registers are synchronized when written:

- Control B (CTRLB)
- Software Trigger (SWTRIG)
- Window Monitor Control (WINCTRL)
- Input Control (INPUTCTRL)
- Window Upper/Lower Threshold (WINUT/WINLT)

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

The following registers are synchronized when read:

- Software Trigger (SWTRIG)
- Input Control (INPUTCTRL)

Required read-synchronization is denoted by the "Read-Synchronized" property in the register description.

### Related Links

[Register Synchronization](#)

### 33.7 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	
		15:8						PRESCALER[2:0]		
0x06	Reserved									
0x07	Reserved									
0x08	WINCTRL	7:0						WINMODE[2:0]		
0x09	Reserved									
0x0B	Reserved									
0x0C	SWTRIG	7:0						START	FLUSH	
0x0D	Reserved									
0x0F	Reserved									
0x10	INPUTCTRL	7:0						MUXPOS[4:0]		
		15:8						MUXNEG[4:0]		
		23:16	INPUTOFFSET[3:0]					INPUTSCAN[3:0]		
		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]							
		15:8	RESULT[15:8]							
0x1C	WINLT	7:0	WINLT[7:0]							
		15:8	WINLT[15:8]							
0x1E	Reserved									
0x1F	Reserved									
0x20	WINUT	7:0	WINUT[7:0]							
		15:8	WINUT[15:8]							
0x22	Reserved									
0x23	Reserved									
0x24	GAINCORR	7:0	GAINCORR[7:0]							
		15:8	GAINCORR[11:8]							
0x26	OFFSETCORR	7:0	OFFSETCORR[7:0]							
		15:8	OFFSETCORR[11:8]							
0x28	CALIB	7:0	LINEARITY_CAL[7:0]							

Offset	Name	Bit Pos.								
		15:8							BIAS_CAL[2:0]	
0x2A	DBGCTRL	7:0								DBGRUN

### 33.8 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.

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.

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.

### 33.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** Write-Protected

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

#### Bit 2 – RUNSTDBY Run in Standby

This bit indicates whether the ADC will continue running in standby sleep mode or not:

Value	Description
0	The ADC is halted during standby sleep mode.
1	The ADC continues normal operation during standby sleep mode.

#### Bit 1 – ENABLE Enable

Due to synchronization, there is a 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.

Value	Description
0	The ADC is disabled.
1	The ADC is enabled.

#### 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 ADC, except DBGCTRL, to their initial state, and the ADC 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.

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.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

### 33.8.2 Reference Control

**Name:** REFCTRL  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** Write-Protected

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

Value	Description
0	Reference buffer offset compensation is disabled.
1	Reference buffer offset compensation is enabled.

**Bits 3:0 – REFSEL[3:0]** Reference Selection

These bits select the reference for the ADC.

**Table 33-5. 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

### 33.8.3 Average Control

**Name:** AVGCTRL  
**Offset:** 0x02  
**Reset:** 0x00  
**Property:** Write-Protected

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

**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.

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

**33.8.4 Sampling Time Control**

**Name:** SAMPCTRL  
**Offset:** 0x03  
**Reset:** 0x00  
**Property:** Write-Protected

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

**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:

$$\text{Sampling time} = (\text{SAMPLEN} + 1) \cdot \left( \frac{\text{CLK}_{\text{ADC}}}{2} \right)$$



### 33.8.5 Control B

**Name:** CTRLB  
**Offset:** 0x04  
**Reset:** 0x0000  
**Property:** Write-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	PRESCALER[2:0]							
Access						R/W	R/W	R/W
Reset						0	0	0
Bit	7	6	5	4	3	2	1	0
	RESSEL[1:0]		CORREN		FREERUN	LEFTADJ	DIFFMODE	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

#### Bits 10:8 – PRESCALER[2:0] Prescaler Configuration

These bits define the ADC clock relative to the peripheral clock.

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 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.

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

# ATSAMHAXEXXA

## ADC – Analog-to-Digital Converter

Value	Description
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.

### Bit 2 – FREERUN Free Running Mode

Value	Description
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.

### Bit 1 – LEFTADJ Left-Adjusted Result

Value	Description
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.

### Bit 0 – DIFFMODE Differential Mode

Value	Description
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.

### 33.8.6 Window Monitor Control

**Name:** WINCTRL  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** Write-Protected, Write-Synchronized

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

**Bits 2:0 – WINMODE[2:0]** Window Monitor Mode  
 These bits enable and define the 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

### 33.8.7 Software Trigger

**Name:** SWTRIG  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** Write-Protected, Write-Synchronized

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

**Bit 1 – START** ADC Start Conversion  
 Writing this bit to zero will have no effect.

Value	Description
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.

**Bit 0 – FLUSH** ADC Conversion Flush  
 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.

Value	Description
0	No flush action.
1	"Writing a '1' to this bit will flush the ADC pipeline. 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 will be cleared after 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.

### 33.8.8 Input Control

**Name:** INPUTCTRL  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** Write-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
					GAIN[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					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/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
				MUXPOS[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

**Bits 27:24 – GAIN[3:0]** Gain Factor Selection  
 These bits set the gain factor of the ADC gain stage.

GAIN[3:0]	Name	Description
0x0	1X	1x
0x1	2X	2x
0x2	4X	4x
0x3	8X	8x
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.

# ATSAMHAXEXXA

## ADC – Analog-to-Digital Converter

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 12:8 – MUXNEG[4:0] Negative Mux Input Selection

These bits define the Mux selection for the negative ADC input. selections.

Value	Name	Description
0x00	PIN0	
0x01	PIN1	
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	ILOGND	I/O ground
0x1A-0x1F	Reserved	

### Bits 4:0 – MUXPOS[4:0] Positive Mux Input Selection

These bits define the Mux selection for the positive ADC input. The following table 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 SamplingControl register must be written.

MUXPOS[4:0]	Group configuration	Description
0x00		
0x01		
0x04	PIN4	ADC AIN4 pin
0x05		
0x06		
0x07	PIN7	ADC AIN7 pin
0x0B	PIN11	ADC AIN11 pin
0x0E		
0x0F		
0x10	PIN16	ADC AIN16 pin
0x11	PIN17	ADC AIN17 pin

# ATSAMHAXEXXA

## ADC – Analog-to-Digital Converter

MUXPOS[4:0]	Group configuration	Description
0x19	BANDGAP	Bandgap voltage
0x1A	SCALED COREVCC	1/4 scaled core supply
0x1B	SCALED IOVCC	1/4 scaled I/O supply

### 33.8.9 Event Control

**Name:** EVCTRL  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** Write-Protected

	7	6	5	4	3	2	1	0
			WINMONEO	RESRDYEO			SYNCEI	STARTEI
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

**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.

Value	Description
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.

Value	Description
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.

**Bit 1 – SYNCEI** Synchronization Event In

Value	Description
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

Value	Description
0	A new conversion will not be triggered on any incoming event.
1	A new conversion will be triggered on any incoming event.



### 33.8.10 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x16  
**Reset:** 0x00  
**Property:** Write-Protected

Bit	7	6	5	4	3	2	1	0
					SYNCRDY	WINMON	OVERRUN	RESRDY
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

**Bit 3 – SYNCRDY** Synchronization Ready Interrupt Enable

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.

Value	Description
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.

**Bit 2 – WINMON** Window Monitor Interrupt Enable

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.

Value	Description
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.

**Bit 1 – OVERRUN** Overrun Interrupt Enable

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.

Value	Description
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.

**Bit 0 – RESRDY** Result Ready Interrupt Enable

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.

# ATSAMHAXEXXA

## ADC – Analog-to-Digital Converter

Value	Description
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.

### 33.8.11 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x17  
**Reset:** 0x00  
**Property:** Write-Protected

Bit	7	6	5	4	3	2	1	0
					SYNCRDY	WINMON	OVERRUN	RESRDY
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

**Bit 3 – SYNCRDY** Synchronization Ready Interrupt Enable

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.

Value	Description
0	The Synchronization Ready interrupt is disabled.
1	The Synchronization Ready interrupt is enabled.

**Bit 2 – WINMON** Window Monitor Interrupt Enable

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.

Value	Description
0	The Window Monitor interrupt is disabled.
1	The Window Monitor interrupt is enabled.

**Bit 1 – OVERRUN** Overrun Interrupt Enable

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.

Value	Description
0	The Overrun interrupt is disabled.
1	The Overrun interrupt is enabled.

**Bit 0 – RESRDY** Result Ready Interrupt Enable

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.

Value	Description
0	The Result Ready interrupt is disabled.
1	The Result Ready interrupt is enabled.

### 33.8.12 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x18  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
					SYNCRDY	WINMON	OVERRUN	RESRDY
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

**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.SYNCBUSY), 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 INTENCLR/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.

**33.8.13 Status**

**Name:** STATUS  
**Offset:** 0x19  
**Reset:** 0x00  
**Property:** -

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

**33.8.14 Result**

**Name:** RESULT  
**Offset:** 0x1A  
**Reset:** 0x0000  
**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 conversion mode without averaging, 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).

**33.8.15 Window Monitor Lower Threshold**

**Name:** WINLT  
**Offset:** 0x1C  
**Reset:** 0x0000  
**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.

**33.8.16 Window Monitor Upper Threshold**

**Name:** WINUT  
**Offset:** 0x20  
**Reset:** 0x0000  
**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.



**33.8.17 Gain Correction**

**Name:** GAINCORR  
**Offset:** 0x24  
**Reset:** 0x0000  
**Property:** Write-Protected

	Bit	15	14	13	12	11	10	9	8
		GAINCORR[11:8]							
Access						R/W	R/W	R/W	R/W
Reset						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 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 gain-correction is a fractional value, a 1-bit integer plus an 11-bit fraction, and therefore  $1/2 \leq \text{GAINCORR} < 2$ . GAINCORR values range from 0.1000000000 to 1.1111111111.

**33.8.18 Offset Correction**

**Name:** OFFSETCORR  
**Offset:** 0x26  
**Reset:** 0x0000  
**Property:** Write-Protected

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

**33.8.19 Calibration**

**Name:** CALIB  
**Offset:** 0x28  
**Reset:** 0x0000  
**Property:** Write-Protected

Bit	15	14	13	12	11	10	9	8
							BIAS_CAL[2:0]	
Access						R/W	R/W	R/W
Reset						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 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.

**33.8.20 Debug Control**

**Name:** DBGCTRL  
**Offset:** 0x2A  
**Reset:** 0x00  
**Property:** Write-Protected

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

**Bit 0 – DBGRUN** Debug Run

This bit can be changed only while the ADC is disabled.

This bit should be written only while a conversion is not ongoing.

Value	Description
0	The ADC is halted during debug mode.
1	The ADC continues normal operation during debug mode.

## 34. AC – Analog Comparators

### 34.1 Overview

The Analog Comparator (AC) supports two individual comparators. Each comparator (COMP) compares the voltage levels on two inputs, and provides a digital output based on this comparison. Each comparator may be configured to generate interrupt requests and/or peripheral events upon several different combinations of input change.

Hysteresis can be adjusted to achieve the optimal operation for each application.

The input selection includes four shared analog port pins and several internal signals. Each comparator output state can also be output on a pin for use by external devices.

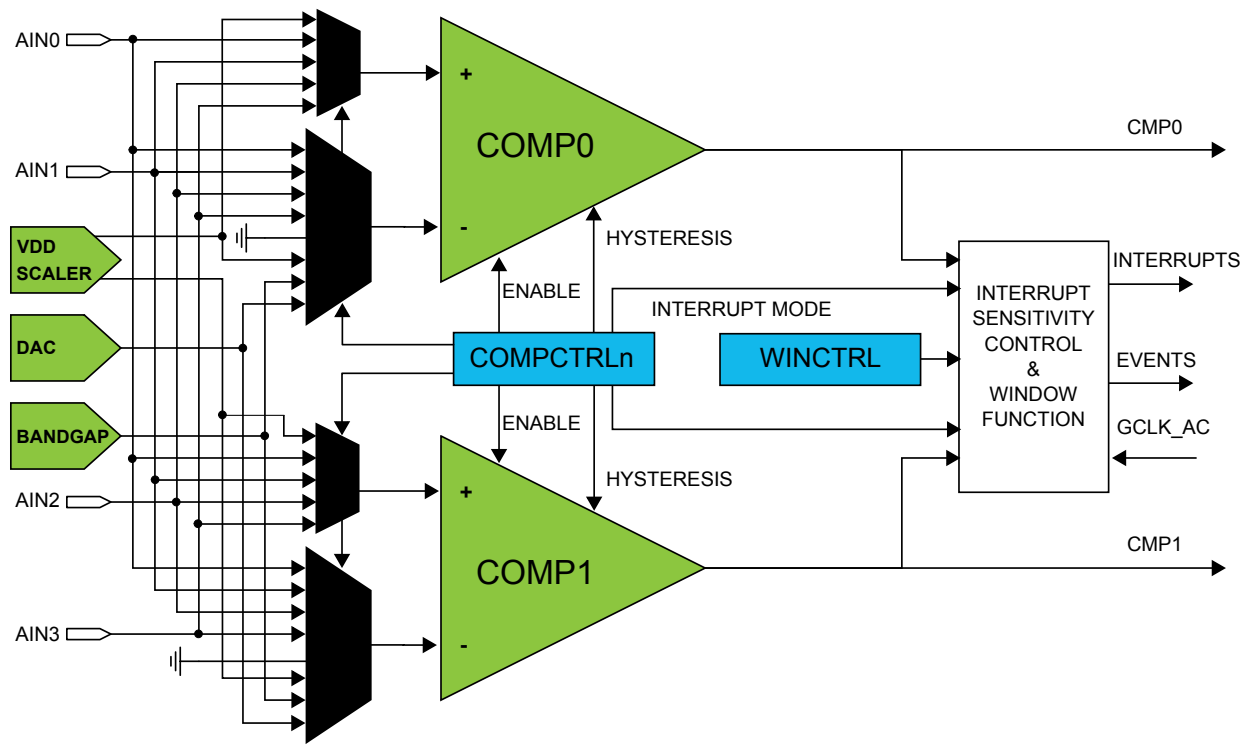
The comparators are grouped in pairs on each port. The AC peripheral implements one pair of comparators. These are called Comparator 0 (COMP0) and Comparator 1 (COMP1). They have identical behaviors, but separate control registers. The pair can be set in window mode to compare a signal to a voltage range instead of a single voltage level.

### 34.2 Features

- Two individual comparators
- Hysteresis: On or Off
- Analog comparator outputs available on pins
  - Asynchronous or synchronous
- Flexible input selection:
  - Four pins selectable for positive or negative inputs
  - Ground (for zero crossing)
  - Bandgap reference voltage
  - 64-level programmable VDD scaler per comparator
  - DAC
- Interrupt generation on:
  - Rising or falling edge
  - Toggle
  - End of comparison
- Window function interrupt generation on:
  - Signal above window
  - Signal inside window
  - Signal below window
  - Signal outside window
- Event generation on:
  - Comparator output
  - Window function inside/outside window
- Optional digital filter on comparator output

### 34.3 Block Diagram

Figure 34-1. Analog Comparator Block Diagram



### 34.4 Signal Description

Signal	Description	Type
AIN[3..0]	Analog input	Comparator inputs
CMP[1..0]	Digital output	Comparator outputs

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

#### Related Links

[I/O Multiplexing and Considerations](#)

### 34.5 Product Dependencies

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

#### 34.5.1 I/O Lines

Using the AC's I/O lines requires the I/O pins to be configured. Refer to *PORT - I/O Pin Controller* for details.

#### Related Links

[PORT - I/O Pin Controller](#)

### 34.5.2 Power Management

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

#### Related Links

[PM – Power Manager](#)

### 34.5.3 Clocks

The AC bus clock (CLK\_AC\_APB) can be enabled and disabled in the Main Clock module, MCLK (see *MCLK - Main Clock*, and the default state of CLK\_AC\_APB can be found in *Peripheral Clock Masking*.

Two generic clocks (GCLK\_AC\_DIG and GCLK\_AC\_ANA) are used by the AC. The digital clock (GCLK\_AC\_DIG) is required to provide the sampling rate for the comparators, while the analog clock (GCLK\_AC\_ANA) is required for low voltage operation ( $V_{DDANA} < 2.5V$ ) to ensure that the resistance of the analog input multiplexors remains low. These clocks must be configured and enabled in the Generic Clock Controller before using the peripheral.

This generic clock is asynchronous to the bus clock (CLK\_AC\_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) for further details.

#### Related Links

[PM – Power Manager](#)

### 34.5.4 DMA

Not applicable.

### 34.5.5 Interrupts

The interrupt request lines are connected to the interrupt controller. Using the AC interrupts requires the interrupt controller to be configured first. Refer to *Nested Vector Interrupt Controller* for details.

#### Related Links

[Nested Vector Interrupt Controller](#)

### 34.5.6 Events

The events are connected to the Event System. Refer to *EVSYS – Event System* for details on how to configure the Event System.

#### Related Links

[EVSYS – Event System](#)

### 34.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging.

### 34.5.8 Register Access Protection

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

- Control B register (CTRLB)

- Interrupt Flag register (INTFLAG)

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

PAC write-protection does not apply to accesses through an external debugger.

#### **Related Links**

[PAC - Peripheral Access Controller](#)

### **34.5.9 Analog Connections**

Each comparator has up to four I/O pins that can be used as analog inputs. Each pair of comparators shares the same four pins. These pins must be configured for analog operation before using them as comparator inputs.

Any internal reference source, such as a bandgap voltage reference, or DAC must be configured and enabled prior to its use as a comparator input.

## **34.6 Functional Description**

### **34.6.1 Principle of Operation**

Each comparator has one positive input and one negative input. Each positive input may be chosen from a selection of analog input pins. Each negative input may be chosen from a selection of both analog input pins and internal inputs, such as a bandgap voltage reference.

The digital output from the comparator is '1' when the difference between the positive and the negative input voltage is positive, and '0' otherwise.

The individual comparators can be used independently (normal mode) or paired to form a window comparison (window mode).

### **34.6.2 Basic Operation**

#### **34.6.2.1 Initialization**

Before enabling the AC, the input and output events must be configured in the Event Control register (EVCTRL). These settings cannot be changed while the AC is enabled.

#### **34.6.2.2 Enabling, Disabling and Resetting**

The AC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The AC is disabled writing a '0' to CTRLA.ENABLE.

The AC is reset by writing a '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the AC will be reset to their initial state, and the AC will be disabled. Refer to *CTRLA* for details.

The individual comparators must be also enabled by writing a '1' to the Enable bit in the Comparator x Control registers (COMPCTRLx.ENABLE). However, when the AC is disabled, this will also disable the individual comparators, but will not clear their COMPCTRLx.ENABLE bits.

#### **Related Links**

[CTRLA](#)



### 34.6.2.3 Comparator Configuration

Each individual comparator must be configured by its respective Comparator Control register (COMPCTRLx) before that comparator is enabled. These settings cannot be changed while the comparator is enabled.

- Select the desired measurement mode with COMPCTRLx.SINGLE. See [Starting a Comparison](#) for more details.
- Select the hysteresis with the COMPCTRLx.HYST bit. See [Input Hysteresis](#) for more details.
- Select the comparator speed versus power with COMPCTRLx.SPEED. See [Propagation Delay vs. Power Consumption](#) for more details.
- Select the interrupt source with COMPCTRLx.INTSEL.
- Select the positive and negative input sources with the COMPCTRLx.MUXPOS and COMPCTRLx.MUXNEG bits. See [Selecting Comparator Inputs](#) for more details.
- Select the filtering option with COMPCTRLx.FLEN.
- Select standby operation with Run in Standby bit (COMPCTRLx.RUNSTDBY).

The individual comparators are enabled by writing a '1' to the Enable bit in the Comparator x Control registers (COMPCTRLx.ENABLE). The individual comparators are disabled by writing a '0' to COMPCTRLx.ENABLE. Writing a '0' to CTRLA.ENABLE will also disable all the comparators, but will not clear their COMPCTRLx.ENABLE bits.

### 34.6.2.4 Starting a Comparison

Each comparator channel can be in one of two different measurement modes, determined by the Single bit in the Comparator x Control register (COMPCTRLx.SINGLE):

- Continuous measurement
- Single-shot

After being enabled, a start-up delay is required before the result of the comparison is ready. This start-up time is measured automatically to account for environmental changes, such as temperature or voltage supply level, and is specified in *Electrical Characteristics*. During the start-up time, the COMP output is not available.

The comparator can be configured to generate interrupts when the output toggles, when the output changes from '0' to '1' (rising edge), when the output changes from '1' to '0' (falling edge) or at the end of the comparison. An end-of-comparison interrupt can be used with the single-shot mode to chain further events in the system, regardless of the state of the comparator outputs. The interrupt mode is set by the Interrupt Selection bit group in the Comparator Control register (COMPCTRLx.INTSEL). Events are generated using the comparator output state, regardless of whether the interrupt is enabled or not.

#### Related Links

[Electrical Characteristics](#)

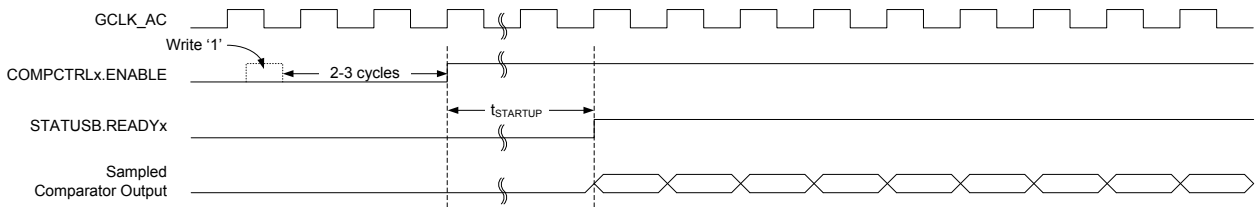
#### Continuous Measurement

Continuous measurement is selected by writing COMPCTRLx.SINGLE to zero. In continuous mode, the comparator is continuously enabled and performing comparisons. This ensures that the result of the latest comparison is always available in the Current State bit in the Status A register (STATUSA.STATEx).

After the start-up time has passed, a comparison is done and STATUSA is updated. The Comparator x Ready bit in the Status B register (STATUSB.READYx) is set, and the appropriate peripheral events and interrupts are also generated. New comparisons are performed continuously until the COMPCTRLx.ENABLE bit is written to zero. The start-up time applies only to the first comparison.

In continuous operation, edge detection of the comparator output for interrupts is done by comparing the current and previous sample. The sampling rate is the CLK\_AC\_DIG frequency. An example of continuous measurement is shown in the next figure.

**Figure 34-2. Continuous Measurement Example**



For low-power operation, comparisons can be performed during sleep modes without a clock. The comparator is enabled continuously, and changes of the comparator state are detected asynchronously. When a toggle occurs, the Power Manager will start CLK\_AC\_DIG to register the appropriate peripheral events and interrupts. The CLK\_AC\_DIG clock is then disabled again automatically, unless configured to wake up the system from sleep.

### Related Links

[Electrical Characteristics](#)

### Single-Shot

Single-shot operation is selected by writing COMPCTRLx.SINGLE to '1'. During single-shot operation, the comparator is normally idle. The user starts a single comparison by writing '1' to the respective Start Comparison bit in the write-only Control B register (CTRLB.STARTx). The comparator is enabled, and after the start-up time has passed, a single comparison is done and STATUSA is updated. Appropriate peripheral events and interrupts are also generated. No new comparisons will be performed.

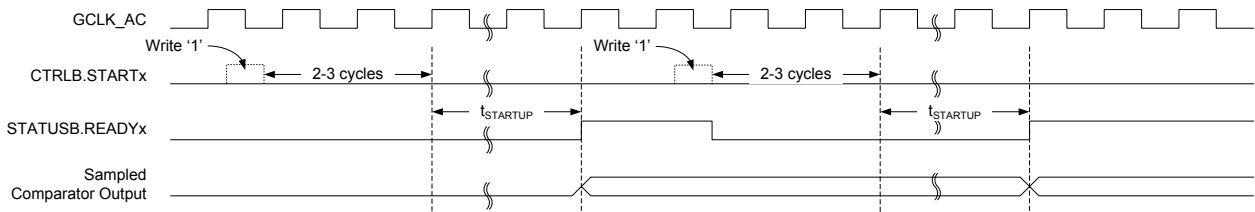
Writing '1' to CTRLB.STARTx also clears the Comparator x Ready bit in the Status B register (STATUSB.READYx). STATUSB.READYx is set automatically by hardware when the single comparison has completed.

To remove the need for polling, an additional means of starting the comparison is also available. A read of the Status C register (STATUSC) will start a comparison on all comparators currently configured for single-shot operation. The read will stall the bus until all enabled comparators are ready. If a comparator is already busy with a comparison, the read will stall until the current comparison is complete, and a new comparison will not be started.

A single-shot measurement can also be triggered by the Event System. Setting the Comparator x Event Input bit in the Event Control Register (EVCTRL.COMPEIx) enables triggering on incoming peripheral events. Each comparator can be triggered independently by separate events. Event-triggered operation is similar to user-triggered operation; the difference is that a peripheral event from another hardware module causes the hardware to automatically start the comparison and clear STATUSB.READYx.

To detect an edge of the comparator output in single-shot operation for the purpose of interrupts, the result of the current measurement is compared with the result of the previous measurement (one sampling period earlier). An example of single-shot operation is shown in the figure below.

**Figure 34-3. Single-Shot Example**



For low-power operation, event-triggered measurements can be performed during sleep modes. When the event occurs, the Power Manager will start CLK\_AC\_DIG. The comparator is enabled, and after the startup time has passed, a comparison is done and appropriate peripheral events and interrupts are also generated. The comparator and CLK\_AC\_DIG are then disabled again automatically, unless configured to wake up the system from sleep.

### Related Links

[Electrical Characteristics](#)

### 34.6.3 Selecting Comparator Inputs

Each comparator has one positive and one negative input. The positive input is one of the external input pins (AINx). The negative input can be fed either from an external input pin (AINx) or from one of the several internal reference voltage sources common to all comparators. The user selects the input source as follows:

- The positive input is selected by the Positive Input MUX Select bit group in the Comparator Control register (COMPCTRLx.MUXPOS)
- The negative input is selected by the Negative Input MUX Select bit group in the Comparator Control register (COMPCTRLx.MUXNEG)

In the case of using an external I/O pin, the selected pin must be configured for analog use in the PORT Controller by disabling the digital input and output. The switching of the analog input multiplexers is controlled to minimize crosstalk between the channels. The input selection must be changed only while the individual comparator is disabled.

**Note:** For internal use of the comparison results by the CCL, this bit must be 0x1 or 0x2.

### 34.6.4 Window Operation

Each comparator pair can be configured to work together in window mode. In this mode, a voltage range is defined, and the comparators give information about whether an input signal is within this range or not. Window mode is enabled by the Window Enable x bit in the Window Control register (WINCTRL.WENx). Both comparators in a pair must have the same measurement mode setting in their respective Comparator Control Registers (COMPCTRLx.SINGLE).

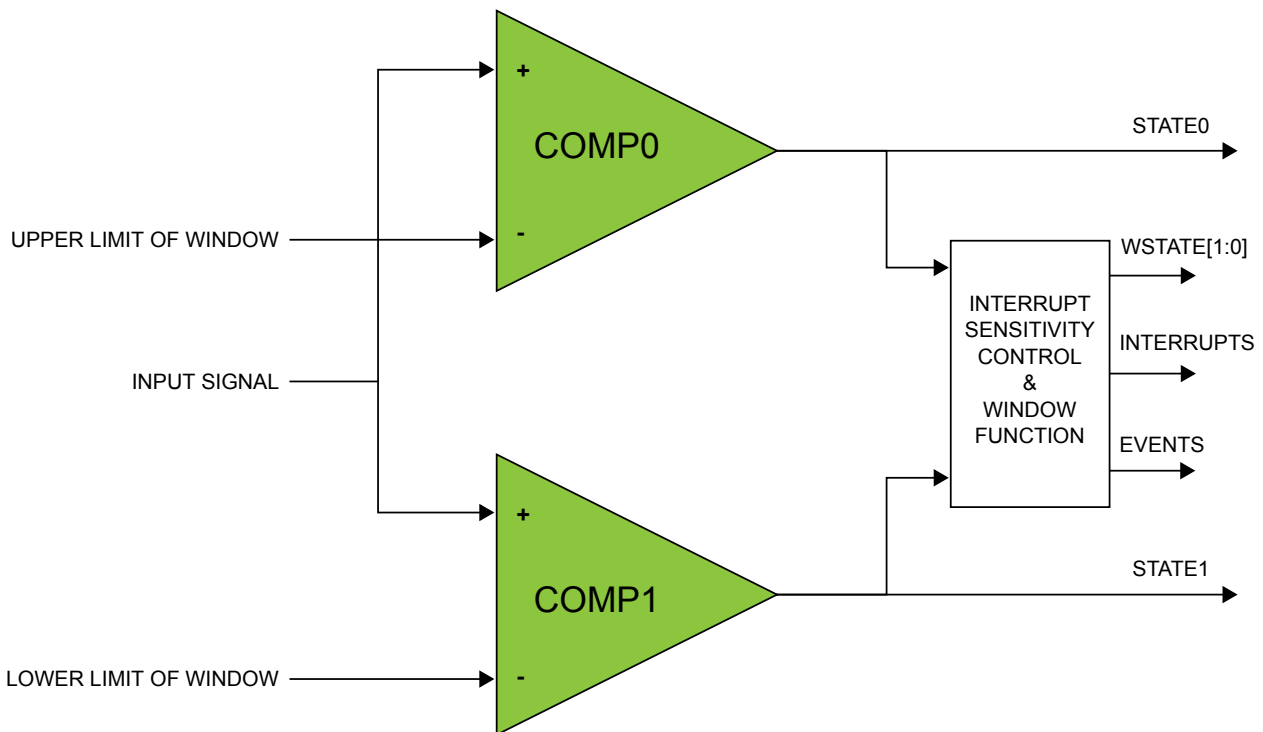
To physically configure the pair of comparators for window mode, the same I/O pin must be chosen as positive input for each comparator, providing a shared input signal. The negative inputs define the range for the window. In [Figure 34-4](#), COMP0 defines the upper limit and COMP1 defines the lower limit of the window, as shown but the window will also work in the opposite configuration with COMP0 lower and COMP1 higher. The current state of the window function is available in the Window x State bit group of the Status register (STATUS.WSTATEx).

Window mode can be configured to generate interrupts when the input voltage changes to below the window, when the input voltage changes to above the window, when the input voltage changes into the window or when the input voltage changes outside the window. The interrupt selections are set by the Window Interrupt Selection bit field in the Window Control register (WINCTRL.WINTSEL). Events are

generated using the inside/outside state of the window, regardless of whether the interrupt is enabled or not. Note that the individual comparator outputs, interrupts and events continue to function normally during window mode.

When the comparators are configured for window mode and single-shot mode, measurements are performed simultaneously on both comparators. Writing '1' to either Start Comparison bit in the Control B register (CTRLB.STARTx) will start a measurement. Likewise either peripheral event can start a measurement.

**Figure 34-4. Comparators in Window Mode**



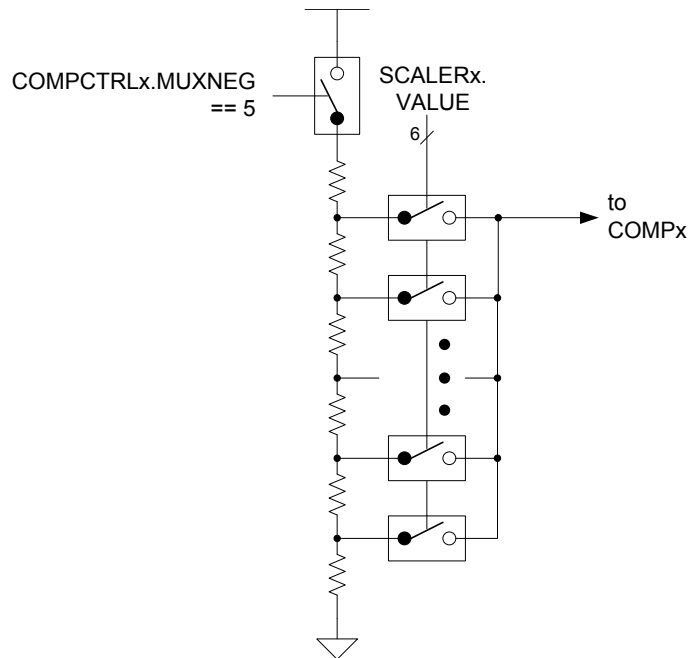
### 34.6.5 Voltage Doubler

The AC contains a voltage doubler that can reduce the resistance of the analog multiplexors when the supply voltage is below 2.5V. The voltage doubler is normally switched on/off automatically based on the supply level. When enabling the comparators, additional start-up time is required for the voltage doubler to settle. If the supply voltage is guaranteed to be above 2.5V, the voltage doubler can be disabled by writing the Low-Power Mux bit in the Control A register (CTRLA.LPMUX) to one. Disabling the voltage doubler saves power and reduces the start-up time.

### 34.6.6 $V_{DDANA}$ Scaler

The  $V_{DDANA}$  scaler generates a reference voltage that is a fraction of the device's supply voltage, with 64 levels. One independent voltage channel is dedicated for each comparator. The scaler of a comparator is enabled when the Negative Input Mux bit field in the respective Comparator Control register (COMPCTRLx.MUXNEG) is set to 0x5 and the comparator is enabled. The voltage of each channel is selected by the Value bit field in the Scaler x registers (SCALERx.VALUE).

**Figure 34-5.  $V_{DDANA}$  Scaler**



### 34.6.7 Input Hysteresis

Application software can selectively enable/disable hysteresis for the comparison. Applying hysteresis will help prevent constant toggling of the output, which can be caused by noise when the input signals are close to each other.

Hysteresis is enabled for each comparator individually by the Hysteresis Enable bit in the Comparator x Control register (`COMPCTRLx.HYSTEN`). Hysteresis is available only in continuous mode (`COMPCTRLx.SINGLE=0`).

### 34.6.8 Propagation Delay vs. Power Consumption

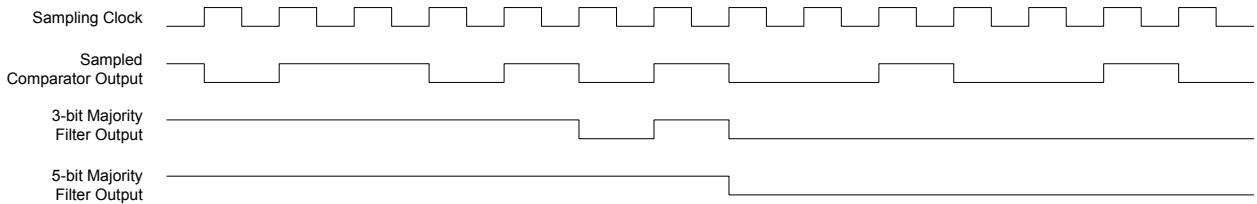
It is possible to trade off comparison speed for power efficiency to get the shortest possible propagation delay or the lowest power consumption. The speed setting is configured for each comparator individually by the Speed bit group in the Comparator x Control register (`COMPCTRLx.SPEED`). The Speed bits select the amount of bias current provided to the comparator, and as such will also affect the start-up time.

### 34.6.9 Filtering

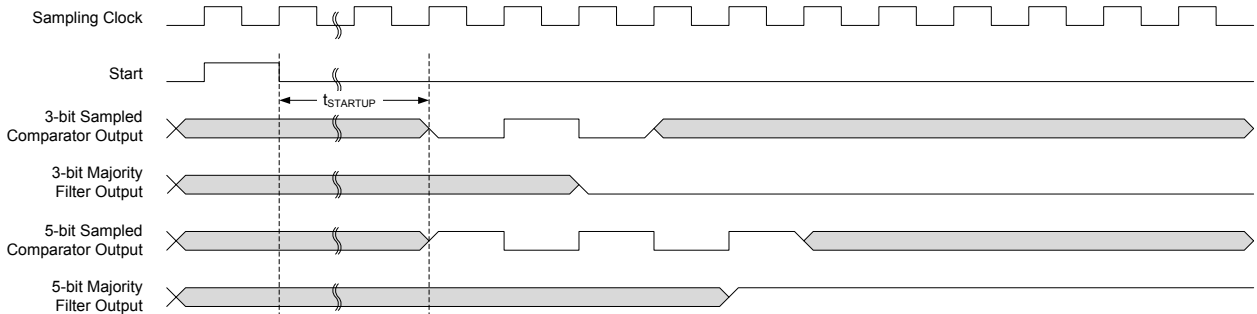
The output of the comparators can be filtered digitally to reduce noise. The filtering is determined by the Filter Length bits in the Comparator Control x register (`COMPCTRLx.FLEN`), and is independent for each comparator. Filtering is selectable from none, 3-bit majority ( $N=3$ ) or 5-bit majority ( $N=5$ ) functions. Any change in the comparator output is considered valid only if  $N/2+1$  out of the last  $N$  samples agree. The filter sampling rate is the `GCLK_AC` frequency.

Note that filtering creates an additional delay of  $N-1$  sampling cycles from when a comparison is started until the comparator output is validated. For continuous mode, the first valid output will occur when the required number of filter samples is taken. Subsequent outputs will be generated every cycle based on the current sample plus the previous  $N-1$  samples, as shown in [Figure 34-6](#). For single-shot mode, the comparison completes after the  $N$ th filter sample, as shown in [Figure 34-7](#).

**Figure 34-6. Continuous Mode Filtering**



**Figure 34-7. Single-Shot Filtering**



During sleep modes, filtering is supported only for single-shot measurements. Filtering must be disabled if continuous measurements will be done during sleep modes, or the resulting interrupt/event may be generated incorrectly.

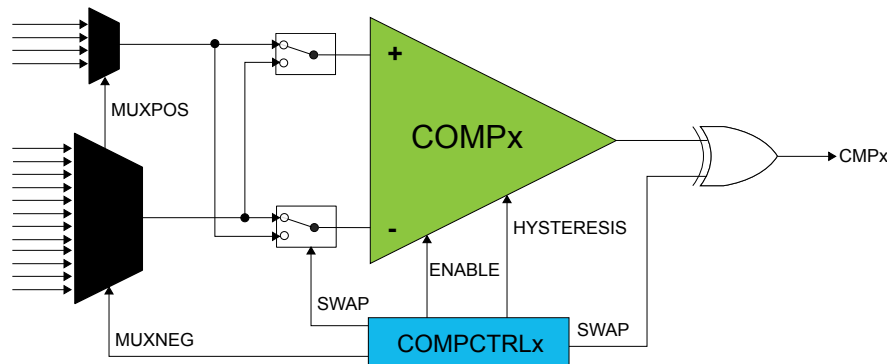
### 34.6.10 Comparator Output

The output of each comparator can be routed to an I/O pin by setting the Output bit group in the Comparator Control x register (COMPCTRLx.OUT). This allows the comparator to be used by external circuitry. Either the raw, non-synchronized output of the comparator or the CLK\_AC-synchronized version, including filtering, can be used as the I/O signal source. The output appears on the corresponding CMP[x] pin.

### 34.6.11 Offset Compensation

The Swap bit in the Comparator Control registers (COMPCTRLx.SWAP) controls switching of the input signals to a comparator's positive and negative terminals. When the comparator terminals are swapped, the output signal from the comparator is also inverted, as shown in Figure 34-8. This allows the user to measure or compensate for the comparator input offset voltage. As part of the input selection, COMPCTRLx.SWAP can be changed only while the comparator is disabled.

**Figure 34-8. Input Swapping for Offset Compensation**



### 34.6.12 Interrupts

The AC has the following interrupt sources:

- Comparator (COMP0, COMP1): Indicates a change in comparator status.
- Window (WIN0): Indicates a change in the window status.

Comparator interrupts are generated based on the conditions selected by the Interrupt Selection bit group in the Comparator Control registers (COMPCTRLx.INTSEL). Window interrupts are generated based on the conditions selected by the Window Interrupt Selection bit group in the Window Control register (WINCTRL.WINTSEL[1:0]).

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 AC is reset. See INFLAG 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. 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.

### Related Links

[Nested Vector Interrupt Controller](#)

### 34.6.13 Events

The AC can generate the following output events:

- Comparator (COMP0, COMP1): Generated as a copy of the comparator status
- Window (WIN0): Generated as a copy of the window inside/outside status

Output events must be enabled to be generated. Writing a one to an Event Output bit in the Event Control register (EVCTRL.COMPEOx) 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.

The AC can take the following action on an input event:

- Single-shot measurement
- Single-shot measurement in window mode

Writing a one to an Event Input bit into the Event Control register (EVCTRL.COMPEIx) enables the corresponding action on input event. Writing a zero to this bit disables the corresponding action on input event. Note that if several events are connected to the AC, the enabled action will be taken on any of the incoming events. Refer to the Event System chapter for details on configuring the event system.

When EVCTRL.COMPEIx is one, the event will start a comparison on COMPx after the start-up time delay. In normal mode, each comparator responds to its corresponding input event independently. For a pair of comparators in window mode, either comparator event will trigger a comparison on both comparators simultaneously.

### 34.6.14 Sleep Mode Operation

The Run in Standby bits in the Comparator x Control registers (COMPCTRLx.RUNSTDBY) control the behavior of the AC during standby sleep mode. Each RUNSTDBY bit controls one comparator. When the bit is zero, the comparator is disabled during sleep, but maintains its current configuration. When the bit is one, the comparator 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 AC interrupt source can wake up the CPU. While the CPU is sleeping, single-shot comparisons are only triggerable by events. The AC can also be used during sleep modes where the clock used by the AC is disabled, provided that the AC is still powered (not in shutdown). In this case, the behavior is slightly different and depends on the measurement mode.

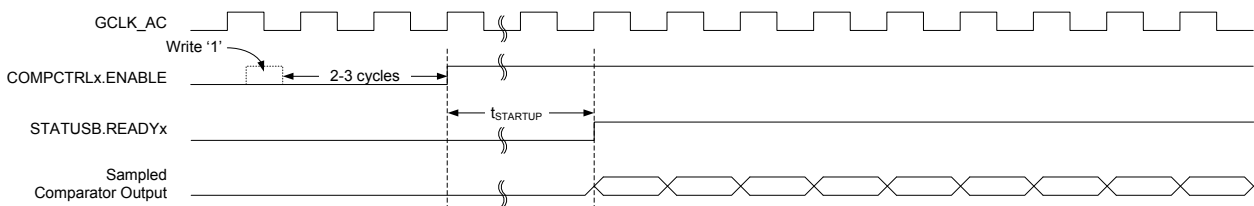
**Table 34-1. Sleep Mode Operation**

COMPCTRLx.MODE	RUNSTDBY=0	RUNSTDBY=1
0 (Continuous)	COMPx disabled	GCLK_AC_DIG stopped, COMPx enabled
1 (Single-shot)	COMPx disabled	GCLK_AC_DIG stopped, COMPx enabled only when triggered by an input event

### 34.6.14.1 Continuous Measurement during Sleep

When a comparator is enabled in continuous measurement mode and GCLK\_AC\_DIG is disabled during sleep, the comparator will remain continuously enabled and will function asynchronously. The current state of the comparator is asynchronously monitored for changes. If an edge matching the interrupt condition is found, GCLK\_AC\_DIG is started to register the interrupt condition and generate events. If the interrupt is enabled in the Interrupt Enable registers (INTENCLR/SET), the AC can wake up the device; otherwise GCLK\_AC\_DIG is disabled until the next edge detection. Filtering is not possible with this configuration.

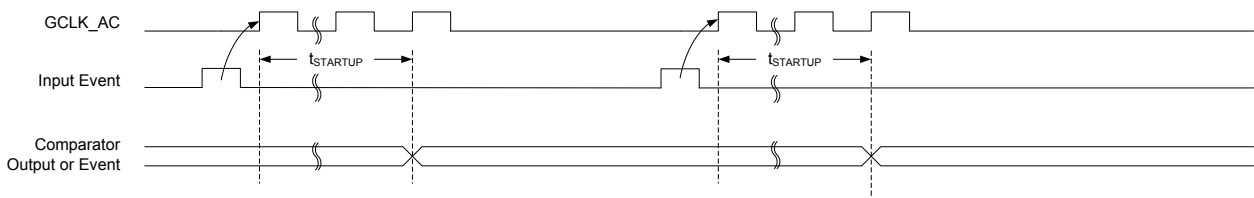
**Figure 34-9. Continuous Mode SleepWalking**



### 34.6.14.2 Single-Shot Measurement during Sleep

For low-power operation, event-triggered measurements can be performed during sleep modes. When the event occurs, the Power Manager will start GCLK\_AC\_DIG. The comparator is enabled, and after the start-up time has passed, a comparison is done, with filtering if desired, and the appropriate peripheral events and interrupts are also generated, as the figure below. The comparator and GCLK\_AC\_DIG are then disabled again automatically, unless configured to wake the system from sleep. Filtering is allowed with this configuration.

**Figure 34-10. Single-Shot SleepWalking**



### 34.6.15 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.



The following bits are synchronized when written:

- Software Reset bit in control register (CTRLA.SWRST)
- Enable bit in control register (CTRLA.ENABLE)
- Enable bit in Comparator Control register (COMPCTRLn.ENABLE)

The following registers are synchronized when written:

- Window Control register (WINCTRL)

Required write-synchronization is denoted by the "Write-Synchronized" property in the register description.

### Related Links

[Register Synchronization](#)

### 34.7 Register Summary

Offset	Name	Bit Pos.									
0x00	<a href="#">CTRLA</a>	7:0	LPMUX					RUNSTDBY	ENABLE	SWRST	
0x01	<a href="#">CTRLB</a>	7:0							STARTx	STARTx	
0x02	<a href="#">EVCTRL</a>	7:0				WINEO0			COMPEOx	COMPEOx	
		15:8							COMPEIx	COMPEIx	
0x04	<a href="#">INTENCLR</a>	7:0				WIN0			COMPx	COMPx	
0x05	<a href="#">INTENSET</a>	7:0				WIN0			COMPx	COMPx	
0x06	<a href="#">INTFLAG</a>	7:0				WIN0			COMPx	COMPx	
0x07	Reserved										
0x08	<a href="#">STATUSA</a>	7:0				WSTATE0[1:0]			STATEx	STATEx	
0x09	<a href="#">STATUSB</a>	7:0	SYNCBUSY						READYx	READYx	
0x0A	<a href="#">STATUSC</a>	7:0				WSTATE0[1:0]			STATEx	STATEx	
0x0B	Reserved										
0x0C	<a href="#">WINCTRL</a>	7:0						WINTSEL0[1:0]		WEN0	
0x0D	Reserved										
...											
0x0F											
0x10	<a href="#">COMPCTRL0</a>	7:0			INTSEL[1:0]			SPEED[1:0]	SINGLE	ENABLE	
		15:8	SWAP			MUXPOS[1:0]			MUXNEG[2:0]		
		23:16						HYST		OUT[1:0]	
		31:24								FLEN[2:0]	
0x14	<a href="#">COMPCTRL1</a>	7:0			INTSEL[1:0]			SPEED[1:0]	SINGLE	ENABLE	
		15:8	SWAP			MUXPOS[1:0]			MUXNEG[2:0]		
		23:16						HYST		OUT[1:0]	
		31:24								FLEN[2:0]	
0x18	Reserved										
...											
0x1F											
0x20	<a href="#">SCALER0</a>	7:0						VALUE[5:0]			
0x21	<a href="#">SCALER1</a>	7:0						VALUE[5:0]			

### 34.8 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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to *Register Access Protection*.

Some registers are synchronized when read and/or written. Synchronization is denoted by the "Write-Synchronized" or the "Read-Synchronized" property in each individual register description. For details, refer to *Synchronization*.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 34.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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

**Bit 7 – LPMUX** Low-Power Mux  
This bit is not synchronized

Value	Description
0	The analog input muxes have low resistance, but consume more power at lower voltages (e.g., are driven by the voltage doubler).
1	The analog input muxes have high resistance, but consume less power at lower voltages (e.g., the voltage doubler is disabled).

**Bit 2 – RUNSTDBY** Run in Standby  
This bit controls the behavior of the comparators during standby sleep mode.  
This bit is not synchronized

Value	Description
0	The comparator pair is disabled during sleep.
1	The comparator pair continues to operate during sleep.

**Bit 1 – ENABLE** Enable  
Due to synchronization, there is a delay from the time when the register is updated until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately after being written. STATUS.SYNCBUSY is set. STATUS.SYNCBUSY is cleared when the peripheral is enabled/disabled

Value	Description
0	The AC is disabled.
1	The AC is enabled. Each comparator must also be enabled individually by the Enable bit in the Comparator Control register (COMPCTRLn.ENABLE).

**Bit 0 – SWRST** Software Reset  
Writing a '0' to this bit has no effect.  
Writing a '1' to this bit resets all registers in the AC to their initial state, and the AC will be disabled.  
Writing a '1' to CTRLA.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.

# ATSAMHAXEXXA

## AC – Analog Comparators

---

---

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

**34.8.2 Control B**

**Name:** CTRLB  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** –

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

**Bits 1,0 – STARTx** Comparator x Start Comparison

Writing a '0' to this field has no effect.

Writing a '1' to STARTx starts a single-shot comparison on COMPx if both the Single-Shot and Enable bits in the Comparator x Control Register are '1' (COMPCTRLx.SINGLE and COMPCTRLx.ENABLE). If comparator x is not implemented, or if it is not enabled in single-shot mode, Writing a '1' has no effect.

This bit always reads as zero.

### 34.8.3 Event Control

**Name:** EVCTRL  
**Offset:** 0x02  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
							COMPEIx	COMPEIx
Access							R/W	R/W
Reset							0	0
Bit	7	6	5	4	3	2	1	0
				WINEO0			COMPEOx	COMPEOx
Access				R/W			R/W	R/W
Reset				0			0	0

#### Bits 9,8 – COMPEIx Comparator x Event Input

Note that several actions can be enabled for incoming events. If several events are connected to the peripheral, the enabled action will be taken for any of the incoming events. There is no way to tell which of the incoming events caused the action.

These bits indicate whether a comparison will start or not on any incoming event.

Value	Description
0	Comparison will not start on any incoming event.
1	Comparison will start on any incoming event.

#### Bit 4 – WINEO0 Window 0 Event Output Enable

These bits indicate whether the window 0 function can generate a peripheral event or not.

Value	Description
0	Window 0 Event is disabled.
1	Window 0 Event is enabled.

#### Bits 1,0 – COMPEOx Comparator x Event Output Enable

These bits indicate whether the comparator x output can generate a peripheral event or not.

Value	Description
0	COMPx event generation is disabled.
1	COMPx event generation is enabled.

### 34.8.4 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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	6	5	4	3	2	1	0
					WIN0			COMPx	COMPx
Access					R/W			R/W	R/W
Reset					0			0	0

**Bit 4 – WIN0** Window 0 Interrupt Enable

Reading this bit returns the state of the Window 0 interrupt enable.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit disables the Window 0 interrupt.

Value	Description
0	The Window 0 interrupt is disabled.
1	The Window 0 interrupt is enabled.

**Bits 1,0 – COMPx** Comparator x Interrupt Enable

Reading this bit returns the state of the Comparator x interrupt enable.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit disables the Comparator x interrupt.

Value	Description
0	The Comparator x interrupt is disabled.
1	The Comparator x interrupt is enabled.



### 34.8.5 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x05  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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).

	7	6	5	4	3	2	1	0
Bit			WIN0				COMPx	COMPx
Access			R/W				R/W	R/W
Reset			0				0	0

**Bit 4 – WIN0** Window 0 Interrupt Enable

Reading this bit returns the state of the Window 0 interrupt enable.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit enables the Window 0 interrupt.

Value	Description
0	The Window 0 interrupt is disabled.
1	The Window 0 interrupt is enabled.

**Bits 1,0 – COMPx** Comparator x Interrupt Enable

Reading this bit returns the state of the Comparator x interrupt enable.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Ready interrupt bit and enable the Ready interrupt.

Value	Description
0	The Comparator x interrupt is disabled.
1	The Comparator x interrupt is enabled.

### 34.8.6 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** –

	Bit	7	6	5	4	3	2	1	0
					WIN0			COMPx	COMPx
Access					R/W			R/W	R/W
Reset					0			0	0

**Bit 4 – WIN0** Window 0

This flag is set according to the Window 0 Interrupt Selection bit group in the WINCTRL register (WINCTRL.WINTSELx) and will generate an interrupt if INTENCLR/SET.WINx is also one.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Window 0 interrupt flag.

**Bits 1,0 – COMPx** Comparator x

Reading this bit returns the status of the Comparator x interrupt flag. If comparator x is not implemented, COMPx always reads as zero.

This flag is set according to the Interrupt Selection bit group in the Comparator x Control register (COMPCTRLx.INTSEL) and will generate an interrupt if INTENCLR/SET.COMPx is also one.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Comparator x interrupt flag.

**34.8.7 Status A**

**Name:** STATUSA  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** –

Bit	7	6	5	4	3	2	1	0
			WSTATE0[1:0]				STATE <sub>x</sub>	STATE <sub>x</sub>
Access			R	R			R	R
Reset			0	0			0	0

**Bits 5:4 – WSTATE0[1:0] Window 0 Current State**

These bits show the current state of the signal if the window 0 mode is enabled.

Value	Name	Description
0x0	ABOVE	Signal is above window
0x1	INSIDE	Signal is inside window
0x2	BELOW	Signal is below window
0x3		Reserved

**Bits 1,0 – STATE<sub>x</sub> Comparator x Current State**

This bit shows the current state of the output signal from COMP<sub>x</sub>. STATE<sub>x</sub> is valid only when STATUSB.READY<sub>x</sub> is one.

**34.8.8 Status B**

**Name:** STATUSB  
**Offset:** 0x09  
**Reset:** 0x00  
**Property:** –

	Bit	7	6	5	4	3	2	1	0
		SYNCBUSY						READY <sub>x</sub>	READY <sub>x</sub>
Access		R						R	R
Reset		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 1,0 – READY<sub>x</sub>** Comparator x Ready

This bit is cleared when the comparator x output is not ready.

This bit is set when the comparator x output is ready.

### 34.8.9 Status C

**Name:** STATUSC  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** –

STATUSC is a copy of STATUSA (see STATUSA register), with the additional feature of automatically starting single-shot comparisons. A read of STATUSC will start a comparison on all comparators currently configured for single-shot operation. The read will stall the bus until all enabled comparators are ready. If a comparator is already busy with a comparison, the read will stall until the current comparison is complete, and a new comparison will not be started.

Bit	7	6	5	4	3	2	1	0
			WSTATE0[1:0]				STATE <sub>x</sub>	STATE <sub>x</sub>
Access			R	R			R	R
Reset			0	0			0	0

#### Bits 5:4 – WSTATE0[1:0] Window 0 Current State

These bits show the current state of the signal if the window 0 mode is enabled.

Value	Name	Description
0x0	ABOVE	Signal is above window
0x1	INSIDE	Signal is inside window
0x2	BELOW	Signal is below window
0x3		Reserved

#### Bits 1,0 – STATE<sub>x</sub> Comparator x Current State

This bit shows the current state of the output signal from COMP<sub>x</sub>. STATE<sub>x</sub> is valid only when STATUSB.READY<sub>x</sub> is one.

### 34.8.10 Window Control

**Name:** WINCTRL  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

	7		6		5		4		3		2		1		0
											WINTSEL0[1:0]		WEN0		
Access											R/W	R/W	R/W		
Reset											0	0	0		

**Bits 2:1 – WINTSEL0[1:0]** Window 0 Interrupt Selection

These bits configure the interrupt mode for the comparator window 0 mode.

Value	Name	Description
0x0	ABOVE	Interrupt on signal above window
0x1	INSIDE	Interrupt on signal inside window
0x2	BELOW	Interrupt on signal below window
0x3	OUTSIDE	Interrupt on signal outside window

**Bit 0 – WEN0** Window 0 Mode Enable

Value	Description
0	Window mode is disabled for comparators 0 and 1.
1	Window mode is enabled for comparators 0 and 1.

### 34.8.11 Comparator Control n

**Name:** COMPCTRL  
**Offset:** 0x10 + n\*0x04 [n=0..1]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
							FLEN[2:0]	
Access						R/W	R/W	R/W
Reset						0	0	0
Bit	23	22	21	20	19	18	17	16
					HYST			OUT[1:0]
Access					R/W			R/W R/W
Reset					0			0 0
Bit	15	14	13	12	11	10	9	8
	SWAP			MUXPOS[1:0]		MUXNEG[2:0]		
Access	R/W			R/W	R/W			R/W
Reset	0			0	0			0
Bit	7	6	5	4	3	2	1	0
			INTSEL[1:0]				SPEED[1:0]	SINGLE ENABLE
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### Bits 26:24 – FLEN[2:0] Filter Length

These bits configure the filtering for comparator n. COMPCTRLn.FLEN can only be written while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Name	Description
0x0	OFF	No filtering
0x1	MAJ3	3-bit majority function (2 of 3)
0x2	MAJ5	5-bit majority function (3 of 5)
0x3-0x7	N/A	Reserved

#### Bit 19 – HYST Hysteresis Enable

This bit indicates the hysteresis mode of comparator n. Hysteresis is available only for continuous mode (COMPCTRLn.SINGLE=0). COMPCTRLn.HYST can be written only while COMPCTRLn.ENABLE is zero.

This bit is not synchronized.

These bits are not synchronized.

Value	Name
0	Hysteresis is disabled.
1	Hysteresis is enabled.

### Bits 17:16 – OUT[1:0] Output

These bits configure the output selection for comparator n. COMPCTRLn.OUT can be written only while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Name	Description
0x0	OFF	The output of COMPn is not routed to the COMPn I/O port
0x1	ASYN	The asynchronous output of COMPn is routed to the COMPn I/O port
0x2	SYNC	The synchronous output (including filtering) of COMPn is routed to the COMPn I/O port
0x3	N/A	Reserved

### Bit 15 – SWAP Swap Inputs and Invert

This bit swaps the positive and negative inputs to COMPn and inverts the output. This function can be used for offset cancellation. COMPCTRLn.SWAP can be written only while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Description
0	The output of MUXPOS connects to the positive input, and the output of MUXNEG connects to the negative input.
1	The output of MUXNEG connects to the positive input, and the output of MUXPOS connects to the negative input.

### Bits 13:12 – MUXPOS[1:0] Positive Input Mux Selection

These bits select which input will be connected to the positive input of comparator n. COMPCTRLn.MUXPOS can be written only while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Name	Description
0x0	PIN0	I/O pin 0
0x1	PIN1	I/O pin 1
0x2	PIN2	I/O pin 2
0x3	PIN3	I/O pin 3

### Bits 10:8 – MUXNEG[2:0] Negative Input Mux Selection

These bits select which input will be connected to the negative input of comparator n. COMPCTRLn.MUXNEG can only be written while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Name	Description
0x0	PIN0	I/O pin 0
0x1	PIN1	I/O pin 1
0x2	PIN2	I/O pin 2
0x3	PIN3	I/O pin 3



Value	Name	Description
0x4	GND	Ground
0x5	VSCALE	VDD scaler
0x6	BANDGAP	Internal bandgap voltage
0x7	DAC	DAC output

### Bits 6:5 – INTSEL[1:0] Interrupt Selection

These bits select the condition for comparator n to generate an interrupt or event. COMPCTRLn.INTSEL can be written only while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Name	Description
0x0	TOGGLE	Interrupt on comparator output toggle
0x1	RISING	Interrupt on comparator output rising
0x2	FALLING	Interrupt on comparator output falling
0x3	EOC	Interrupt on end of comparison (single-shot mode only)

### Bits 3:2 – SPEED[1:0] Speed Selection

This bit indicates the speed/propagation delay mode of comparator n. COMPCTRLn.SPEED can be written only while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Name	Description
0x0	LOW	Low speed
0x1	HIGH	High speed
0x2-0x3	N/A	Reserved

### Bit 1 – SINGLE Single-Shot Mode

This bit determines the operation of comparator n. COMPCTRLn.SINGLE can be written only while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Description
0	Comparator n operates in continuous measurement mode.
1	Comparator n operates in single-shot mode.

### Bit 0 – ENABLE Enable

Writing a zero to this bit disables comparator n.

Writing a one to this bit enables comparator n. After writing to this bit, the value read back will not change until the action initiated by the writing is complete.

Due to synchronization, there is a latency of at least two GCLK\_AC\_DIG clock cycles from updating the register until the comparator is enabled/disabled. The bit will continue to read the previous state while the change is in progress. Writing a one to COMPCTRLn.ENABLE will prevent further changes to the other bits in COMPCTRLn. These bits remain protected until COMPCTRLn.ENABLE is written to zero and the write is synchronized.

**34.8.12 Scaler n**

**Name:** SCALER  
**Offset:** 0x20 + n\*0x01 [n=0..1]  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

**Bits 5:0 – VALUE[5:0] Scaler Value**

These bits define the scaling factor for channel n of the  $V_{DD}$  voltage scaler. The output voltage,  $V_{SCALE}$ , is:

$$V_{SCALE} = \frac{V_{DD} \cdot (VALUE+1)}{64}$$

## 35. DAC – Digital-to-Analog Converter

### 35.1 Overview

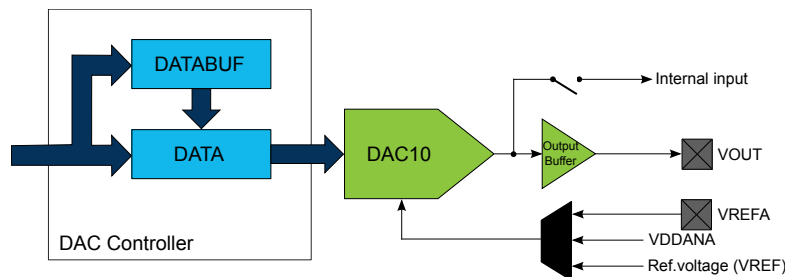
The Digital-to-Analog Converter (DAC) converts a digital value to a voltage. The DAC has one channel with 10-bit resolution, and it is capable of converting up to 350,000 samples per second (350ksps).

### 35.2 Features

- DAC with 10-bit resolution
- Up to 350ksps conversion rate
- Multiple trigger sources
- High-drive capabilities
- Output can be used as input to the Analog Comparator (AC)
- DMA support

### 35.3 Block Diagram

Figure 35-1. DAC Block Diagram



### 35.4 Signal Description

Signal Name	Type	Description
VOUT	Analog output	DAC output
VREFA	Analog input	External reference

#### Related Links

[I/O Multiplexing and Considerations](#)

### 35.5 Product Dependencies

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

#### 35.5.1 I/O Lines

Using the DAC Controller's I/O lines requires the I/O pins to be configured using the port configuration (PORT).

**Related Links**

[PORT - I/O Pin Controller](#)

**35.5.2 Power Management**

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

The DAC 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.

**Related Links**

[PM – Power Manager](#)

**35.5.3 Clocks**

The DAC bus clock (CLK\_DAC\_APB) can be enabled and disabled by the Power Manager, and the default state of CLK\_DAC\_APB can be found in the *Peripheral Clock Masking* section.

A generic clock (GCLK\_DAC) is required to clock the DAC Controller. This clock must be configured and enabled in the Generic Clock Controller before using the DAC Controller. Refer to *GCLK – Generic Clock Controller* for details.

This generic clock is asynchronous to the bus clock (CLK\_DAC\_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) for further details.

**Related Links**

[Peripheral Clock Masking](#)

[GCLK - Generic Clock Controller](#)

**35.5.4 DMA**

The DMA request line is connected to the DMA Controller (DMAC). Using the DAC Controller DMA requests requires to configure the DMAC first.

**Related Links**

[DMAC – Direct Memory Access Controller](#)

**35.5.5 Interrupts**

The interrupt request line is connected to the interrupt controller. Using the DAC Controller interrupt(s) requires the interrupt controller to be configured first.

**Related Links**

[Nested Vector Interrupt Controller](#)

**35.5.6 Events**

The events are connected to the Event System.

**Related Links**

[EVSYS – Event System](#)

**35.5.7 Debug Operation**

When the CPU is halted in debug mode the DAC will halt normal operation. Any on-going conversions will be completed. The DAC can be forced to continue normal operation during debugging. If the DAC 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.

### **35.5.8 Register Access Protection**

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

- Interrupt Flag Status and Clear (INTFLAG) register
- Data Buffer (DATABUF) register

Optional write-protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in each individual register description.

PAC write-protection does not apply to accesses through an external debugger

#### **Related Links**

[PAC - Peripheral Access Controller](#)

### **35.5.9 Analog Connections**

The DAC has one output pin (VOUT) and one analog input pin (VREFA) that must be configured first.

When internal input is used, it must be enabled before DAC Controller is enabled.

## **35.6 Functional Description**

### **35.6.1 Principle of Operation**

The DAC converts the digital value located in the Data register (DATA) into an analog voltage on the DAC output (VOUT).

A conversion is started when new data is written to the Data register. The resulting voltage is available on the DAC output after the conversion time. A conversion can also be started by input events from the Event System.

### **35.6.2 Basic Operation**

#### **35.6.2.1 Initialization**

The following registers are enable-protected, meaning they can only be written when the DAC is disabled (CTRLA.ENABLE is zero):

- Control B register (CTRLB)
- Event Control register (EVCTRL)

Enable-protection is denoted by the Enable-Protected property in the register description.

Before enabling the DAC, it must be configured by selecting the voltage reference using the Reference Selection bits in the Control B register (CTRLB.REFSEL).

#### **35.6.2.2 Enabling, Disabling and Resetting**

The DAC Controller is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The DAC Controller is disabled by writing a '0' to CTRLA.ENABLE.

The DAC Controller is reset by writing a '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the DAC will be reset to their initial state, and the DAC Controller will be disabled. Refer to the CTRLA register for details.

### 35.6.2.3 Enabling the Output Buffer

To enable the DAC output on the  $V_{OUT}$  pin, the output driver must be enabled by writing a one to the External Output Enable bit in the Control B register (CTRLB.EOEN).

The DAC output buffer provides a high-drive-strength output, and is capable of driving both resistive and capacitive loads. To minimize power consumption, the output buffer should be enabled only when external output is needed.

### 35.6.2.4 Digital to Analog Conversion

The DAC converts a digital value (stored in the DATA register) into an analog voltage. The conversion range is between GND and the selected DAC voltage reference. The default voltage reference is the internal reference voltage. Other voltage reference options are the analog supply voltage (VDDANA) and the external voltage reference (VREFA). The voltage reference is selected by writing to the Reference Selection bits in the Control B register (CTRLB.REFSEL).

The output voltage from the DAC can be calculated using the following formula:

$$V_{OUT} = \frac{DATA}{0x3FF} \cdot VREF$$

A new conversion starts as soon as a new value is loaded into DATA. DATA can either be loaded via the APB bus during a CPU write operation, using DMA, or from the DATABUF register when a START event occurs. Refer to [Events](#) for details. As there is no automatic indication that a conversion is done, the sampling period must be greater than or equal to the specified conversion time.

### 35.6.3 DMA Operation

The DAC generates the following DMA request:

- Data Buffer Empty (EMPTY): The request is set when data is transferred from DATABUF to the internal data buffer of DAC. The request is cleared when DATABUF register is written, or by writing a one to the EMPTY bit in the Interrupt Flag register (INTFLAG.EMPTY).

For each Start Conversion event, DATABUF is transferred into DATA and the conversion starts. When DATABUF is empty, the DAC generates the DMA request for new data. As DATABUF is initially empty, a DMA request is generated whenever the DAC is enabled.

If the CPU accesses the registers that are the source of a DMA request set/clear condition, the DMA request can be lost or the DMA transfer can be corrupted, if enabled.

When DAC registers are write-protected by Peripheral Access Controller, DATABUF cannot be written. To bypass DATABUF write protection, Bypass DATABUF Write Protection bit (CTRLB.BDWP) must be written to '1'

### 35.6.4 Interrupts

The DAC Controller has the following interrupt sources:

- Data Buffer Empty (EMPTY): Indicates that the internal data buffer of the DAC is empty.
- Underrun (UNDERRUN): Indicates that the internal data buffer of the DAC is empty and a DAC start of conversion event occurred. Refer to [Events](#) for details.
- Synchronization Ready (SYNCRDY): this asynchronous interrupt 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).

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 DAC is reset. See 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. 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..

### Related Links

[Nested Vector Interrupt Controller](#)

### 35.6.5 Events

The DAC Controller can generate the following output events:

- Data Buffer Empty (EMPTY): Generated when the internal data buffer of the DAC is empty. Refer to DMA Operation for details.

Writing a '1' to an Event Output bit in the Event Control register (EVCTRL.EMPTIEO) enables the corresponding output event. Writing a '0' to this bit disables the corresponding output event.

The DAC can take the following action on an input event:

- Start Conversion (START): DATABUF value is transferred into DATA as soon as the DAC is ready for the next conversion, and then conversion is started. START is considered as asynchronous to GCLK\_DAC thus it is resynchronized in DAC Controller. Refer to [Digital to Analog Conversion](#) for details.

Writing a '1' to an Event Input bit in the Event Control register (EVCTRL.STARTEI) enables the corresponding action on an input event. Writing a '0' to this bit disables the corresponding action on input event.

**Note:** When several events are connected to the DAC Controller, the enabled action will be taken on any of the incoming events.

By default, DAC Controller detects rising edge events. Falling edge detection can be enabled by writing a '1' to EVCTRL.INVEX.

### Related Links

[EVSYS – Event System](#)

### 35.6.6 Sleep Mode Operation

The generic clock for the DAC is running in idle sleep mode. If the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY) is one, the DAC output buffer will keep its value in standby sleep mode. If CTRLA.RUNSTDBY is zero, the DAC output buffer will be disabled in standby sleep mode.

### 35.6.7 Synchronization

Due to the asynchronicity between main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. 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 its busy bit is one, the operation is discarded and an error is generated.

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)
- All bits in the Data register (DATA)
- All bits in the Data Buffer register (DATABUF)

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

The following bits need synchronization when read:

- All bits in the Data register (DATA)

### 35.6.8 Additional Features

#### 35.6.8.1 DAC as an Internal Reference

The DAC output can be internally enabled as input to the analog comparator. This is enabled by writing a one to the Internal Output Enable bit in the Control B register (CTRLB.IOEN). It is possible to have the internal and external output enabled simultaneously.

The DAC output can also be enabled as input to the Analog-to-Digital Converter. In this case, the output buffer must be enabled.

#### 35.6.8.2 Data Buffer

The Data Buffer register (DATABUF) and the Data register (DATA) are linked together to form a two-stage FIFO. The DAC uses the Start Conversion event to load data from DATABUF into DATA and start a new conversion. The Start Conversion event is enabled by writing a one to the Start Event Input bit in the Event Control register (EVCTRL.STARTEI). If a Start Conversion event occurs when DATABUF is empty, an Underrun interrupt request is generated if the Underrun interrupt is enabled.

The DAC can generate a Data Buffer Empty event when DATABUF becomes empty and new data can be loaded to the buffer. The Data Buffer Empty event is enabled by writing a one to the Empty Event Output bit in the Event Control register (EVCTRL.EMPTYEO). A Data Buffer Empty interrupt request is generated if the Data Buffer Empty interrupt is enabled.

#### 35.6.8.3 Voltage Pump

When the DAC is used at operating voltages lower than 2.5V, the voltage pump must be enabled. This enabling is done automatically, depending on operating voltage.

The voltage pump can be disabled by writing a one to the Voltage Pump Disable bit in the Control B register (CTRLB.VPD). This can be used to reduce power consumption when the operating voltage is above 2.5V.

The voltage pump uses the asynchronous GCLK\_DAC clock, and requires that the clock frequency be at least four times higher than the sampling period.



### 35.7 Register Summary

Offset	Name	Bit Pos.								
0x00	<a href="#">CTRLA</a>	7:0						RUNSTDBY	ENABLE	SWRST
0x01	<a href="#">CTRLB</a>	7:0	REFSEL[1:0]		BDWP	VPD		LEFTADJ	IOEN	EOEN
0x02	<a href="#">EVCTRL</a>	7:0							EMPTYEO	STARTEI
0x03	Reserved									
0x04	<a href="#">INTENCLR</a>	7:0						SYNCRDY	EMPTY	UNDERRUN
0x05	<a href="#">INTENSET</a>	7:0						SYNCRDY	EMPTY	UNDERRUN
0x06	<a href="#">INTFLAG</a>	7:0						SYNCRDY	EMPTY	UNDERRUN
0x07	<a href="#">STATUS</a>	7:0	SYNCBUSY							
0x08	<a href="#">DATA</a>	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
0x0A ... 0x0B	Reserved									
0x0C	<a href="#">DATABUF</a>	7:0	DATABUF[7:0]							
		15:8	DATABUF[15:8]							

### 35.8 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). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#).

Some registers are synchronized when read and/or written. Synchronization is denoted by the "Write-Synchronized" or the "Read-Synchronized" property in each individual register description. For details, refer to [Synchronization](#).

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

### 35.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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

**Bit 2 – RUNSTDBY** Run in Standby  
 This bit is not synchronized

Value	Description
0	The DAC output buffer is disabled in standby sleep mode.
1	The DAC output buffer can be enabled in standby sleep mode.

**Bit 1 – ENABLE** Enable DAC Controller

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 corresponding bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled or being enabled.

**Bit 0 – SWRST** Software Reset

Writing '0' to this bit has no effect.

Writing '1' to this bit resets all registers in the DAC to their initial state, and the DAC will be disabled.

Writing a '1' to CTRLA.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 SYNCBUSY.SWRST will both be cleared when the reset is complete.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

### 35.8.2 Control B

**Name:** CTRLB  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
	REFSEL[1:0]			BDWP	VPD	LEFTADJ	IOEN	EOEN
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

**Bits 7:6 – REFSEL[1:0]** Reference Selection  
This bit field selects the Reference Voltage for the DAC.

Value	Name	Description
0x0	INTREF	Internal voltage reference
0x1	VDDANA	Analog voltage supply
0x2	VREFA	External reference
0x3		Reserved

**Bit 4 – BDWP** Bypass DATABUF Write Protection  
This bit can bypass DATABUF write protection.

Value	Description
0	DATABUF register is write-protected by Peripheral Access Controller.
1	DATABUF register is not write-protected.

**Bit 3 – VPD** Voltage Pump Disabled  
This bit controls the behavior of the voltage pump.

Value	Description
0	Voltage pump is turned on/off automatically
1	Voltage pump is disabled.

**Bit 2 – LEFTADJ** Left-Adjusted Data  
This bit controls how the 10-bit conversion data is adjusted in the Data and Data Buffer registers.

Value	Description
0	DATA and DATABUF registers are right-adjusted.
1	DATA and DATABUF registers are left-adjusted.

**Bit 1 – IOEN** Internal Output Enable

Value	Description
0	Internal DAC output not enabled.
1	Internal DAC output enabled to be used by the AC.

**Bit 0 – EOEN** External Output Enable

# ATSAMHAXEXXA

## DAC – Digital-to-Analog Converter

Value	Description
0	The DAC output is turned off.
1	The high-drive output buffer drives the DAC output to the $V_{OUT}$ pin.

**35.8.3 Event Control**

**Name:** EVCTRL  
**Offset:** 0x02  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

**Bit 1 – EMPTYEO** Data Buffer Empty Event Output

This bit indicates whether or not the Data Buffer Empty event is enabled and will be generated when the Data Buffer register is empty.

Value	Description
0	Data Buffer Empty event is disabled and will not be generated.
1	Data Buffer Empty event is enabled and will be generated.

**Bit 0 – STARTEI** Start Conversion Event Input

This bit indicates whether or not the Start Conversion event is enabled and data are loaded from the Data Buffer register to the Data register upon event reception.

Value	Description
0	A new conversion will not be triggered on any incoming event.
1	A new conversion will be triggered on any incoming event.

### 35.8.4 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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).

	7	6	5	4	3	2	1	0
						SYNCRDY	EMPTY	UNDERRUN
Access						R/W	R/W	R/W
Reset						0	0	0

**Bit 2 – SYNCRDY** Synchronization Ready Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Synchronization Ready Interrupt Enable bit, which disables the Synchronization Ready interrupt.

Value	Description
0	The Synchronization Ready interrupt is disabled.
1	The Synchronization Ready interrupt is enabled.

**Bit 1 – EMPTY** Data Buffer Empty Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Data Buffer Empty Interrupt Enable bit, which disables the Data Buffer Empty interrupt.

Value	Description
0	The Data Buffer Empty interrupt is disabled.
1	The Data Buffer Empty interrupt is enabled.

**Bit 0 – UNDERRUN** Underrun Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Data Buffer Underrun Interrupt Enable bit, which disables the Data Buffer Underrun interrupt.

Value	Description
0	The Data Buffer Underrun interrupt is disabled.
1	The Data Buffer Underrun interrupt is enabled.

### 35.8.5 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x05  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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).

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

**Bit 2 – SYNCRDY** Synchronization Ready Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Synchronization Ready Interrupt Enable bit, which disables the Synchronization Ready interrupt.

Value	Description
0	The Synchronization Ready interrupt is disabled.
1	The Synchronization Ready interrupt is enabled.

**Bit 1 – EMPTY** Data Buffer Empty Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Data Buffer Empty Interrupt Enable bit, which enables the Data Buffer Empty interrupt.

Value	Description
0	The Data Buffer Empty interrupt is disabled.
1	The Data Buffer Empty interrupt is enabled.

**Bit 0 – UNDERRUN** Underrun Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Data Buffer Underrun Interrupt Enable bit, which enables the Data Buffer Underrun interrupt.

Value	Description
0	The Data Buffer Underrun interrupt is disabled.
1	The Data Buffer Underrun interrupt is enabled.

### 35.8.6 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** PAC Write-Protection

	7	6	5	4	3	2	1	0
						SYNCRDY	EMPTY	UNDERRUN
Access						R/W	R/W	R/W
Reset						0	0	0

**Bit 2 – SYNCRDY** Synchronization Ready Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Synchronization Ready Interrupt Enable bit, which disables the Synchronization Ready interrupt.

Value	Description
0	The Synchronization Ready interrupt is disabled.
1	The Synchronization Ready interrupt is enabled.

**Bit 1 – EMPTY** Data Buffer Empty

This flag is cleared by writing a '1' to it or by writing new data to DATABUF.

This flag is set when data is transferred from DATABUF to DATA, and the DAC is ready to receive new data in DATABUF, and will generate an interrupt request if INTENCLR/SET.EMPTY is one.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Data Buffer Empty interrupt flag.

**Bit 0 – UNDERRUN** Underrun

This flag is cleared by writing a '1' to it.

This flag is set when a start conversion event occurs when DATABUF is empty, and will generate an interrupt request if INTENCLR/SET.UNDERRUN is one.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Underrun interrupt flag.



**35.8.7 Status**

**Name:** STATUS  
**Offset:** 0x07  
**Reset:** 0x00  
**Property:** -

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

**35.8.8 Data DAC**

**Name:** DATA  
**Offset:** 0x08  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Write-Synchronized

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

**Bits 15:0 – DATA[15:0]** Data value to be converted  
 DATA register contains the 10-bit value that is converted to a voltage by the DAC. The adjustment of these 10 bits within the 16-bit register is controlled by CTRLB.LEFTADJ.

**Table 35-1. Valid Data Bits**

CTRLB.LEFTADJ	DATA	Description
0	DATA[9:0]	Right adjusted, 10-bits
1	DATA[15:6]	Left adjusted, 10-bits

**35.8.9 Data Buffer**

**Name:** DATABUF  
**Offset:** 0x0C  
**Reset:** 0x0000  
**Property:** Write-Synchronized

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

**Bits 15:0 – DATABUF[15:0] Data Buffer**  
 DATABUF contains the value to be transferred into DATA register.

## 36. PTC - Peripheral Touch Controller

### 36.1 Overview

The Peripheral Touch Controller (PTC) acquires signals in order to detect a touch on the capacitive sensors. The external capacitive touch sensor is typically formed on a PCB, and the sensor electrodes are connected to the analog front end of the PTC through the I/O pins in the device. The PTC supports both self- and mutual-capacitance sensors.

In the mutual-capacitance mode, sensing is done using capacitive touch matrices in various X-Y configurations, including indium tin oxide (ITO) sensor grids. The PTC requires one pin per X-line and one pin per Y-line.

In the self-capacitance mode, the PTC requires only one pin (Y-line) for each touch sensor.

The number of available pins and the assignment of X- and Y-lines is depending on both package type and device configuration. Refer to the Configuration Summary and I/O Multiplexing table for details.

#### Related Links

[I/O Multiplexing and Considerations](#)

### 36.2 Features

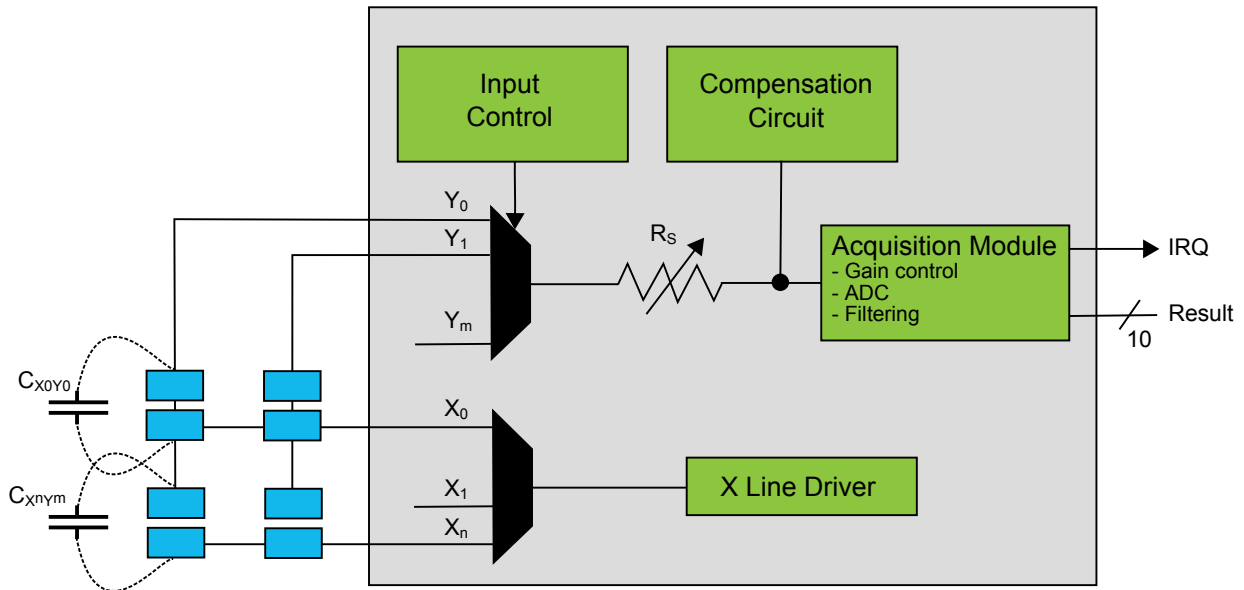
- Low-power, high-sensitivity, environmentally robust capacitive touch buttons, sliders, wheels
- Supports wake-up on touch from Standby Sleep mode
- Supports mutual capacitance and self-capacitance sensing
  - Mix-and-match mutual-and self-capacitance sensors
- One pin per electrode – no external components
- Load compensating charge sensing
  - Parasitic capacitance compensation and adjustable gain for superior sensitivity
- Zero drift over the temperature and  $V_{DD}$  range
  - Auto calibration and recalibration of sensors
- Single-shot charge measurement
- Hardware noise filtering and noise signal desynchronization for high conducted immunity
- Selectable channel change delay allows choosing the settling time on a new channel, as required
- Acquisition-start triggered by command or through auto-triggering feature
- Low CPU utilization through interrupt on acquisition-complete

#### Related Links

[I/O Multiplexing and Considerations](#)

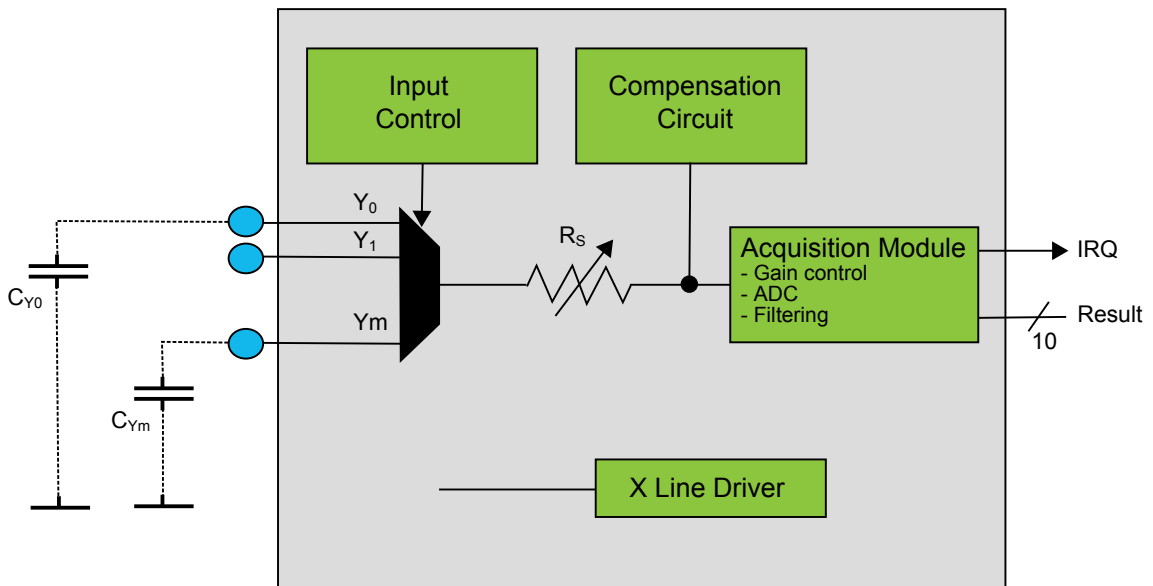
36.3 Block Diagram

Figure 36-1. PTC Block Diagram Mutual-Capacitance



**Note:** For SAMHA0/1 the  $R_S = 0, 20, 50, 100 \text{ K}\Omega$ .

Figure 36-2. PTC Block Diagram Self-Capacitance



**Note:** For SAMHA0/1 the  $R_S = 0, 20, 50, 100 \text{ K}\Omega$ .

## 36.4 Signal Description

**Table 36-1. Signal Description for PTC**

Name	Type	Description
Y[m:0]	Analog	Y-line (Input/Output)
X[n:0]	Digital	X-line (Output)

**Note:** The number of X- and Y-lines are device dependent. Refer to *Configuration Summary* for details.

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

### Related Links

[I/O Multiplexing and Considerations](#)

## 36.5 System Dependencies

In order to use this Peripheral, configure the other components of the system as described in the following sections.

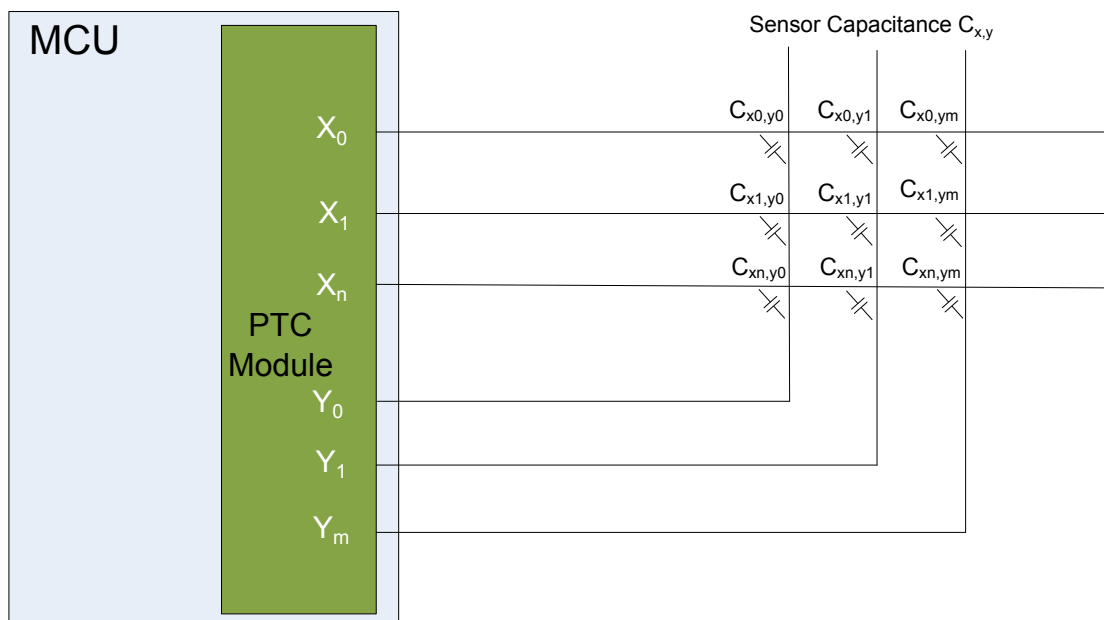
### 36.5.1 I/O Lines

The I/O lines used for analog X-lines and Y-lines must be connected to external capacitive touch sensor electrodes. External components are not required for normal operation. However, to improve the EMC performance, a series resistor of 1 kΩ or more can be used on X-lines and Y-lines.

#### 36.5.1.1 Mutual-Capacitance Sensor Arrangement

A mutual-capacitance sensor is formed between two I/O lines - an X electrode for transmitting and Y electrode for sensing. The mutual capacitance between the X and Y electrode is measured by the Peripheral Touch Controller.

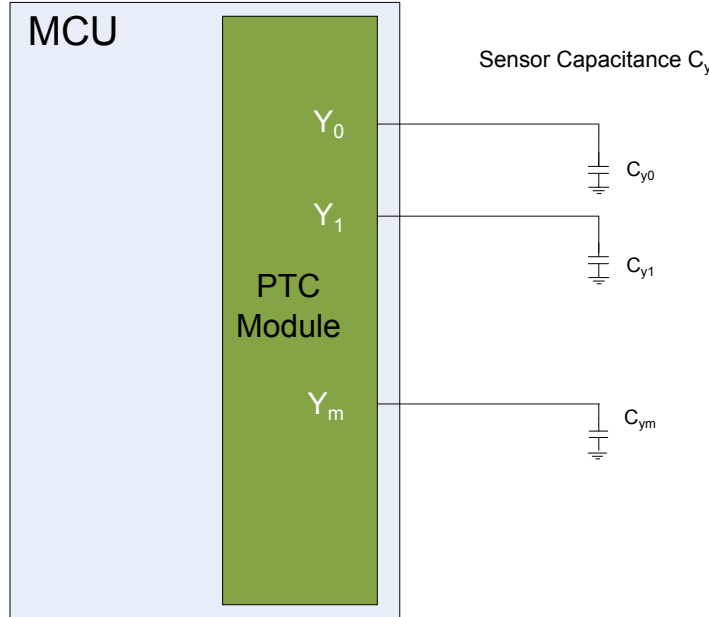
**Figure 36-3. Mutual Capacitance Sensor Arrangement**



**36.5.1.2 Self-Capacitance Sensor Arrangement**

A self-capacitance sensor is connected to a single pin on the Peripheral Touch Controller through the Y electrode for sensing the signal. The sense electrode capacitance is measured by the Peripheral Touch Controller.

**Figure 36-4. Self-Capacitance Sensor Arrangement**



For more information about designing the touch sensor, refer to [Buttons, Sliders and Wheels Touch Sensor Design Guide](#).

**36.5.2 Clocks**

The PTC is clocked by the GCLK\_PTC clock. The PTC operates from an asynchronous clock source and the operation is independent of the main system clock and its derivative clocks, such as the peripheral bus clock (CLK\_APB). A number of clock sources can be selected as the source for the asynchronous GCLK\_PTC. The clock source is selected by configuring the Generic Clock Selection ID in the Generic Clock Control register. For more information about selecting the clock sources, refer to *GCLK - Generic Clock Controller*.

The selected clock must be enabled in the Power Manager, before it can be used by the PTC. By default these clocks are disabled. The frequency range of GCLK\_PTC is 400kHz to 4MHz.

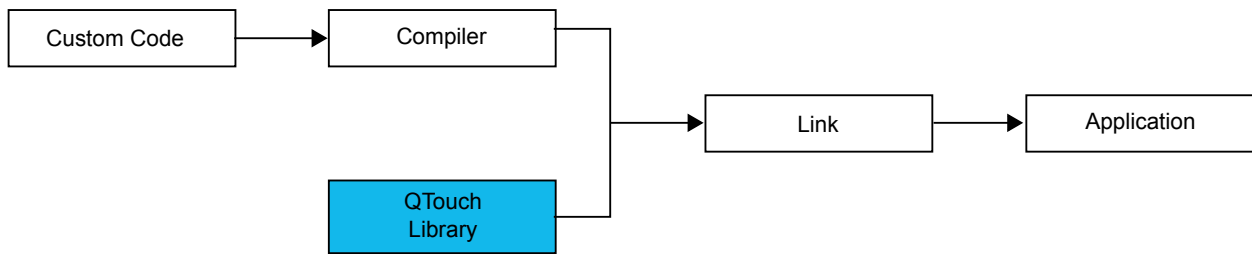
**Related Links**

- [GCLK - Generic Clock Controller](#)
- [PM – Power Manager](#)

**36.6 Functional Description**

In order to access the PTC, the user must use the Atmel Start QTouch Configurator to configure and link the QTouch Library firmware with the application software. QTouch Library can be used to implement buttons, sliders, and wheels in a variety of combinations on a single interface.

**Figure 36-5. QTouch Library Usage**



For more information about QTouch Library, refer to the [QTouch Library Peripheral Touch Controller User Guide](#).



## 37. Electrical Characteristics

### 37.1 Disclaimer

All typical values are measured at  $T = 25^{\circ}\text{C}$  unless otherwise specified. All minimum and maximum values are valid across operating temperature and voltage unless otherwise specified.

### 37.2 Absolute Maximum Ratings

#### 37.2.1 Maximum Ratings of the SiP

Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any 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 37-1.**

Parameters	Symbol	Minimal	Typical	Maximum	Unit
ESD according to IBEE LIN EMC Test specification 1.0 following IEC 61000-4-2 - Pin VS, LIN to GND (with external circuitry acc. applications diagram)		$\pm 6$			kV
ESD HBM following STM5.1 with 1.5k $\Omega$ /100pF - Pin VS, LIN to GND		$\pm 6$			kV
HBM ESD ANSI/ESD-STM5.1 JESD22-A114 AEC-Q100 (002)		$\pm 3$			kV
CDM ESD STM 5.3.1		$\pm 500$			V
Machine Model ESD AEC-Q100-RevF(003)		$\pm 100$			V
Storage temperature	$T_S$	-55		+150	$^{\circ}\text{C}$

#### 37.2.2 Maximum Ratings of the LIN-SBC

Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any 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 37-2.**

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Supply voltage $V_S$	$V_S$	-0.3		+40	V
Pulse time $\leq 500\text{ms}$ $T_C = 25^{\circ}\text{C}$ Output current $I_{VCC} \leq 85\text{mA}$	$V_S$			+43.5	V
Pulse time $\leq 2\text{min}$ $T_C = 25^{\circ}\text{C}$ Output current $I_{VCC} \leq 85\text{mA}$	$V_S$			28	V

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Logic pins voltage levels (RxD, TxD, EN, NRES)	$V_{Logic}$	-0.3		+5.5	V
Logic pins output DC currents	$I_{Logic}$	-5		+5	mA

**Table 37-3.**

Parameter	Symbol	Minimum	Typical	Maximum	Unit
LIN					
DC voltage	$V_{LIN}$	-27		+40	V
Pulse time < 500ms		-27		+43.5	V
$V_{VCC}$					
DC voltage	$V_{VCC}$	-0.3		+5.5	V
DC input current	$I_{VCC}$	-		+200	mA
Virtual Junction temperature	$T_{vj}$	-40		+150	°C

### 37.2.3 Maximum Ratings of the Microcontroller

Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any 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 37-4.**

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Power supply voltage	$V_{DD}$	0		3.8	V
Current into a VDD pin	$I_{DD}$			92 <sup>(1)</sup>	mA
Current out of a GND pin	$I_{GND}$			130 <sup>(1)</sup>	mA
Pin voltage with respect to GND and VDD	$V_{PIN}$	GND-0.3V		$V_{DD}+0.3V$	V
Storage temperature	$T_{Storage}$	-60		+150	°C

**Note:**

- Maximum source current is 46mA and maximum sink current is 65mA per cluster. A cluster is a group of GPIOs as shown in the [GPIO Clusters](#) table. Also note that each VDD/GND pair is connected to two clusters so current consumption through the pair will be a sum of the clusters source/sink currents.

### 37.3 LIN System Basis Chip (LIN-SBC) Block

#### 37.3.1 Thermal Characteristics

**Table 37-5.**

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Thermal resistance junction top die to bottom die	$R_{thJj}$		16		K/W
Thermal resistance junction bottom die to heat slug	$R_{thjC}$		8		K/W
Thermal resistance junction to ambient, where heat slug is soldered to PCB according to JEDEC	$R_{thja}$		64		K/W
Thermal shutdown of VCC regulator	$T_{VCCoff}$	150	165	180	°C
Thermal shutdown of LIN output	$T_{LINoff}$	150	165	180	°C
Thermal shutdown hysteresis	$T_{hys}$		10		°C

#### 37.3.2 Electrical Characteristics

**Table 37-6. Electrical Characteristics**

$5V < V_S < 28V$ ,  $-40^{\circ}C < T_{vj} < 150^{\circ}C$ ; unless otherwise specified all values refer to GND pins.

No.	Parameters	Test Conditions	Pin	Symbol	Min.	Typ.	Max.	Unit	Type*
1	VS pin								
1.1	Nominal DC voltage range		VS	$V_S$	5	13.5	28	V	A
1.2	Supply current in sleep mode	Sleep mode $V_{LIN} > V_S 0.5V$ $V_S < 14V$ , $T = 27^{\circ}C$	VS	$I_{VSsleep}$	6	9	12	$\mu A$	B

# ATSAMHAXEXXA

## Electrical Characteristics

No.	Parameters	Test Conditions	Pin	Symbol	Min.	Typ.	Max.	Unit	Type*
		Sleep mode $V_{LIN} > V_S 0.5V$ $V_S < 14V$	VS	$I_{VSsleep}$	3	10	15	$\mu A$	A
		Sleep mode, $V_{LIN} = 0V$ bus shorted to GND $V_S < 14V$	VS	$I_{VSsleep\_short}$	20	50	100	$\mu A$	A
1.3	Supply current in silent mode (SBC) / Active mode (voltage regulator)	Bus recessive $5.5V < V_S < 14V$ without load at VCC $T = 27^\circ C$	VS	$I_{VSsilent}$	30	47	58	$\mu A$	B
		Bus recessive $5.5V < V_S < 14V$ without load at VCC	VS	$I_{VSsilent}$	30	50	64	$\mu A$	A
		Bus recessive $2.0V < V_S < 5.5V$ without load at VCC	VS	$I_{VSsilent}$	50	130	170	$\mu A$	A
		Silent mode $5.5V < V_S < 14V$ bus shorted to GND without load at VCC	VS	$I_{VSsilent\_short}$	50	80	120	$\mu A$	A
1.4	Supply current in normal mode	Bus recessive $V_S < 14V$ without load at VCC	VS	$I_{VSrec}$	150	230	290	$\mu A$	A
1.5	Supply current in normal mode	Bus dominant (internal LIN pull-up resistor active) $V_S < 14V$ without load at VCC	VS	$I_{VSdom}$	200	700	950	$\mu A$	A
1.6	Supply current in fail-safe mode	Bus recessive $5.5V < V_S < 14V$ without load at VCC	VS	$I_{VSfail}$	40	55	80	$\mu A$	A
		Bus recessive $2.0V < V_S < 5.5V$ without load at VCC	VS	$I_{VSfail}$	50	130	170	$\mu A$	A
1.7	VS undervoltage threshold (switching from normal to fail-safe mode)	Decreasing supply voltage	VS	$V_{VS\_th\_N\_F\_down}$	3.9	4.3	4.7	V	A
		Increasing supply voltage	VS	$V_{VS\_th\_F\_N\_up}$	4.1	4.6	4.9	V	A
1.8	VS undervoltage hysteresis		VS	$V_{VS\_hys\_F\_N}$	0.1	0.25	0.4	V	A
1.9	VS operation threshold (switching to unpowered mode)	Switch to unpowered mode	VS	$V_{VS\_th\_U\_down}$	1.9	2.05	2.3	V	A

# ATSAMHAXEXXA

## Electrical Characteristics

No.	Parameters	Test Conditions	Pin	Symbol	Min.	Typ.	Max.	Unit	Type*
		Switch from unpowered to fail-safe mode	VS	V <sub>VS_th_U_F_up</sub>	2.0	2.25	2.4	V	A
1.10	VS undervoltage hysteresis		VS	V <sub>VS_hys_U</sub>	0.1	0.2	0.3	V	A
3	TXD input/output pin (only SBC)								
3.1	Low-level voltage input		TXD	V <sub>TXDL</sub>	-0.3		+0.8	V	A
3.2	High-level voltage input		TXD	V <sub>TXDH</sub>	2		V <sub>CC</sub> + 0.3V	V	A
3.3	Pull-up resistor	V <sub>TXD</sub> = 0V	TXD	R <sub>TXD</sub>	40	70	100	kΩ	A
3.4	High-level leakage current	V <sub>TXD</sub> = V <sub>CC</sub>	TXD	I <sub>TXD</sub>	-3		+3	μA	A
4	EN input pin (only SBC)								
4.1	Low-level voltage input		EN	V <sub>ENL</sub>	-0.3		+0.8	V	A
4.2	High-level voltage input		EN	V <sub>ENH</sub>	2		V <sub>CC</sub> + 0.3V	V	A
4.3	Pull-down resistor	V <sub>EN</sub> = V <sub>CC</sub>	EN	R <sub>EN</sub>	50	125	200	kΩ	A
4.4	Low-level input current	V <sub>EN</sub> = 0V	EN	I <sub>EN</sub>	-3		+3	μA	A
5	NRES open drain output pin								
5.1	Low-level output voltage	V <sub>S</sub> ≥ 5.5V I <sub>NRES</sub> = 2mA	NRES	V <sub>NRESL</sub>		0.2	0.4	V	A
5.2	Undervoltage reset time	V <sub>VS</sub> ≥ 5.5V C <sub>NRES</sub> = 20pF	NRES	t <sub>Reset</sub>	2	4	6	ms	A
5.3	Reset debounce time for falling edge	V <sub>VS</sub> ≥ 5.5V C <sub>NRES</sub> = 20pF	NRES	t <sub>res_f</sub>	0.5		10	μs	A
5.4	Switch off leakage current	V <sub>NRES</sub> = 5.5V	NRES	I <sub>NRES_L</sub>	-3		+3	μA	A
8	VCC voltage regulator								
8.1	Output voltage V <sub>CC</sub>	4V < V <sub>S</sub> < 18V (0mA to 50mA)	VCC	V <sub>CCnor</sub>	3.234		3.366	V	A
		4.5V < V <sub>S</sub> < 18V (0mA to 85mA)	VCC	V <sub>CCnor</sub>	3.234		3.366	V	C
8.2	Output voltage V <sub>CC</sub> at low V <sub>S</sub>	3V < V <sub>S</sub> < 4V	VCC	V <sub>CClow</sub>	V <sub>VS</sub> - V <sub>D</sub>		3.366	V	A
8.3	Regulator drop voltage	V <sub>S</sub> > 3V, I <sub>VCC</sub> = -15mA	VCC	V <sub>D1</sub>		100	150	mV	A
8.4	Regulator drop voltage	V <sub>S</sub> > 3V, I <sub>VCC</sub> = -50mA	VCC	V <sub>D2</sub>		300	500	mV	A

# ATSAMHAXEXXA

## Electrical Characteristics

No.	Parameters	Test Conditions	Pin	Symbol	Min.	Typ.	Max.	Unit	Type*
8.5	Line regulation maximum	$4V < V_S < 18V$	VCC	$V_{CCline}$		0.1	0.2	%	A
8.6	Load regulation maximum	$5mA < I_{VCC} < 50mA$	VCC	$V_{CCload}$		0.1	0.5	%	A
8.7	Output current limitation	$V_S > 4V$	VCC	$I_{VCClim}$		-180	-120	mA	A
8.8	Load capacity	MLC capacitor (ESR < 0.8 $\Omega$ )	VCC	$C_{load}$	3.7	4.7		$\mu F$	D
8.9	VCC undervoltage threshold (NRES ON)	Referred to VCC $V_S > 4V$	VCC	$V_{VCC\_th\_uv\_down}$	2.3	2.5	2.8	V	A
	VCC undervoltage threshold (NRES OFF)	Referred to VCC $V_S > 4V$	VCC	$V_{VCC\_th\_uv\_up}$	2.5	2.6	2.9	V	A
8.10	Hysteresis of VCC undervoltage threshold	Referred to VCC $V_S > 4V$	VCC	$V_{VCC\_hys\_uv}$	100	200	300	mV	A
8.11	Ramp-up time $V_S > 4V$ to $V_{CC} = 3.3V$	$C_{VCC} = 4.7\mu F$ $I_{load} = -5mA$ at VCC	VCC	$t_{VCC}$		1	1.5	ms	A
10	LIN bus driver (only SBC): bus load conditions: Load 1 (small): 1nF, 1k $\Omega$ ; Load 2 (large): 10nF, 500 $\Omega$ ; $C_{RXD} = 20pF$ , Load 3 (medium): 6.8nF, 660 $\Omega$ characterized on samples 12.7 and 12.8 specifies the timing parameters for proper operation at 20kb/s and 12.9 and 12.10 at 10.4kb/s								
10.1	Driver recessive output voltage	Load1/Load2	LIN	$V_{BUSrec}$	$0.9 \times V_S$		$V_S$	V	A
10.2	Driver dominant voltage	$V_{V_S} = 7V$ $R_{load} = 500\Omega$	LIN	$V_{LoSUP}$			1.2	V	A
10.3	Driver dominant voltage	$V_{V_S} = 18V$ $R_{load} = 500\Omega$	LIN	$V_{HiSUP}$			2	V	A
10.4	Driver dominant voltage	$V_{V_S} = 7V$ $R_{load} = 1000\Omega$	LIN	$V_{LoSUP\_1k}$	0.6			V	A
10.5	Driver dominant voltage	$V_{V_S} = 18V$ $R_{load} = 1000\Omega$	LIN	$V_{HiSUP\_1k}$	0.8			V	A
10.6	Pull-up resistor to $V_S$	The serial diode is mandatory	LIN	$R_{LIN}$	20	30	47	k $\Omega$	A
10.7	Voltage drop at the serial diodes	In pull-up path with $R_{slave}$ $I_{SerDiode} = 10mA$	LIN	$V_{SerDiode}$	0.4		1.0	V	D
10.8	LIN current limitation $V_{BUS} = V_{Bat\_max}$		LIN	$I_{BUS\_LIM}$	40	120	200	mA	A
10.9	Input leakage current at the receiver including pull-up resistor as specified	Input leakage current driver off $V_{BUS} = 0V$ $V_{Bat} = 12V$	LIN	$I_{BUS\_PAS\_dom}$	-1	-0.35		mA	A

# ATSAMHAXEXXA

## Electrical Characteristics

No.	Parameters	Test Conditions	Pin	Symbol	Min.	Typ.	Max.	Unit	Type*
10.10	Leakage current LIN recessive	Driver off $8V < V_{Bat} < 18V$ $8V < V_{BUS} < 18V$ $V_{BUS} < V_{Bat}$	LIN	$I_{BUS\_PAS\_rec}$		10	20	$\mu A$	A
10.11	Leakage current when control unit disconnected from ground. Loss of local ground must not affect communication in the residual network	$GND_{Device} = V_S$ $V_{Bat} = 12V$ $0V < V_{BUS} < 18V$	LIN	$I_{BUS\_NO\_gnd}$	-10	+0.5	+10	$\mu A$	A
10.12	Leakage current at disconnected battery. Node has to sustain the current that can flow under this condition. Bus must remain operational under this condition.	$V_{Bat}$ disconnected $V_{SUP\_Device} = GND$ $0V < V_{BUS} < 18V$	LIN	$I_{BUS\_NO\_bat}$		0.1	2	$\mu A$	A
10.13	Capacitance on pin LIN to GND		LIN	$C_{LIN}$			20	pF	D
11	LIN bus receiver (only SBC)								
11.1	Center of receiver threshold	$V_{BUS\_CNT} = (V_{th\_dom} + V_{th\_rec})/2$	LIN	$V_{BUS\_CNT}$	$0.475 \times V_S$	$0.5 \times V_S$	$0.525 \times V_S$	V	A
11.2	Receiver dominant state	$V_{EN} = 3.3V$	LIN	$V_{BUSdom}$	-27		$0.4 \times V_S$	V	A
11.3	Receiver recessive state	$V_{EN} = 3.3V$	LIN	$V_{BUSrec}$	$0.6 \times V_S$		40	V	A
11.4	Receiver input hysteresis	$V_{hys} = V_{th\_rec} - V_{th\_dom}$	LIN	$V_{BUShys}$	$0.028 \times V_S$	$0.1 \times V_S$	$0.175 \times V_S$	V	A
11.5	Pre-wake detection LIN high-level input voltage		LIN	$V_{LINH}$	$V_S - 2V$		$V_S + 0.3V$	V	A
11.6	Pre-wake detection LIN low-level input voltage	Activates the LIN receiver	LIN	$V_{LINL}$	-27		$V_S - 3.3V$	V	A
12	Internal timers (only SBC)								
12.1	Dominant time for wake-up via LIN bus	$V_{LIN} = 0V$	LIN	$t_{bus}$	50	100	150	$\mu s$	A
12.2	Time delay for mode change from fail-safe into normal mode via EN pin	$V_{EN} = 3.3V$	EN	$t_{norm}$	5	15	20	$\mu s$	A

# ATSAMHAXEXXA

## Electrical Characteristics

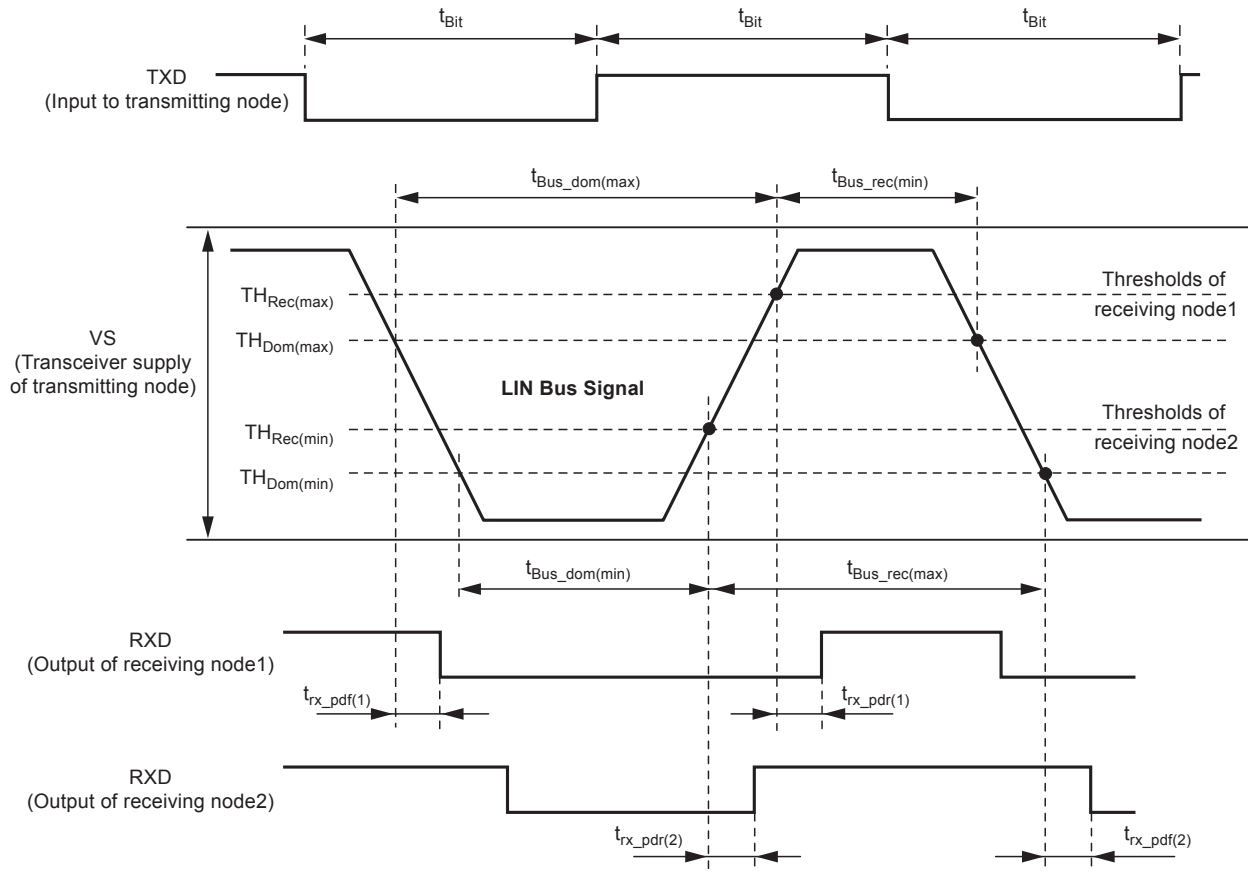
No.	Parameters	Test Conditions	Pin	Symbol	Min.	Typ.	Max.	Unit	Type*
12.3	Time delay for mode change from normal mode to sleep mode via EN pin	$V_{EN} = 0V$	EN	$t_{sleep}$	5	15	20	$\mu s$	A
12.5	TXD dominant time-out time	$V_{TXD} = 0V$	TXD	$t_{dom}$	20	40	60	ms	A
12.6	Time delay for mode change from silent mode into normal mode via EN pin	$V_{EN} = 3.3V$	EN	$t_{s\_n}$	5	15	40	$\mu s$	A
12.7	Duty cycle 1	$TH_{Rec(max)} = 0.744 \times V_S$ $TH_{Dom(max)} = 0.581 \times V_S$ $V_S = 7.0V$ to $18V$ $t_{Bit} = 50\mu s$ $D1 = t_{bus\_rec(min)}/(2 \times t_{Bit})$	LIN	D1	0.396				A
12.8	Duty cycle 2	$TH_{Rec(min)} = 0.422 \times V_S$ $TH_{Dom(min)} = 0.284 \times V_S$ $V_S = 7.6V$ to $18V$ $t_{Bit} = 50\mu s$ $D2 = t_{bus\_rec(max)}/(2 \times t_{Bit})$	LIN	D2			0.581		A
12.9	Duty cycle 3	$TH_{Rec(max)} = 0.778 \times V_S$ $TH_{Dom(max)} = 0.616 \times V_S$ $V_S = 7.0V$ to $18V$ $t_{Bit} = 96\mu s$ $D3 = t_{bus\_rec(min)}/(2 \times t_{Bit})$	LIN	D3	0.417				A
12.10	Duty cycle 4	$TH_{Rec(min)} = 0.389 \times V_S$ $TH_{Dom(min)} = 0.251 \times V_S$ $V_S = 7.6V$ to $18V$ $t_{Bit} = 96\mu s$ $D4 = t_{bus\_rec(max)}/(2 \times t_{Bit})$	LIN	D4			0.590		A
12.11	Slope time falling and rising edge at LIN	$V_S = 7.0V$ to $18V$	LIN	$t_{SLOPE\_fall}$ $t_{SLOPE\_rise}$	3.5		22.5	$\mu s$	A
13	Receiver electrical AC parameters of the LIN physical layer LIN receiver, RXD load conditions: $C_{RXD} = 20pF$								



No.	Parameters	Test Conditions	Pin	Symbol	Min.	Typ.	Max.	Unit	Type*
13.1	Propagation delay of receiver	$V_S = 7.0V$ to $18V$ $t_{rx\_pd} = \max(t_{rx\_pdr}, t_{rx\_pdf})$	RXD	$t_{rx\_pd}$			6	$\mu s$	A
13.2	Symmetry of receiver propagation delay rising edge minus falling edge	$V_S = 7.0V$ to $18V$ $t_{rx\_sym} = t_{rx\_pdr} - t_{rx\_pdf}$	RXD	$t_{rx\_sym}$	-2		+2	$\mu s$	A

\*) Type means: A = 100% tested, B = 100% correlation tested, C = Characterized on samples, D = Design parameter

### 37.3.3 BUS Timing Characteristics



### 37.4 General Operating Ratings

The microcontroller must operate within the ratings listed in Table 37-2 in order for all other electrical characteristics and typical characteristics of the device to be valid.

**Table 37-7. General Operating Conditions**

Parameter	Condition	Symbol	Minimum	Typical	Maximum	Unit
Power supply voltage		V <sub>DD</sub>	2.7	3.3	3.63	V
Analog supply voltage		V <sub>DDANA</sub>	2.7	3.3	3.63	V
Temperature range case		T <sub>C</sub>	-40	25	105	°C
Temperature range case SAMHA0		T <sub>C</sub>	-40	25	115	°C
Virtual Junction temperature		T <sub>vj</sub>	-	-	125	°C

## 37.5 Thermal Considerations

### 37.5.1 Thermal Resistance Data

The following Table summarizes the thermal resistance data depending on the package.

**Table 37-8. Thermal Resistance Data**

Package Type	θ <sub>JA</sub>	θ <sub>JC</sub>
48-pin VQFN	54°C/W	24°C/W

### 37.5.2 Junction Temperature

The average chip-junction temperature, T<sub>J</sub>, in °C can be obtained from the following:

1.  $T_J = T_A + (P_D \times \theta_{JA})$
2.  $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

where:

- θ<sub>JA</sub> = Package thermal resistance, Junction-to-ambient (°C/W), see Thermal Resistance Data
- θ<sub>JC</sub> = Package thermal resistance, Junction-to-case thermal resistance (°C/W), see Thermal Resistance Data
- θ<sub>HEATSINK</sub> = Thermal resistance (°C/W) specification of the external cooling device
- P<sub>D</sub> = Device power consumption (W)
- T<sub>A</sub> = Ambient temperature (°C)

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature T<sub>J</sub> in °C.

### 37.6 Supply Characteristics

The following characteristics are applicable to the operating temperature range:  $T_C = -40^{\circ}\text{C}$  to  $105^{\circ}\text{C}$ , unless otherwise specified and are valid for a junction temperature up to  $T_{VJ} = 125^{\circ}\text{C}$ . Refer to *Power Supply and Start-Up Considerations*.

**Table 37-9. Supply Characteristics**

Conditions	Symbol	Voltage		
		Min.	Max.	Unit
Full Voltage Range	$V_{DDIO}$ $V_{DDIN}$ $V_{DDANA}$	2.7	3.63	V

**Table 37-10. Supply Rates**

Conditions	Symbol	Fall Rate	Rise Rate	Unit
		Max.	Max.	
DC supply peripheral I/Os, internal regulator and analog supply voltage	$V_{DDIO}$ $V_{DDIN}$ $V_{DDANA}$	0.05	0.1	V/ $\mu\text{s}$

### 37.7 Maximum Clock Frequencies

**Table 37-11. Maximum GCLK Generator Output Frequencies**

Description	Conditions	Symbol	Max.	Unit
GCLK Generator Output Frequency	Undivided	$f_{GCLKGEN0} / f_{GCLK\_MAIN}$ $f_{GCLKGEN1}$ $f_{GCLKGEN2}$	96	MHz
	Divided	$f_{GCLKGEN3}$ $f_{GCLKGEN4}$ $f_{GCLKGEN5}$	48	MHz

**Table 37-12. Maximum Peripheral Clock Frequencies**

Description	Symbol	Max.	Unit
CPU clock frequency	$f_{CPU}$	48	MHz
AHB clock frequency	$f_{AHB}$	48	MHz
APBA clock frequency	$f_{APBA}$	48	MHz
APBB clock frequency	$f_{APBB}$	48	MHz
APBC clock frequency	$f_{APBC}$	48	MHz

# ATSAMHAXEXXA

## Electrical Characteristics

Description	Symbol	Max.	Unit
DFLL48M Reference clock frequency	$f_{\text{GCLK\_DFLL48M\_REF}}$	33	KHz
FDPLL96M Reference clock frequency	$f_{\text{GCLK\_DPLL}}$	2	MHz
FDPLL96M 32k Reference clock frequency	$f_{\text{GCLK\_DPLL\_32K}}$	32	KHz
WDT input clock frequency	$f_{\text{GCLK\_WDT}}$	48	MHz
RTC input clock frequency	$f_{\text{GCLK\_RTC}}$	48	MHz
EIC input clock frequency	$f_{\text{GCLK\_EIC}}$	48	MHz
EVSYS channel 0 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_0}}$	48	MHz
EVSYS channel 1 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_1}}$	48	MHz
EVSYS channel 2 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_2}}$	48	MHz
EVSYS channel 3 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_3}}$	48	MHz
EVSYS channel 4 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_4}}$	48	MHz
EVSYS channel 5 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_5}}$	48	MHz
EVSYS channel 6 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_6}}$	48	MHz
EVSYS channel 7 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_7}}$	48	MHz
EVSYS channel 8 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_8}}$	48	MHz
EVSYS channel 9 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_9}}$	48	MHz
EVSYS channel 10 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_10}}$	48	MHz
EVSYS channel 11 input clock frequency	$f_{\text{GCLK\_EVSYS\_CHANNEL\_11}}$	48	MHz
Common SERCOM slow input clock frequency	$f_{\text{GCLK\_SERCOMx\_SLOW}}$	48	MHz
SERCOM0 input clock frequency	$f_{\text{GCLK\_SERCOM0\_CORE}}$	48	MHz
SERCOM1 input clock frequency	$f_{\text{GCLK\_SERCOM1\_CORE}}$	48	MHz
SERCOM2 input clock frequency	$f_{\text{GCLK\_SERCOM2\_CORE}}$	48	MHz
SERCOM3 input clock frequency	$f_{\text{GCLK\_SERCOM3\_CORE}}$	48	MHz
SERCOM4 input clock frequency	$f_{\text{GCLK\_SERCOM4\_CORE}}$	48	MHz
SERCOM5 input clock frequency	$f_{\text{GCLK\_SERCOM5\_CORE}}$	48	MHz
TCC0, TCC1 input clock frequency	$f_{\text{GCLK\_TCC0}}, \text{GCLK\_TCC1}$	96	MHz
TCC2, TC3 input clock frequency	$f_{\text{GCLK\_TCC2}}, \text{GCLK\_TC3}$	96	MHz
TC4, TC5 input clock frequency	$f_{\text{GCLK\_TC4}}, \text{GCLK\_TC5}$	48	MHz
TC6, TC7 input clock frequency	$f_{\text{GCLK\_TC6}}, \text{GCLK\_TC7}$	48	MHz
ADC input clock frequency	$f_{\text{GCLK\_ADC}}$	48	MHz
AC digital input clock frequency	$f_{\text{GCLK\_AC\_DIG}}$	48	MHz
AC analog input clock frequency	$f_{\text{GCLK\_AC\_ANA}}$	64	kHz

Description	Symbol	Max.	Unit
DAC input clock frequency	$f_{\text{GCLK\_DAC}}$	350	kHz
PTC input clock frequency	$f_{\text{GCLK\_PTC}}$	48	MHz

### 37.8 Power Consumption

The values in [Table 37-13](#) are measured values of power consumption of the microcontroller through the VDDIN, VDDIO and VDDANA pins under the following conditions, except where noted. Note that the values are given for case temperatures. The power consumption of the SBC through the VS pin found in [Table 37-6](#) has to be added to obtain the full power consumption for the device.

- Operating conditions
  - $V_{\text{VDDIN}} = 3.3\text{V}$
  - $V_{\text{VDDIN}} = 2.7\text{V}$ , CPU is running on Flash with 1 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 (32 kHz crystal oscillator) running with external 32kHz crystal
  - DFLL48M using XOSC32K as reference and running at 48 MHz
- 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
- Cache enabled
- BOD33 disabled

**Table 37-13. Current Consumption**

Mode	Conditions	$T_{\text{VJ}}$	$V_{\text{CC}}$	Typ.	Max.	Unit
ACTIVE	CPU running a While(1) algorithm	25°C	3.3V	3.32	3.63	mA
		105°C	3.3V	3.57	3.98	
	CPU running a While(1) algorithm, with GCLKIN as reference	25°C	3.3V	$64 \times \text{Freq} + 110$	$70 \times \text{Freq} + 131$	$\mu\text{A}$ (with freq in MHz)
		105°C	3.3V	$65 \times \text{Freq} + 342$	$65 \times \text{Freq} + 764$	

# ATSAMHAXEXXA

## Electrical Characteristics

Mode	Conditions	T <sub>VJ</sub>	V <sub>CC</sub>	Typ.	Max.	Unit	
	CPU running a Fibonacci algorithm	25°C	3.3V	4.03	4.35	mA	
		105°C	3.3V	4.29	4.76		
	CPU running a Fibonacci algorithm, with GCLKIN as reference	25°C	3.3V	79 × Freq + 110	85 × Freq + 133	μA (with freq in MHz)	
		105°C	3.3V	80 × Freq + 346	81 × Freq + 771		
	CPU running a CoreMark algorithm	25°C	3.3V	5.08	5.63	mA	
		105°C	3.3V	5.41	5.95		
	CPU running a CoreMark algorithm, with GCLKIN as reference	25°C	3.3V	101 × Freq + 113	110 × Freq + 132	μA (with freq in MHz)	
		105°C	3.3V	103 × Freq + 347	104 × Freq + 748		
	IDLE0	Default operating conditions	25°C	3.3V	2.24	2.41	mA
			105°C	3.3V	2.49	2.92	
	IDLE1	Default operating conditions	25°C	3.3V	1.69	1.82	
			105°C	3.3V	1.91	2.33	
IDLE2	Default operating conditions	25°C	3.3V	1.23	1.32		
		105°C	3.3V	1.44	1.85		
STANDBY <sup>(1)</sup>	XOSC32K running	25°C	3.3V	4.6	15.0	μA	
	RTC running at 1kHz	105°C	3.3V	95.0	390.0		
	XOSC32K and RTC stopped	25°C	3.3V	3.4	14.0		
		105°C	3.3V	94.0	388.0		
STANDBY <sup>(2)</sup>	XOSC32K running	25°C	3.3V	61.0	72.0	μA	
	RTC running at 1kHz	105°C	3.3V	174.0	452.0		
	XOSC32K and RTC stopped	25°C	3.3V	60.0	71.0		
		105°C	3.3V	173.0	450.0		

1. Measurements done with SYSCTRL.VREG.RUNSTDBY=0 (low power configuration).
2. Measurements done with SYSCTRL.VREG.RUNSTDBY=1 (normal configuration).

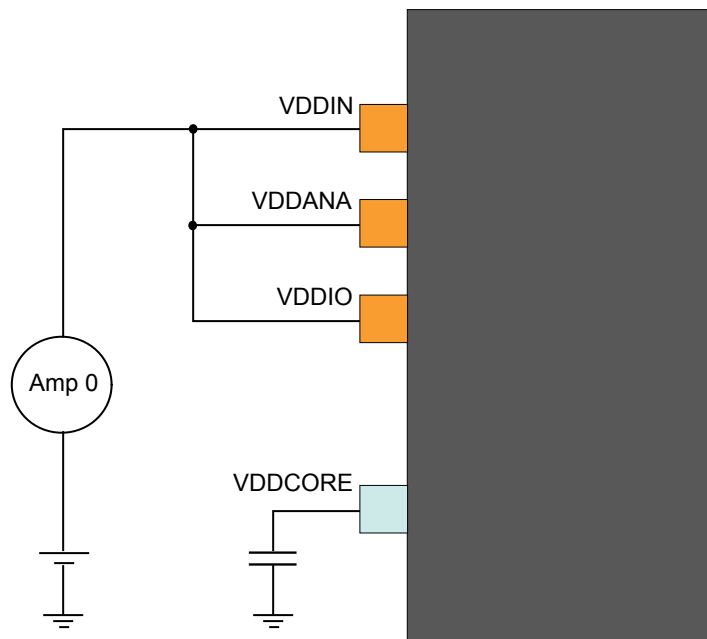
**Table 37-14. Wake-up Time<sup>(1)</sup>**

Mode	T <sub>C</sub>	Typ.	Unit
IDLE0	25°C	2.3	μs
IDLE1		21.1	
IDLE2		22.0	

Mode	$T_C$	Typ.	Unit
STANDBY		29.6	
IDLE0	105°C	2.3	μs
IDLE1		22.9	
IDLE2		23.8	
STANDBY		29.8	

1. OSC8M used as main clock source, cache disabled.

**Figure 37-1. Measurement Schematic**



## 37.9 Peripheral Power Consumption

### 37.9.1 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 these default conditions, the power consumption  $I_{\text{default}}$  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  $I_{\text{periph}}$
- 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 is the value ( $I_{\text{periph}} - I_{\text{default}}$ ), using the same measurement method as for global power consumption measurement

**Table 37-15. Typical Peripheral Current Consumption**

Peripheral	Conditions	Typical	Unit
RTC	fGCLK_RTC = 32kHz, 32bit counter mode	7.4	μA
WDT	fGCLK_WDT = 32kHz, normal mode with EW	5.5	
AC	Both fGCLK = 8MHz, Enable both COMP	31.3	
TCx <sup>(1)</sup>	fGCLK = 8MHz, Enable + COUNTER in 8bit mode	50	
TCC2	fGCLK = 8MHz, Enable + COUNTER	95.5	
TCC1	fGCLK = 8MHz, Enable + COUNTER	167.5	
TCC0	fGCLK = 8MHz, Enable + COUNTER	180.3	



Peripheral	Conditions	Typical	Unit
SERCOMx.I2CM <sup>(2)</sup>	fGCLK = 8MHz, Enable	69.7	
SERCOMx.I2CS	fGCLK = 8MHz, Enable	29.2	
SERCOMx.SPI	fGCLK = 8MHz, Enable	64.6	
SERCOMx.USART	fGCLK = 8MHz, Enable	65.5	
DMAC <sup>(3)</sup>	RAM to RAM transfer	399.5	

**Note:**

1. All TCs from 4 to 7 share the same power consumption values.
2. All SERCOMs from 0 to 5 share the same power consumption values.
3. The value includes the power consumption of the R/W access to the RAM.

### 37.10 HA1\_I/O Pin Characteristics

#### 37.10.1 Normal I/O Pins

**Table 37-16. Normal I/O Pins Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Pull-up - Pull-down resistance		R <sub>PULL</sub>	20	40	60	kΩ
Input low-level voltage	V <sub>DD</sub> = 2.7V-3.63V	V <sub>IL</sub>	-	-	0.3 × V <sub>DD</sub>	V
Input high-level voltage	V <sub>DD</sub> = 2.7V-3.63V	V <sub>IH</sub>	0.55 × V <sub>DD</sub>	-	-	
Output low-level voltage	V <sub>DD</sub> > 2.7V, I <sub>OL</sub> maxI	V <sub>OL</sub>	-	0.1 × V <sub>DD</sub>	0.2 × V <sub>DD</sub>	
Output high-level voltage	V <sub>DD</sub> > 2.7V, I <sub>OH</sub> maxII	V <sub>OH</sub>	0.8 × V <sub>DD</sub>	0.9 × V <sub>DD</sub>	-	
Output low-level current	V <sub>DD</sub> = 2.7V-3V, PORT.PINCFG.DRVSTR=0	I <sub>OL</sub>	-	-	1	
	V <sub>DD</sub> = 3V-3.63V, PORT.PINCFG.DRVSTR=0		-	-	2.5	
	V <sub>DD</sub> = 2.7V-3V, PORT.PINCFG.DRVSTR=1		-	-	3	
	V <sub>DD</sub> = 3V-3.63V, PORT.PINCFG.DRVSTR=1		-	-	10	
Output high-level current	V <sub>DD</sub> = 2.7V-3V, PORT.PINCFG.DRVSTR=0	I <sub>OH</sub>	-	-	0.70	

# ATSAMHAXEXXA

## Electrical Characteristics

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
	$V_{DD} = 3V-3.63V$ , PORT.PINCFG.DRVSTR=0		-	-	2	
	$V_{DD} = 2.7V-3V$ , PORT.PINCFG.DRVSTR=1		-	-	2	
	$V_{DD} = 3V-3.63V$ , PORT.PINCFG.DRVSTR=1		-	-	7	
Rise time <sup>(1)</sup>	PORT.PINCFG.DRVSTR = 0load = 5pF, $V_{DD} = 3.3V$	$t_{RISE}$	-	-	15	ns
	PORT.PINCFG.DRVSTR = 1load = 20pF, $V_{DD} = 3.3V$		-	-	15	
Fall time <sup>(1)</sup>	PORT.PINCFG.DRVSTR = 0load = 5pF, $V_{DD} = 3.3V$	$t_{FALL}$	-	-	15	ns
	PORT.PINCFG.DRVSTR = 1load = 20pF, $V_{DD} = 3.3V$		-	-	15	
Input leakage current	Pull-up resistors disabled	$I_{LEAK}$	-1	$\pm 0.015$	1	$\mu A$

**Note:** These values are based on simulation. These values are not covered by test limits in production or characterization.

### 37.10.2 I<sup>2</sup>C Pins

Refer to the *I/O Multiplexing and Considerations* section to get the list of I<sup>2</sup>C pins.

**Table 37-17. I<sup>2</sup>C Pins Characteristics in I<sup>2</sup>C Configuration**

Parameter	Condition	Symbol	Min.	Typ.	Max.	Unit
Pull-up - Pull-down resistance		$R_{PULL}$	20	40	60	k $\Omega$
Input low-level voltage	$V_{DD} = 2.7V-3.63V$	$V_{IL}$	-	-	$0.3 \times V_{DD}$	V
Input high-level voltage	$V_{DD} = 2.7V-3.63V$	$V_{IH}$	$0.55 \times V_{DD}$	-	-	
Hysteresis of Schmitt trigger inputs		$V_{HYS}$	$0.08 \times V_{DD}$	-	-	
Output low-level voltage	$V_{DD} > 2.0V$ , $I_{OL} = 3mA$	$V_{OL}$	-	-	0.4	
	$V_{DD} \leq 2.0V$ , $I_{OL} = 2mA$		-	-	$0.2 \times V_{DD}$	
Capacitance for each I/O Pin		$C_I$				pF
Output low-level current	$V_{OL} = 0.4V$ Standard, Fast and HS Modes	$I_{OL}$	3			mA
	$V_{OL} = 0.4V$ Fast Mode +		20	-	-	

# ATSAMHAXEXXA

## Electrical Characteristics

Parameter	Condition	Symbol	Min.	Typ.	Max.	Unit
	$V_{OL} = 0.6V$		6	-	-	
SCL clock frequency		$f_{SCL}$	-	-	3.4	MHz
Value of pull-up resistor	$f_{SCL} \leq 100kHz$	$R_P$				$\Omega$
	$f_{SCL} > 100kHz$					

I<sup>2</sup>C pins timing characteristics can be found in the *SERCOM in I<sup>2</sup>C Mode Timing* section.

**Table 37-18. I<sup>2</sup>C Pins Characteristics in I/O Configuration**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Pull-up - Pull-down resistance		$R_{PULL}$	20	40	60	k $\Omega$
Input low-level voltage	$V_{DD} = 2.7V-3.63V$	$V_{IL}$	-	-	$0.3 \times V_{DD}$	V
Input high-level voltage	$V_{DD} = 2.7V-3.63V$	$V_{IH}$	$0.55 \times V_{DD}$	-	-	
Output low-level voltage	$V_{DD} > 2.7V$ , IOL max	$V_{OL}$	-	$0.1 \times V_{DD}$	$0.2 \times V_{DD}$	
Output high-level voltage	$V_{DD} > 2.7V$ , IOH max	$V_{OH}$	$0.8 \times V_{DD}$	$0.9 \times V_{DD}$	-	
Output low-level current	$V_{DD} = 2.7V-3V$ , PORT.PINCFG. DRVSTR=0	$I_{OL}$	-	-	1	mA
	$V_{DD} = 3V-3.63V$ , PORT.PINCFG. DRVSTR=0		-	-	2.5	
	$V_{DD} = 2.7V-3V$ , PORT.PINCFG. DRVSTR=1		-	-	3	
	$V_{DD} = 3V-3.63V$ , PORT.PINCFG. DRVSTR=1		-	-	10	
	$V_{DD} = 2.7V-3V$ , PORT.PINCFG. DRVSTR=0		-	-	0.70	
	$V_{DD} = 3V-3.63V$ , PORT.PINCFG. DRVSTR=0		-	-	2	
	$V_{DD} = 2.7V-3V$ , PORT.PINCFG. DRVSTR=1		-	-	2	
Output high-level current	$V_{DD} = 3V-3.63V$ , PORT.PINCFG. DRVSTR=1	$I_{OH}$	-	-		
	$V_{DD} = 2.7V-3V$ , PORT.PINCFG. DRVSTR=0		-	-		
	$V_{DD} = 3V-3.63V$ , PORT.PINCFG. DRVSTR=0		-	-		
	$V_{DD} = 2.7V-3V$ , PORT.PINCFG. DRVSTR=1		-	-		

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
	PORT.PINCFG. DRVSTR=1		-	-	7	
Rise time	load = 20pF, V <sub>DD</sub> = 3.3V	t <sub>RISE</sub>			15	ns
	PORT.PINCFG. DRVSTR=1					
	load = 5pF, V <sub>DD</sub> = 3.3V				15	
	PORT.PINCFG. DRVSTR=0					
Fall time	load = 20pF, V <sub>DD</sub> = 3.3V	t <sub>FALL</sub>			15	ns
	PORT.PINCFG. DRVSTR=1					
	load = 5pF, V <sub>DD</sub> = 3.3V				15	
	PORT.PINCFG. DRVSTR=0					
Input leakage current	Pull-up resistors disabled	I <sub>LEAK</sub>	-1	0.015	1	μA

### 37.10.3 XOSC Pin

XOSC pins behave as normal pins when used as normal I/Os. Refer to table “Normal I/O Pins Characteristics”.

### 37.10.4 XOSC32 Pin

XOSC32 pins behave as normal pins when used as normal I/Os. Refer to table “Normal I/O Pins Characteristics”.

### 37.10.5 External Reset Pin

Reset pin has the same electrical characteristics as normal I/O pins. Refer to table “Normal I/O Pins Characteristics”.

### 37.10.6 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 37-19. Injection Current<sup>(1)</sup>**

Symbol	Description	min	max	Unit
I <sub>inj1</sub> <sup>(2)</sup>	IO pin injection current	-1	+1	mA
I <sub>inj2</sub> <sup>(3)</sup>	IO pin injection current	-15	+15	mA
I <sub>injtotal</sub>	Sum of IO pins injection current	-45	+45	mA

1. Injecting current may have an effect on the accuracy of Analog blocks
2. Conditions for  $V_{pin}$ :  $V_{pin} < GND-0.6V$  or  $3.6V < V_{pin} \leq 4.2V$ .

Conditions for  $V_{DD}$ :  $3V < V_{DD} \leq 3.6V$ .

If  $V_{pin}$  is lower than  $GND-0.6V$ , a current limiting resistor is required. The negative DC injection current limiting resistor  $R$  is calculated as  $R = |(GND-0.6V - V_{pin})/I_{inj1}|$ .

If  $V_{pin}$  is greater than  $V_{DD}+0.6V$ , a current limiting resistor is required. The positive DC injection current limiting resistor  $R$  is calculated as  $R = (V_{pin}-(V_{DD}+0.6))/I_{inj1}$ .

3. Conditions for  $V_{pin}$ :  $V_{pin} < GND-0.6V$  or  $V_{pin} \leq 3.6V$ .

Conditions for  $V_{DD}$ :  $V_{DD} \leq 3V$ .

If  $V_{pin}$  is lower than  $GND-0.6V$ , a current limiting resistor is required. The negative DC injection current limiting resistor  $R$  is calculated as  $R = |(GND-0.6V - V_{pin})/I_{inj2}|$ .

If  $V_{pin}$  is greater than  $V_{DD}+0.6V$ , a current limiting resistor is required. The positive DC injection current limiting resistor  $R$  is calculated as  $R = (V_{pin}-(V_{DD}+0.6))/I_{inj2}$ .

## 37.11 Analog Characteristics

### 37.11.1 Voltage Regulator Characteristics

**Table 37-20. Voltage Regulator Electrical Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
DC calibrated output voltage	Voltage regulator normal mode	$V_{DDCORE}$	1.1	1.23	1.3	V

**Note:** Supplying any external components using  $V_{DDCORE}$  pin is not allowed to assure the integrity of the core supply voltage.

**Table 37-21. Decoupling Requirements**

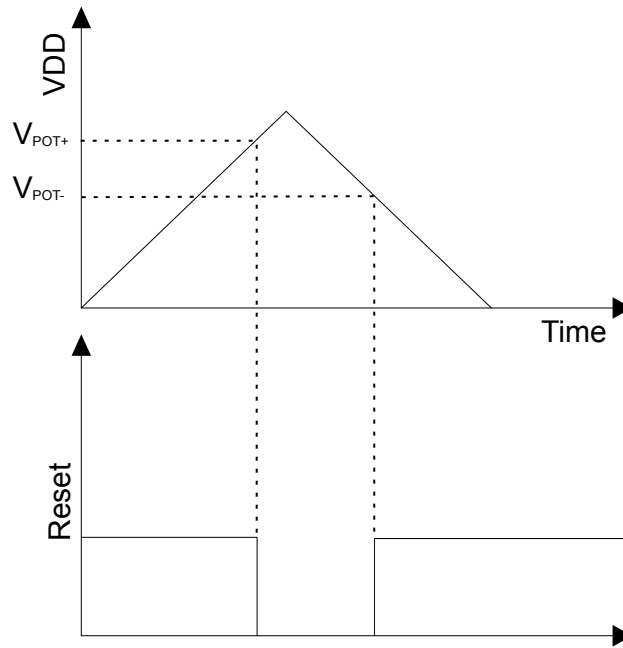
Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Input regulator capacitor, between $V_{DDIN}$ and GND		$C_{IN}$	-	1	-	$\mu F$
Output regulator capacitor, between $V_{DDCORE}$ and GND		$C_{OUT}$	0.8	1	-	$\mu F$

### 37.11.2 Power-On Reset (POR) Characteristics

**Table 37-22. POR Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Voltage threshold level on $V_{DDin}$ rising	$V_{DD}$ falls at 1V/ms or slower	$V_{POT+}$	1.27	1.43	1.59	V
Voltage threshold level on $V_{DDin}$ falling		$V_{POT-}$	0.69	0.97	1.32	V

**Figure 37-2. POR Operating Principle**



### 37.11.3 Brown-Out Detectors Characteristics

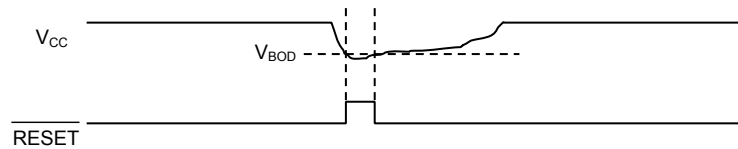
#### 37.11.3.1 BOD33

**Table 37-23. BOD33 LEVEL Value**

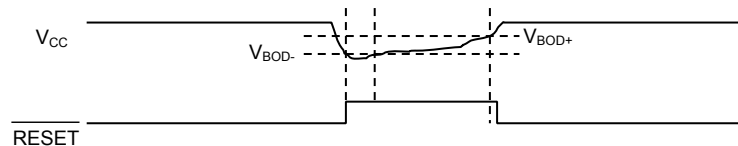
BOD33.LEVEL	Conditions	Symbol	Min.	Typ.	Max.	Unit
39	Hysteresis ON	VBOD+	-	2.77	2.92	V
48			-	3.08	3.3	
39	Hysteresis OFF	VBOD- or VBOD	2.7	2.76	2.81	
48			3	3.07	3.2	

**Note:** Refer to *NVM User Row Mapping* for the BOD33 default value settings.

**Figure 37-3. BOD33 Hysteresis OFF**



**Figure 37-4. BOD33 Hysteresis ON**



**Table 37-24. BOD33 Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Step size, between adjacent values in BOD33.LEVEL			-	34	-	mV
Hysteresis ON	Hysteresis ON	$V_{HYST}$	35	-	170	mV
Detection time	Time with $V_{DDANA} < V_{TH}$ necessary to generate a reset signal	$t_{DET}$	-	0.9 <sup>(1)</sup>	-	$\mu s$
Startup time		$t_{STARTUP}$	-	2.2 <sup>(1)</sup>	-	$\mu s$

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

**Table 37-25. BOD33 Power Consumption**

Parameter	Conditions	$T_j$	$V_{CC}$	Typ.	Max.	Unit
BOD33	IDLE2, Mode CONT	25°C	3.3V	25	48	$\mu A$
		-40 to +105°C	3.3V	-	51	
	STDBY, Mode SAMPL	25°C	3.3V	0.132	0.38	
		-40 to +105°C	3.3V	-	1.5	

### 37.11.4 Analog-to-Digital (ADC) Characteristics

**Table 37-26. Operating Conditions**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Resolution		RES	8	-	12	bits
ADC Clock frequency		$f_{CLK\_ADC}$	30	-	2100	kHz
Sample rate <sup>(1)</sup>	Single shot		5	-	300	ksps
	Free running		5	-	350	ksps
Sampling time <sup>(1)</sup>			0.5	-	-	cycles
Conversion time <sup>(1)</sup>	1x Gain		6	-	-	cycles
Voltage reference range		$V_{REF}$	1.0	-	$V_{DDANA} - 0.6$	V
Internal 1V reference <sup>(2)</sup>		$V_{REFINT} 1V$	-	1	-	V
Internal ratiometric reference 0 <sup>(2)</sup>		$V_{REFINT} V_{CC0}$	-	$V_{DDANA}/1.48$	-	V
Internal ratiometric reference 0 <sup>(2)</sup> error	$2.0V < V_{DDANA} < 3.63V$	$V_{REFINT} V_{CC0}$ Voltage Error	-1	-	1	%

# ATSAMHAXEXXA

## Electrical Characteristics

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Internal ratiometric reference 1 <sup>(2)</sup>	VDDANA > 2.0V	$V_{REFINT} V_{CC1}$	-	$V_{DDANA}/2$	-	V
Internal ratiometric reference 1 <sup>(2)</sup> error	2.0V < VDDANA < 3.63V	$V_{REFINT} V_{CC1}$ Voltage Error	-1	-	1	%
Conversion range <sup>(1)</sup>	Differential mode		$-V_{REF}/GAIN$	-	$+V_{REF}/GAIN$	V
	Single-ended mode		0	-	$+V_{REF}/GAIN$	V
Sampling capacitance <sup>(2)</sup>		$C_{SAMPLE}$	-	3.5	-	pF
Input channel source resistance <sup>(2)</sup>		$R_{SAMPLE}$	-	-	3.5	k $\Omega$
DC supply current <sup>(1)</sup>	$f_{CLK\_ADC} = 2.1MHz^{(3)}$	$I_{DD}$	-	2.9	4.1	mA

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, 1 Conversion at gain 1x takes 6 clock cycles of the ADC clock.

**Table 37-27. Differential Mode**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Effective Number Of Bits	With gain compensation	ENOB	-	10.4	10.8	bits
Total Unadjusted Error	1x Gain	TUE	1.2	7.0	38.0	LSB
Integral Nonlinearity	1x Gain	INL	0.7	1.30	5.6	LSB
Differential Nonlinearity	1x Gain	DNL	-	$\pm 0.7$	$\pm 0.95$	LSB
Gain Error	Ext. Ref 1x		-	$\pm 3$	$\pm 13$	mV
	$V_{REF} = V_{DDANA}/1.48$		-	$\pm 11$	$\pm 55$	mV
	Bandgap		-	$\pm 2$	$\pm 35$	mV
Gain Accuracy <sup>(5)</sup>	Ext. Ref. 0.5x		-	$\pm 0.1$	$\pm 0.8$	%
	Ext. Ref. 2x to 16x		-	$\pm 0.6$	$\pm 0.9$	%
Offset Error	Ext. Ref. 1x		-	$\pm 2$	$\pm 35$	mV
	$V_{REF} = V_{DDANA}/1.48$		-	$\pm 3$	$\pm 40$	mV
	Bandgap		-	$\pm 3$	$\pm 50$	mV
Spurious Free Dynamic Range	1x Gain	SFDR	65	71.5	76	dB
Signal-to-Noise and Distortion	$F_{CLK\_ADC} = 2.1MHz$	SINAD	58	65	67	dB
Signal-to-Noise Ratio	$F_{IN} = 40kHz$	SNR	60	66	68.6	dB



Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Total Harmonic Distortion	$A_{IN} = 95\%FSR$	THD	-75	-71	-67	dB
Noise RMS	$T = 25^{\circ}C$		0.6	1	2.5	mV

- Maximum numbers are based on characterization and not tested in production, and valid for 5% to 95% of the input voltage range.
- Dynamic parameter numbers are based on characterization and not tested in production.
- Respect the input common mode voltage through the following equations (where  $V_{CM\_IN}$  is the Input channel common mode voltage):  
 If  $|V_{IN}| > V_{REF}/4$   
 $V_{CM\_IN} < 0.95 \times V_{DDANA} + V_{REF}/4 - 0.75V$   
 $V_{CM\_IN} > V_{REF}/4 - 0.05 \times V_{DDANA} - 0.1V$   
 If  $|V_{IN}| < V_{REF}/4$   
 $V_{CM\_IN} < 1.2 \times V_{DDANA} - 0.75V$   
 $V_{CM\_IN} > 0.2 \times V_{DDANA} - 0.1V$
- The ADC channels on pins PA08, PA09, PA10, PA11 are powered from the VDDIO power supply. The ADC performance of these pins will not be the same as all the other ADC channels on pins powered from the VDDANA power supply.
- The gain accuracy represents the gain error expressed in percent.  
 Gain accuracy (%) = (Gain Error in V  $\times$  100) / (2  $\times$  Vref/GAIN)

**Table 37-28. Single-Ended Mode**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Effective Number of Bits	With gain compensation	ENOB	-	9.6	10.1	Bits
Total Unadjusted Error	1x gain	TUE	3	11	74	LSB
Integral Nonlinearity	1x gain	INL	1	4	11	LSB
Differential Nonlinearity	1x gain	DNL	-	$\pm 0.5$	$\pm 0.95$	LSB
Gain Error	Ext. Ref. 1x		-	$\pm 0.9$	$\pm 10$	mV
Gain Accuracy <sup>(4)</sup>	Ext. Ref. 0.5x		-	$\pm 0.2$	$\pm 0.5$	%
	Ext. Ref. 2x to 16X		-	$\pm 0.15$	$\pm 0.3$	%
Offset Error	Ext. Ref. 1x		-	$\pm 3$	$\pm 40$	mV
Spurious Free Dynamic Range	1x Gain	SFDR	63	68	70.1	dB
Signal-to-Noise and Distortion	$F_{CLK\_ADC} = 2.1MHz$	SINAD	55	60.1	62.5	dB
Signal-to-Noise Ratio	$F_{IN} = 40kHz$	SNR	54	61	64	dB
Total Harmonic Distortion	$A_{IN} = 95\%FSR$	THD	-70	-68	-65	dB
Noise RMS	$T = 25^{\circ}C$		-	1	5	mV

**Note:**

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 VCM\_IN is the Input channel common mode voltage) for all VIN:  

$$V_{CM\_IN} < 0.7 \times V_{DDANA} + V_{REF}/4 - 0.75V$$

$$V_{CM\_IN} > V_{REF}/4 - 0.3 \times V_{DDANA} - 0.1V$$
3. The ADC channels on pins PA08, PA09, PA10, PA11 are powered from the VDDIO power supply. The ADC performance of these pins will not be the same as all the other ADC channels on pins powered from the VDDANA power supply.
4. The gain accuracy represents the gain error expressed in percent.  

$$\text{Gain accuracy (\%)} = (\text{Gain Error in V} \times 100) / (V_{ref}/GAIN)$$

### 37.11.4.1 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 37-29. 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 at 25°C	66.0	65.0	72.8	10.5
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

### 37.11.4.2 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 37-30. Offset and Gain Correction Feature**

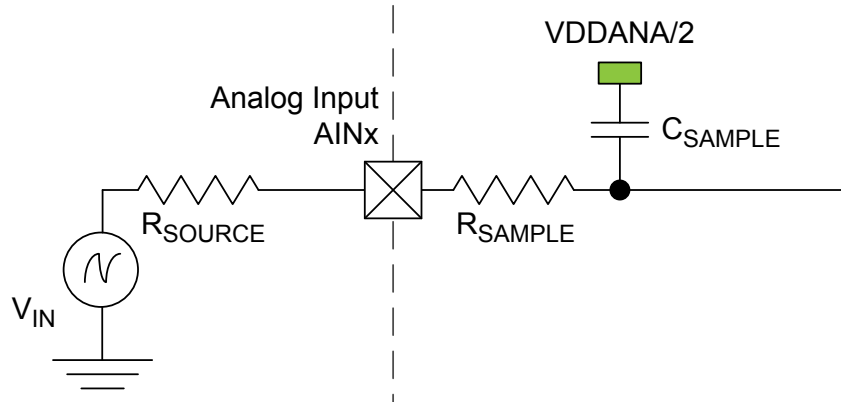
Gain Factor	Conditions	Offset Error (mV)	Gain Error (mV)	Total Unadjusted Error (LSB)
0.5x	In differential mode, 1x gain, $V_{DDANA} = 3.0V$ , $V_{REF} = 1.0V$ , 350kSps at 25°C	0.25	1.0	2.4
1x		0.20	0.10	1.5
2x		0.15	-0.15	2.7
8x		-0.05	0.05	3.2
16x		0.10	-0.05	6.1

### 37.11.4.3 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 ( $C_{\text{SAMPLE}}$ ). In addition, the source resistance ( $R_{\text{SOURCE}}$ ) must be taken into account when calculating the required sample and hold time. The next figure shows the ADC input channel equivalent circuit.

**Figure 37-5. ADC Input**



To achieve  $n$  bits of accuracy, the  $C_{\text{SAMPLE}}$  capacitor must be charged at least to a voltage of

$$V_{\text{CSAMPLE}} \geq V_{\text{IN}} \times (1 + 2^{-(n+1)})$$

The minimum sampling time  $t_{\text{SAMPLEHOLD}}$  for a given  $R_{\text{SOURCE}}$  can be found using this formula:

$$t_{\text{SAMPLEHOLD}} \geq (R_{\text{SAMPLE}} + R_{\text{SOURCE}}) \times (C_{\text{SAMPLE}}) \times (n + 1) \times \ln(2)$$

for a 12 bits accuracy:  $t_{\text{SAMPLEHOLD}} \geq (R_{\text{SAMPLE}} + R_{\text{SOURCE}}) \times (C_{\text{SAMPLE}}) \times 9.02$

where

$$t_{\text{SAMPLEHOLD}} = \frac{1}{2 \times f_{\text{ADC}}}$$

### 37.11.5 Digital-to-Analog Converter (DAC) Characteristics

**Table 37-31. Operating Conditions<sup>(1)</sup>**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
$V_{\text{DDANA}}$	Analog supply voltage		2.7	-	3.63	V
$AV_{\text{REF}}$	External reference voltage		1.0	-	$V_{\text{DDANA}} - 0.6$	V
	Internal reference voltage 1		-	1	-	V
	Internal reference voltage 2		-	$V_{\text{DDANA}}$	-	V
	Linear output voltage range		0.05	-	$V_{\text{DDANA}} - 0.05$	V
	Minimum resistive load		5	-	-	k $\Omega$
	Maximum capacitance load		-	-	100	pF
$I_{\text{DD}}$	DC supply current <sup>(2)</sup>	Voltage pump disabled	-	175	256	$\mu\text{A}$

1. These values are based on specifications otherwise noted.
2. These values are based on characterization. These values are not covered by test limits in production.

**Table 37-32. Clock and Timing<sup>(1)</sup>**

Parameter	Conditions	Min.	Typ.	Max.	Unit	
Conversion rate	C <sub>load</sub> = 100pF R <sub>load</sub> > 5kΩ	Normal mode	-	-	350	ksps
		For Δ <sub>DATA</sub> = ±1	-	-	1000	
Startup time	V <sub>DDNA</sub> > 2.6V	-	-	2.85	μs	
	V <sub>DDNA</sub> < 2.6V	-	-	10	μs	

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

**Table 37-33. Accuracy Characteristics<sup>(1)</sup>**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit			
RES	Input resolution		-	-	10	Bits			
INL	Integral nonlinearity	V <sub>REF</sub> = Ext 1.0V	V <sub>DD</sub> = 1.6V	±0.2	±0.5	±1	LSB		
			V <sub>DD</sub> = 3.6V	±0.2	±0.4	±1.2			
		V <sub>REF</sub> = V <sub>DDANA</sub>	V <sub>DD</sub> = 1.6V	±0.2	±0.6	±1.2			
			V <sub>DD</sub> = 3.6V	±0.2	±0.5	±1.3			
		V <sub>REF</sub> = INT1V	V <sub>DD</sub> = 1.6V	±0.4	±0.7	±2			
			V <sub>DD</sub> = 3.6V	±0.4	±0.8	±6			
DNL	Differential nonlinearity	V <sub>REF</sub> = Ext 1.0V	V <sub>DD</sub> = 1.6V	±0.1	±0.3	±0.8	LSB		
			V <sub>DD</sub> = 3.6V	±0.1	±0.3	±0.8			
		V <sub>REF</sub> = V <sub>DDANA</sub>	V <sub>DD</sub> = 1.6V	±0.1	±0.2	±0.5			
			V <sub>DD</sub> = 3.6V	±0.1	±0.2	±1			
		V <sub>REF</sub> = INT1V	V <sub>DD</sub> = 1.6V	±0.3	±0.6	±3			
			V <sub>DD</sub> = 3.6V	±0.3	±0.8	±7			
		Gain error		V <sub>REF</sub> = Ext. V <sub>REF</sub>	-	±4		±16	mV
				V <sub>REF</sub> = V <sub>DDANA</sub>	-	±12		±60	mV
V <sub>REF</sub> = INT1V	-			±1	±22	mV			
Offset error		V <sub>REF</sub> = Ext. V <sub>REF</sub>	-	±1	±13	mV			
		V <sub>REF</sub> = V <sub>DDANA</sub>	-	±2.5	±21	mV			
		V <sub>REF</sub> = INT1V	-	±1.5	±20	mV			

1. All values measured using a conversion rate of 350ksps.

### 37.11.6 Analog Comparator Characteristics

**Table 37-34. Electrical and Timing**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Positive input voltage range			0	-	$V_{DDANA}$	V
Negative input voltage range			0	-	$V_{DDANA}$	
Offset	Hysteresis = 0, Fast mode		-26	0	26	mV
	Hysteresis = 0, Low power mode		-28	0	28	mV
Hysteresis	Hysteresis = 1, Fast mode		8	50	102	mV
	Hysteresis = 1, Low power mode		14	50	75	mV
Propagation delay	Changes for $V_{ACM} = V_{DDANA}/2$ 100mV overdrive, Fast mode			90	180	ns
	Changes for $V_{ACM} = V_{DDANA}/2$ 100mV overdrive, Low power mode			302	534	ns
Startup time	Enable to ready delay Fast mode	$t_{STARTUP}$		1	2	$\mu$ s
	Enable to ready delay Low power mode			-	14	23
INL <sup>(3)</sup>		$V_{SCALE}$	-1.4	0.201	1.4	LSB
DNL <sup>(3)</sup>			-0.9	0.022	0.9	LSB
Offset Error <sup>(1)(2)</sup>			-0.2	0.056	0.92	LSB
Gain Error <sup>(1)(2)</sup>			-0.89	0.079	0.89	LSB

1. According to the standard equation  $V(X) = V_{LSB} \times (X + 1)$ ;  $V_{LSB} = V_{DDANA}/64$
2. Data computed with the Best Fit method
3. Data computed using histogram

### 37.11.7 Internal 1.1V Bandgap Reference Characteristics

**Table 37-35. Bandgap and Internal 1.1V Reference Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Internal 1.1V Bandgap reference	After calibration at $T = 25^{\circ}\text{C}$ , over $[-40,$ $+105]^{\circ}\text{C}$ , $V_{DD} = 3.3\text{V}$	INT1V	1.07	1.1	1.12	V
	Over voltage at $25^{\circ}\text{C}$		1.08	1.1	1.11	V

## 37.12 NVM Characteristics

**Table 37-36. Maximum Operating Frequency**

V <sub>DD</sub> range	NVM Wait States	Maximum Operating Frequency	Unit
2.7V to 3.63V	0	24	MHz
	1	48	MHz

Note that on this flash technology, a maximum number of 8 consecutive writes are allowed per row. Once this number is reached, a row erase is mandatory.

**Table 37-37. Flash Endurance and Data Retention**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Retention after up to 25k	Average case 55°C	Ret <sub>NVM25k</sub>	10	50	-	Years
Retention after up to 2.5k	Average case 55°C	Ret <sub>NVM2.5k</sub>	20	100	-	Years
Retention after up to 100	Average case 55°C	Ret <sub>NVM100</sub>	25	>100	-	Years
Cycling Endurance(1)	-40°C < T <sub>C</sub> < 105°C	Cyc <sub>NVM</sub>	25k	150k	-	Cycles

An endurance cycle is a write and an erase operation.

**Table 37-38. EEPROM Emulation(1) Endurance and Data Retention**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Retention after up to 100k	Average case 55°C	Ret <sub>EEPROM100k</sub>	10	50	-	Years
Retention after up to 10k	Average case 55°C	Ret <sub>EEPROM10k</sub>	20	100	-	Years
Cycling Endurance(2)	-40°C < T <sub>C</sub> < 105°C	Cyc <sub>EEPROM</sub>	100k	600k	-	Cycles

The EEPROM emulation is a software emulation described in the App note AT03265. An endurance cycle is a write and an erase operation.

**Table 37-39. NVM Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Page programming time	-	t <sub>FPP</sub>	-	-	2.5	ms
Row erase time	-	t <sub>FRE</sub>	-	-	6	ms
DSU chip erase time (CHIP_ERASE)	-	t <sub>FCE</sub>	-	-	240	ms

## 37.13 Oscillators Characteristics

All temperature values are T<sub>C</sub> unless otherwise stated.

### 37.13.1 Crystal Oscillator (XOSC) Characteristics

#### 37.13.1.1 Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN.

**Table 37-40. Digital Clock Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
XIN clock frequency	Digital mode	$F_{Xin}$	-	-	32	MHz
XIN clock duty cycle	Digital mode	$DC_{xin}$	-	-	-	%

### 37.13.1.2 XOSC Characteristics

The following table describes the characteristics for the oscillator when a crystal is connected between XIN and XOUT. 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 37-41. Crystal Oscillator Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Crystal oscillator frequency		$f_{OUT}$	0.4	-	32	MHz
Crystal Equivalent Series Resistance  Safety Factor = 3 The AGC does not have any noticeable impact on these measurements.	f = 0.455 MHz, $C_L = 100pF$ XOSC.GAIN = 0	ESR	-	-	5.6K	$\Omega$
	f = 2MHz, $C_L = 20pF$ XOSC.GAIN = 0		-	-	330	
	f = 4MHz, $C_L = 20pF$ XOSC.GAIN = 1		-	-	240	
	f = 8 MHz, $C_L = 20pF$ XOSC.GAIN = 2		-	-	105	
	f = 16 MHz, $C_L = 20pF$ XOSC.GAIN = 3		-	-	60	
	f = 32MHz, $C_L = 18pF$ XOSC.GAIN = 4		-	-	55	

# ATSAMHAXEXXA

## Electrical Characteristics

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Parasitic capacitor load		$C_{XIN}$	-	5.9	-	pF
Parasitic capacitor load		$C_{XOUT}$	-	3.2	-	pF

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Startup time	f = 2MHz, $C_L = 20\text{pF}$ , XOSC.GAIN = 0, ESR = 600Ω	$t_{STARTUP}$	-	15.6K	51.0K	cycles
	f = 4MHz, $C_L = 20\text{pF}$ , XOSC.GAIN = 1, ESR = 100Ω		-	6.3K	20.1K	
	f = 8 MHz, $C_L = 20\text{pF}$ , XOSC.GAIN = 2, ESR = 35Ω		-	6.2K	20.3K	
	f = 16 MHz, $C_L = 20\text{pF}$ , XOSC.GAIN = 3, ESR = 25Ω		-	7.7K	21.2K	
	f = 32MHz, $C_L = 18\text{pF}$ , XOSC.GAIN = 4, ESR = 40Ω		-	6.0K	14.2K	

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Current Consumption	f = 2MHz, $C_L = 20\text{pF}$ , XOSC.GAIN = 0, AGC off		-	89	190	μA
	f = 2MHz,		-	82	187	



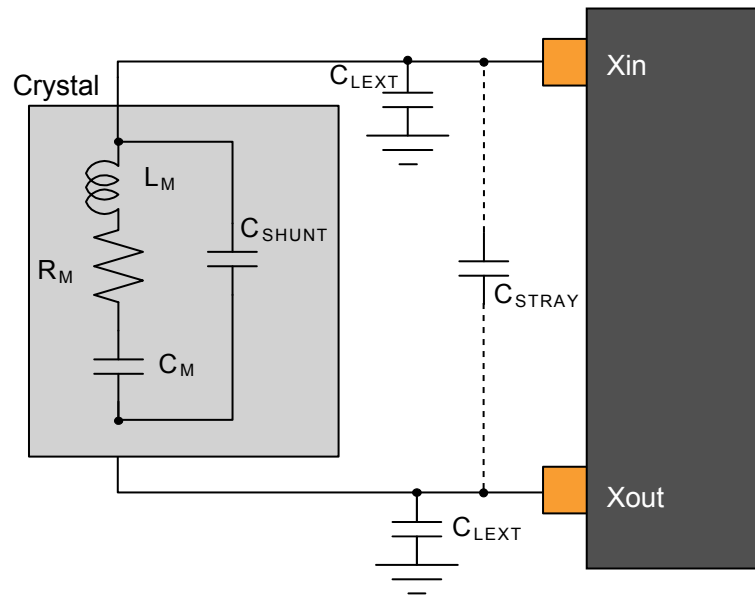
# ATSAMHAXEXXA

## Electrical Characteristics

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
	$C_L = 20\text{pF}$ , XOSC.GAIN = 0, AGC on					
	$f = 4\text{MHz}$ , $C_L = 20\text{pF}$ , XOSC.GAIN = 1, AGC off		-	140	256	
	$f = 4\text{MHz}$ , $C_L = 20\text{pF}$ , XOSC.GAIN = 1, AGC on		-	102	219	
	$f = 8\text{MHz}$ , $C_L = 20\text{pF}$ , XOSC.GAIN = 2, AGC off		-	243	380	
Current Consumption	$f = 8\text{MHz}$ , $C_L = 20\text{pF}$ , XOSC.GAIN = 2, AGC on		-	166	299	$\mu\text{A}$
	$f = 16\text{MHz}$ , $C_L = 20\text{pF}$ , XOSC.GAIN = 3, AGC off		-	493	685	
	$f = 16\text{MHz}$ , $C_L = 20\text{pF}$ , XOSC.GAIN = 3, AGC on		-	293	480	
	$f = 32\text{MHz}$ , $C_L = 18\text{pF}$ , XOSC.GAIN = 4,		-	1343	1975	

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
	AGC off					
	f = 32MHz, C <sub>L</sub> = 18pF, XOSC.GAIN = 4, AGC on		-	555	776	

**Figure 37-6. Oscillator Connection**



### 37.13.2 External 32 kHz Crystal Oscillator (XOSC32K) Characteristics

#### 37.13.2.1 Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN32 pin.

**Table 37-42. Digital Clock Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
XIN32 clock frequency		f <sub>CPXIN32</sub>	-	32.768	-	kHz
XIN32 clock duty cycle		DCxin	-	50	-	%

#### 37.13.2.2 XOSC32K Characteristics

Figure 37-6 and the equation in [XOSC Characteristics](#) also apply to the 32 kHz oscillator connection. The user must choose a crystal oscillator where the crystal load capacitance C<sub>L</sub> is within the range given in the table. The exact value of C<sub>L</sub> can be found in the crystal datasheet.

**Table 37-43. 32kHz Crystal Oscillator Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Crystal oscillator frequency		f <sub>OUT</sub>	-	32768	-	Hz
Startup time	ESR <sub>XTAL</sub> = 39.9 kΩ, C <sub>L</sub> = 12.5 pF	t <sub>STARTUP</sub>	-	28K	30K	cycles
Crystal load capacitance		C <sub>L</sub>	-	-	12.5	pF
Crystal shunt capacitance		C <sub>SHUNT</sub>	-	0.1	-	pF
Parasitic capacitor load		C <sub>XIN32</sub>	-	3.2	-	pF
Parasitic capacitor load		C <sub>XOUT32</sub>	-	3.7	-	pF
Current consumption		I <sub>XOSC32K</sub>	-	1.22	2.2	μA
Crystal equivalent series resistance f = 32.768kHz Safety Factor = 3	C <sub>L</sub> =12.5pF	ESR	-	-	100	kΩ

### 37.13.3 Digital Frequency Locked Loop (DFLL48M) Characteristics

**Table 37-44. DFLL48M Characteristics - Open Loop Mode**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Output frequency	DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512 over [-10, +105]C, over [2.7, 3.6]V	f <sub>OUT</sub>	44.75	48	49	MHz
Output frequency	DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512 over [-40, +105]C, over [2.7, 3.6]V	f <sub>OUT</sub>	43.75	48	49	MHz
Output frequency	DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512 at 25°C, over [2.7, 3.6]V	f <sub>OUT</sub>	45.5	48	49	MHz
Power consumption on V <sub>DDIN</sub>	DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512	I <sub>DFLL</sub>	-	403	453	μA
Startup time	DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512 f <sub>OUT</sub> within 90 % of final value	t <sub>STARTUP</sub>	-	8.6	11.5	μs

**Table 37-45. DFLL48M Characteristics - Closed Loop Mode<sup>(1)</sup>**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Average Output frequency	f <sub>REF</sub> = XTAL, 32.768kHz, 100ppm DFLLMUL = 1464	f <sub>CloseOUT</sub>	47.963	47.972	47.981	MHz
Reference frequency		f <sub>REF</sub>	0.732	32.768	33	kHz
Cycle to Cycle jitter	f <sub>REF</sub> = XTAL, 32.768kHz, 100ppm DFLLMUL = 1464	Jitter	-	-	0.42	ns
Power consumption on V <sub>DDIN</sub>	f <sub>REF</sub> = XTAL, 32.768kHz, 100ppm	I <sub>DFLL</sub>	-	403	453	μA
Lock time	f <sub>REF</sub> = XTAL, 32.768kHz, 100ppm DFFLMUL = 1464 DFLLVAL.COARSE = DFLL48M COARSE CAL DFLLVAL.FINE = 512 DFLLCTRL.BPLCKC = 1 DFLLCTRL.QLDIS = 0 DFLLCTRL.CCDIS = 1 DFLLMUL.FSTEP = 10	t <sub>LOCK</sub>	-	350	1500	μs

**Note:**

- To ensure 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.

### 37.13.4 Ultra Low Power Internal 32kHz RC Oscillator (OSCULP32K) Characteristics

**Table 37-46. Ultra Low Power Internal 32kHz RC Oscillator Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Output frequency	All temperatures T <sub>C</sub> Calibrated against a 32.768kHz reference at 25°C, over [-40, +105]°C, over [2.7, 3.63]V	f <sub>OUT</sub>	24.576	32.768	40.960	kHz
	Calibrated against a 32.768kHz reference at 25°C, at VDD = 3.3V		31.457	32.768	34.078	

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
	Calibrated against a 32.768kHz reference at 25°C, over [2.7, 3.63]V		31.293	32.768	34.570	
Duty Cycle		Duty	-	50	-	%

1. These values are based on simulation. These values are not covered by test limits in production or characterization.
2. This oscillator is always on.

### 37.13.5 32.768kHz Internal oscillator (OSC32K) Characteristics

**Table 37-47. 32kHz RC Oscillator Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Output frequency	All temperatures $T_C$ Calibrated against a 32.768kHz reference at 25°C, over [-40, +105]C, over [2.7, 3.63]V	$f_{OUT}$	26.214	32.768	39.321	kHz
	Calibrated against a 32.768kHz reference at 25°C, at $V_{DD} = 3.3V$		32.113	32.768	33.423	
	Calibrated against a 32.768kHz reference at 25°C, over [2.7, 3.63]V		31.457	32.768	34.079	
Current consumption		$I_{OSC32K}$		0.67	4.06	$\mu A$
Startup time		$t_{STARTUP}$		1	2	cycle
Duty Cycle		Duty		50		%

### 37.13.6 8MHz RC Oscillator (OSC8M) Characteristics

**Table 37-48. Internal 8MHz RC Oscillator Characteristics**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Output frequency	Calibrated against a 8MHz reference at 25°C, over [-10, +70]C,	$f_{OUT}$	7.84	8	8.16	MHz

# ATSAMHAXEXXA

## Electrical Characteristics

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
	over [2.7, 3.6]V		7.80	8	8.20	
	Calibrated against a 8MHz reference at 25°C, over [-10, +105]°C, over [2.7, 3.6]V					
	Calibrated against a 8MHz reference at 25°C, over [-40, +105]°C, over [2.7, 3.6]V					
	Calibrated against a 8MHz reference at 25°C, over [2.7, 3.6]V					
Current consumption	IDLE2 on OSC32K versus IDLE2 on calibrated OSC8M enabled at 8MHz (FRANGE=1, PRESC=0)	I <sub>OSC8M</sub>	-	64	96	μA
Startup time		t <sub>STARTUP</sub>	-	2.3	3.9	μs
Duty cycle		Duty	-	50	-	%

### 37.13.7 Fractional Digital Phase-Locked Loop (FDPLL96M) Characteristics

**Table 37-49. FDPLL96M Characteristics<sup>(1)</sup>**

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
Input frequency		f <sub>IN</sub>	32	-	2000	KHz
Output frequency		f <sub>OUT</sub>	48	-	96	MHz
Current consumption	f <sub>IN</sub> = 32kHz, f <sub>OUT</sub> = 48MHz	I <sub>FDPLL96M</sub>	-	500	-	μA
	f <sub>IN</sub> = 32kHz, f <sub>OUT</sub> = 96MHz		-	900	-	
Period jitter	f <sub>IN</sub> = 32kHz, f <sub>OUT</sub> = 48MHz	Jp	-	2.1	4.0	%
	f <sub>IN</sub> = 32kHz, f <sub>OUT</sub> = 96MHz		-	4.0	11.0	
	f <sub>IN</sub> = 2MHz, f <sub>OUT</sub> = 48MHz		-	2.2	4.0	
	f <sub>IN</sub> = 2MHz, f <sub>OUT</sub> = 96MHz		-	4.7	12.0	
Lock Time	After startup, time to get lock signal. f <sub>IN</sub> = 32kHz, f <sub>OUT</sub> = 96MHz	t <sub>LOCK</sub>	-	1.2	2	ms

Parameter	Conditions	Symbol	Min.	Typ.	Max.	Unit
	$f_{IN} = 2\text{MHz}$ , $f_{OUT} = 96\text{MHz}$		-	25	35	$\mu\text{s}$
Duty cycle		Duty	40	50	60	%

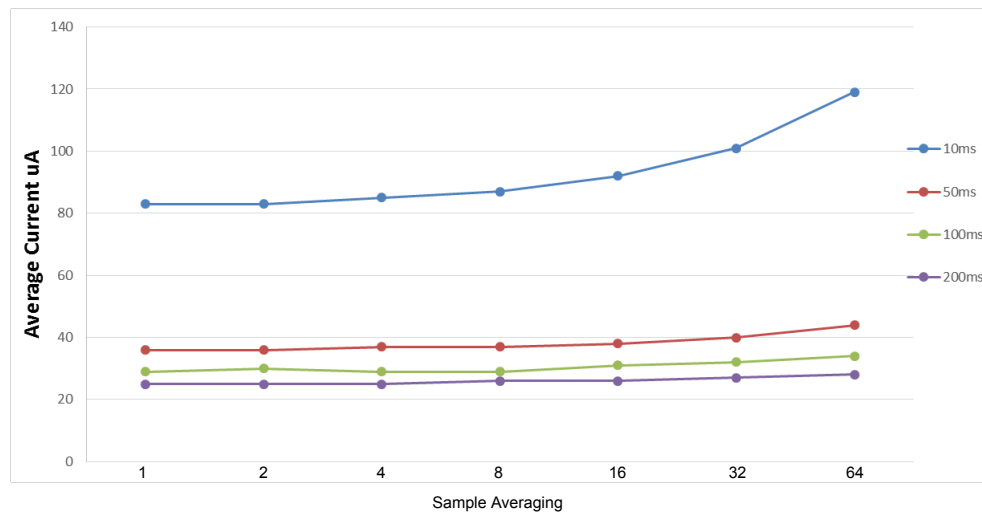
- All values have been characterized with `FILTSEL[1/0]` as default value.

### 37.14 PTC Typical Characteristics

#### 37.14.1

$V_{CC} = 3.3\text{C}$  and  $f_{CPU} = 48\text{MHz}$  for the following PTC measurements.

**Figure 37-7. 1 Sensor / PTC\_GCLK = 4MHz / FREQ\_MODE\_NONE**



**Figure 37-8. 1 Sensor / PTC\_GCLK = 2MHz / FREQ\_MODE\_HOP**

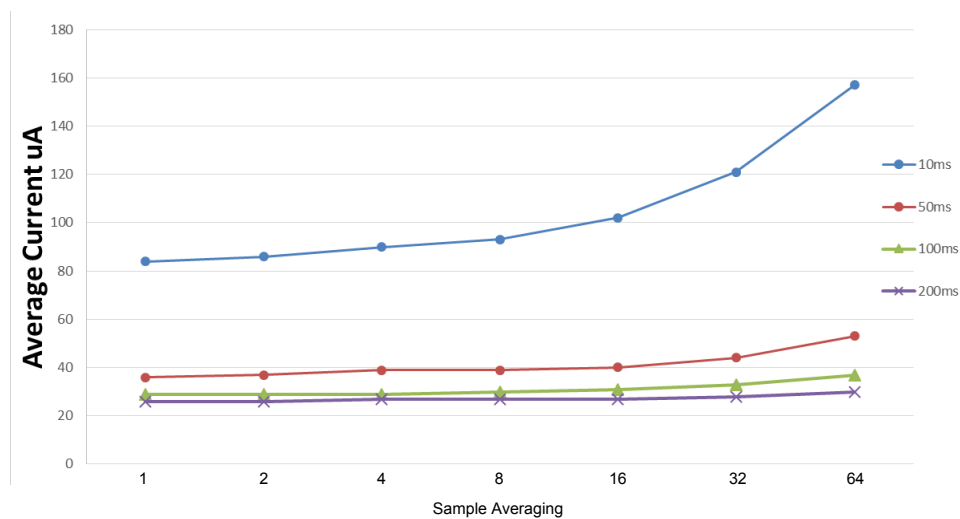


Figure 37-9. 10 Sensor / PTC\_GCLK = 4MHz / FREQ\_MODE\_NONE

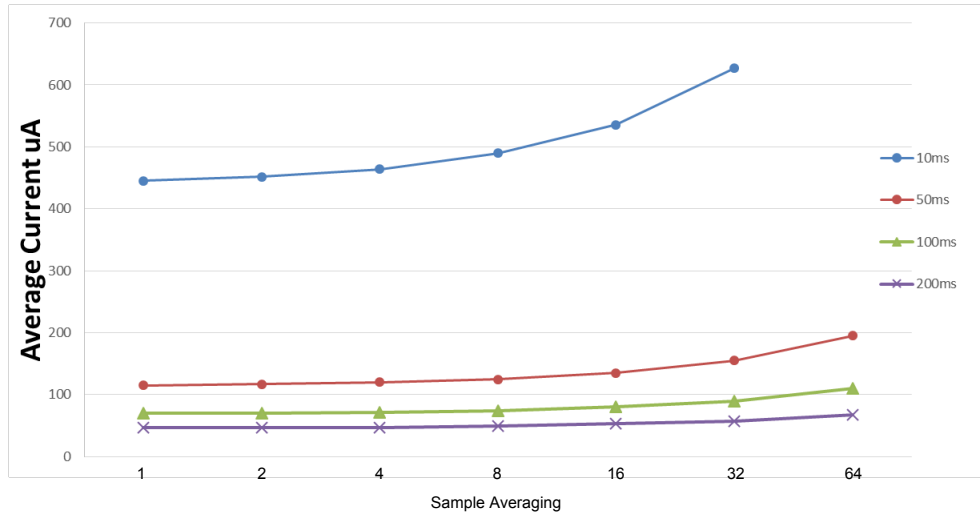


Figure 37-10. 10 Sensor / PTC\_GCLK = 2MHz / FREQ\_MODE\_HOP

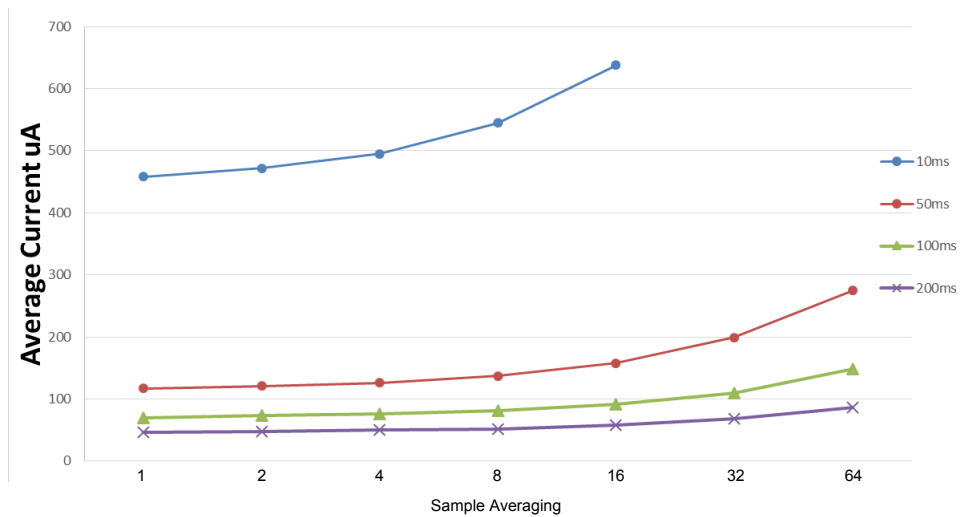




Figure 37-11. 100 Sensor / PTC\_GCLK = 4MHz / FREQ\_MODE\_NONE

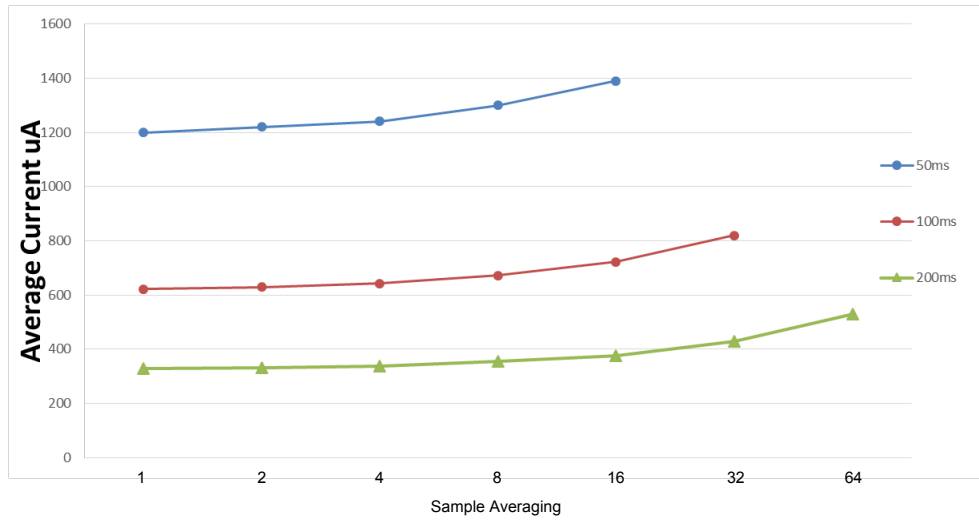
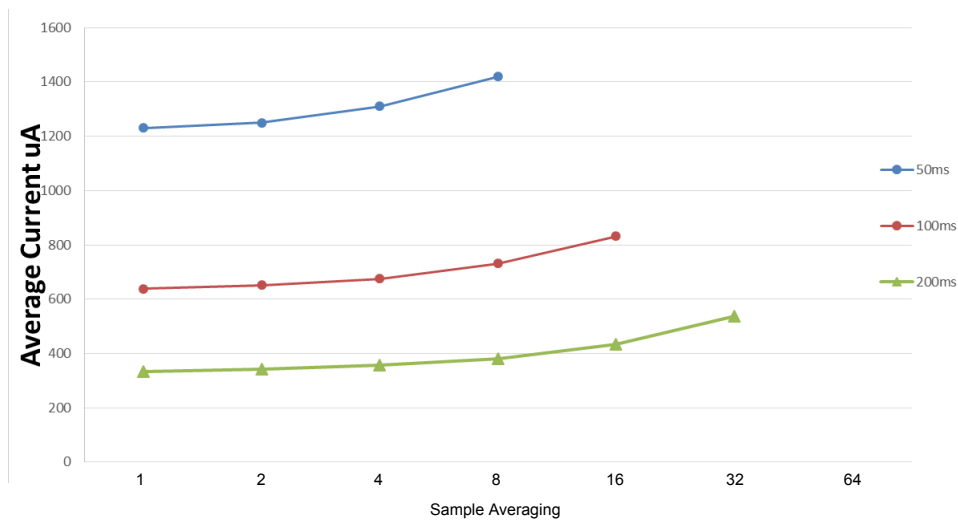


Figure 37-12. 100 Sensor / PTC\_GCLK = 2MHz / FREQ\_MODE\_HOP



## 37.15 Timing Characteristics

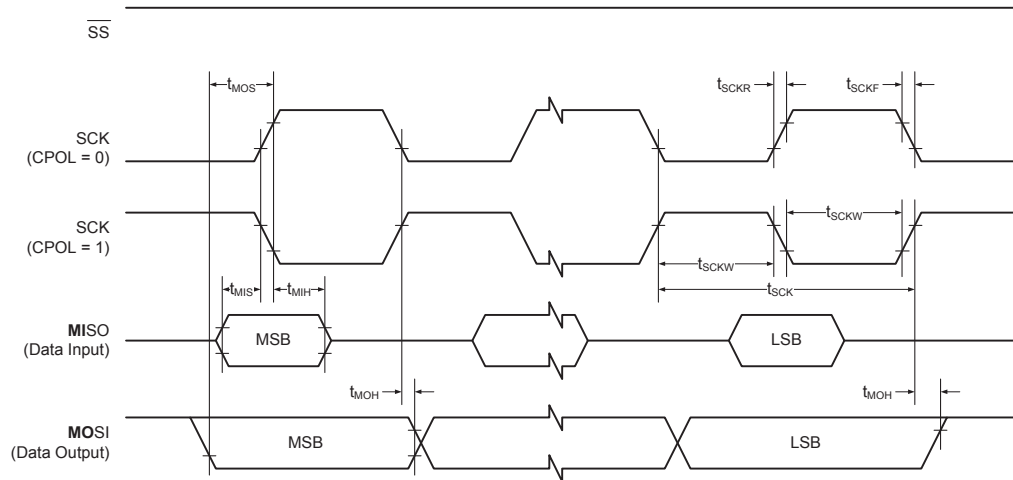
### 37.15.1 External Reset

Table 37-50. External Reset Characteristics

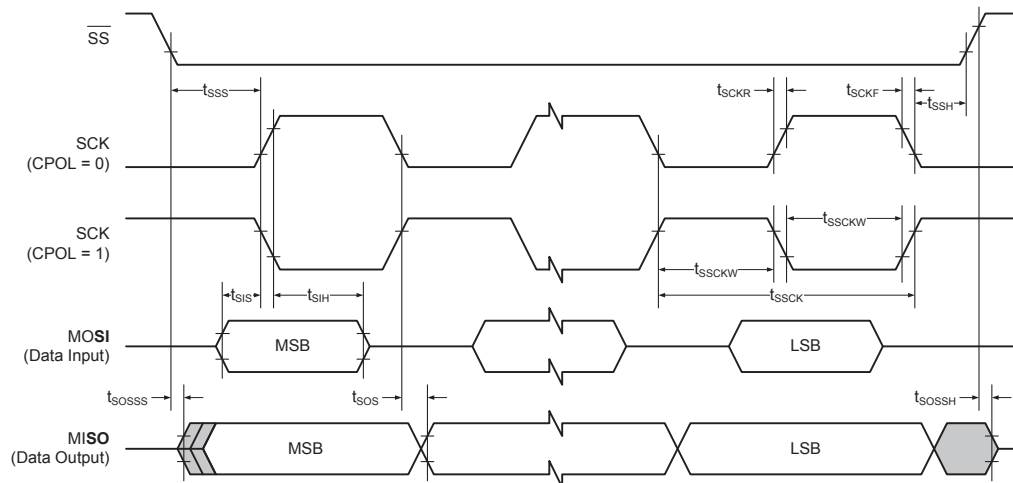
Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
$t_{EXT}$	Minimum reset pulse width		10	-	-	ns

### 37.15.2 SERCOM in SPI Mode Timing

**Figure 37-13. SPI Timing Requirements in Master Mode**



**Figure 37-14. SPI Timing Requirements in Slave Mode**



**Table 37-51. SPI Timing Characteristics and Requirements<sup>(1)</sup>**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
$t_{SCK}$	SCK period	Master		84		ns
$t_{SCKW}$	SCK high/low width	Master	-	$0.5 \cdot t_{SCK}$	-	
$t_{SCKR}$	SCK rise time <sup>(2)</sup>	Master	-	-	-	
$t_{SCKF}$	SCK fall time <sup>(2)</sup>	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	$1 \cdot t_{CLK\_APB}$	-	-	

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
$t_{SSCKW}$	SCK high/low width	Slave	$0.5 \cdot t_{SSCK}$	-	-	
$t_{SSCKR}$	SCK rise time <sup>(2)</sup>	Slave	-	-	-	
$t_{SSCKF}$	SCK fall time <sup>(2)</sup>	Slave	-	-	-	
$t_{SIS}$	MOSI setup to SCK	Slave	$t_{SSCK}/2 - 9$	-	-	
$t_{SIH}$	MOSI hold after SCK	Slave	$t_{SSCK}/2 - 3$	-	-	
$t_{SSS}$	SS setup to SCK	Slave PRELOADEN=1	$2 \cdot t_{CLK\_APB} + t_{SOS}$	-	-	
		Slave 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 SS low	Slave	-	18	-	
$t_{SOSH}$	MISO hold after SS high	Slave	-	10	-	

1. These values are based on simulation. These values are not covered by test limits in production.
2. See I/O Pin characteristics

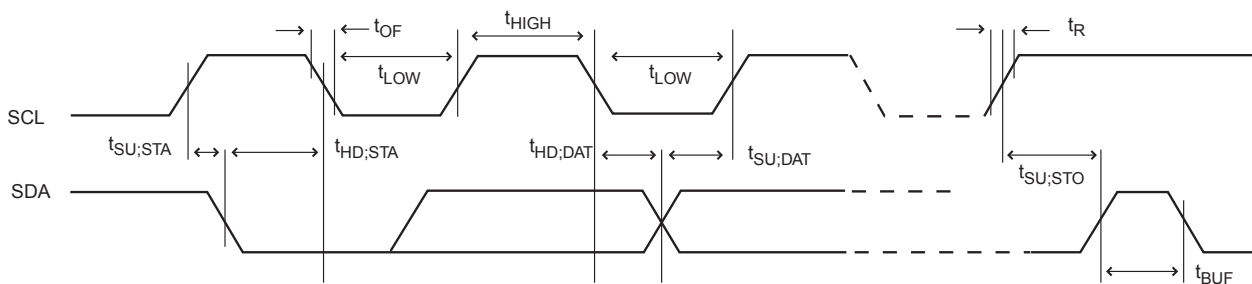
### Related Links

[HA1\\_I/O Pin Characteristics](#)

### 37.15.3 SERCOM in I<sup>2</sup>C Mode Timing

This section describes the requirements for devices connected to the I<sup>2</sup>C Interface Bus.

**Figure 37-15. I<sup>2</sup>C Interface Bus Timing**



**Table 37-52. I<sup>2</sup>C Interface Timing**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units	
$t_R$	Rise time for both SDA and SCL	Standard / Fast Mode	$C_b^{(2)} = 400\text{pF}$	-	230	350	ns
		Fast Mode +	$C_b^{(2)} = 550\text{pF}$	-	60	100	

# ATSAMHAXEXXA

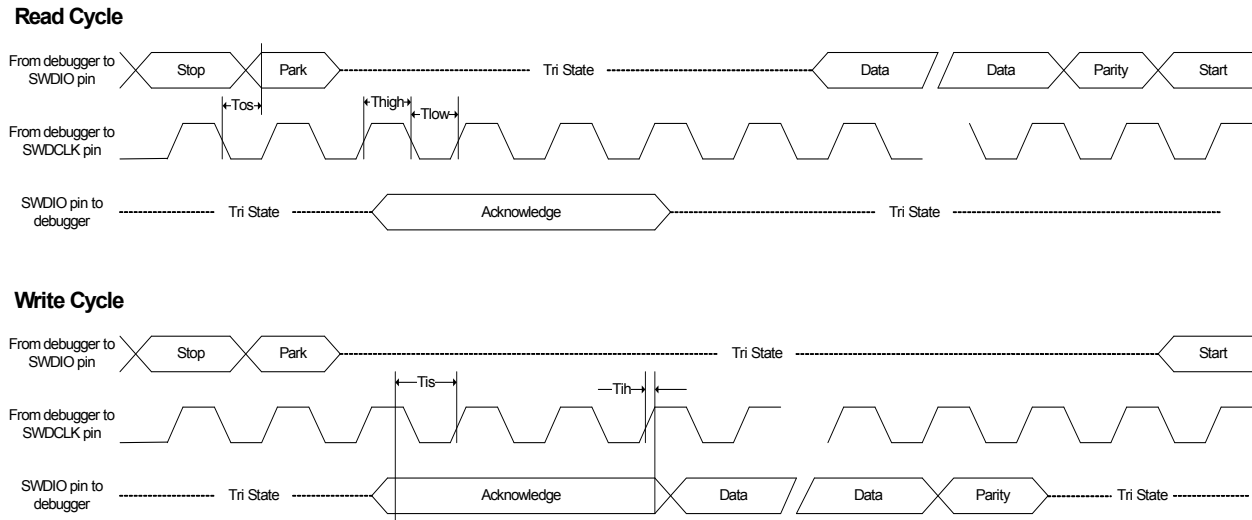
## Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units	
		High Speed Mode	$C_b^{(2)} = 100\text{pF}$		30	60	
$t_{\text{OF}}$	Output fall time from $V_{\text{IHmin}}$ to $V_{\text{ILmax}}$	Standard / Fast Mode	$10\text{pF} < C_b^{(2)} < 400\text{pF}$		25	50	
		Fast Mode +	$10\text{pF} < C_b^{(2)} < 550\text{pF}$		20	30	
		High Speed Mode	$10\text{pF} < C_b^{(2)} < 100\text{pF}$		10	20	
$t_{\text{HD;STA}}$	Hold time (repeated) START condition		$f_{\text{SCL}} > 100\text{kHz}$ , Master	$t_{\text{LOW}-9}$	-	-	
$t_{\text{LOW}}$	Low period of SCL Clock		$f_{\text{SCL}} > 100\text{kHz}$	113	-	-	
$t_{\text{BUF}}$	Bus free time between a STOP and a START condition		$f_{\text{SCL}} > 100\text{kHz}$	$t_{\text{LOW}}$	-	-	
$t_{\text{SU;STA}}$	Setup time for a repeated START condition		$f_{\text{SCL}} > 100\text{kHz}$ , Master	$t_{\text{LOW}+7}$	-	-	
$t_{\text{HD;DAT}}$	Data hold time		$f_{\text{SCL}} > 100\text{kHz}$ , Master	9	-	12	
$t_{\text{SU;DAT}}$	Data setup time		$f_{\text{SCL}} > 100\text{kHz}$ , Master	104	-	-	
$t_{\text{SU;STO}}$	Setup time for STOP condition		$f_{\text{SCL}} > 100\text{kHz}$ , Master	$t_{\text{LOW}+9}$	-	-	
$t_{\text{SU;DAT;rx}}$	Data setup time (receive mode)		$f_{\text{SCL}} > 100\text{kHz}$ , Slave	51	-	56	
$t_{\text{HD;DAT;tx}}$	Data hold time (send mode)		$f_{\text{SCL}} > 100\text{kHz}$ , Slave	71	90	138	

1. These values are based on simulation. These values are not covered by test limits in production.
2.  $C_b$  = Capacitive load on each bus line. Otherwise noted, value of  $C_b$  set to 20pF.

### 37.15.4 SWD Timing

**Figure 37-16. SWD Interface Signals**



**Table 37-53. SWD Timings<sup>(1)</sup>**

Symbol	Parameter	Conditions	Min.	Max.	Units
$T_{high}$	SWDCLK High period	$V_{VDDIO}$ from 3.0 V to 3.6 V, maximum external capacitor = 40 pF	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.

## 38. Schematic Checklist

### 38.1 Introduction

This chapter describes a common checklist which should be used when starting and reviewing the schematics for a SAMHA0/1 design. This chapter illustrates a recommended power supply connection, how to connect external analog references, programmer, debugger, oscillator and crystal.

#### 38.1.1 Operation in Noisy Environment

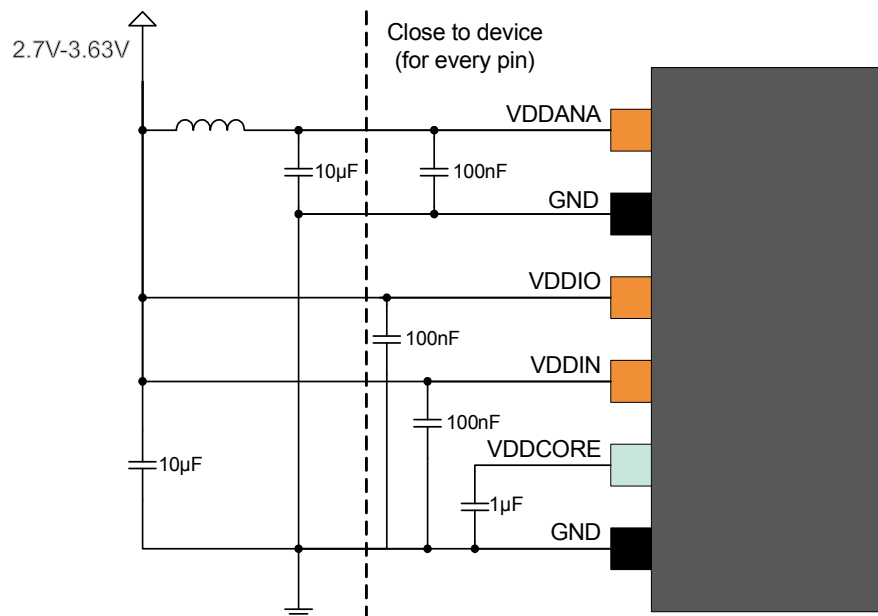
If the device is operating in an environment with much electromagnetic noise, it must be protected from this noise to ensure reliable operation. In addition to following best practice EMC design guidelines, the recommendations listed in the schematic checklist sections must be followed. In particular, placing decoupling capacitors very close to the power pins, an 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.

### 38.2 Power Supply

The SAMHA0/1 supports a single power supply from 2.7V - 3.63V.

#### 38.2.1 Power Supply Connections

**Figure 38-1. Power Supply Schematic**



**Table 38-1. Power Supply Connections,  $V_{\text{DDCORE}}$  From Internal Regulator**

Signal Name	Recommended Pin Connection	Description
$V_{\text{DDIO}}$	2.7V - 3.63V Decoupling/filtering capacitors 100nF <sup>(1)(2)</sup> and 10µF <sup>(1)</sup>	Digital supply voltage

Signal Name	Recommended Pin Connection	Description
	Decoupling/filtering inductor 10 $\mu$ H <sup>(1)(3)</sup>	
V <sub>DDANA</sub>	2.7V - 3.63V Decoupling/filtering capacitors 100nF <sup>(1)(2)</sup> and 10 $\mu$ F <sup>(1)</sup> Ferrite bead <sup>(4)</sup> prevents the V <sub>DD</sub> noise interfering the V <sub>DDANA</sub>	Analog supply voltage
V <sub>DDCORE</sub>	Decoupling/filtering capacitor 1 $\mu$ F <sup>(1)(2)</sup>	Core supply voltage / external decoupling pin
GND		Ground

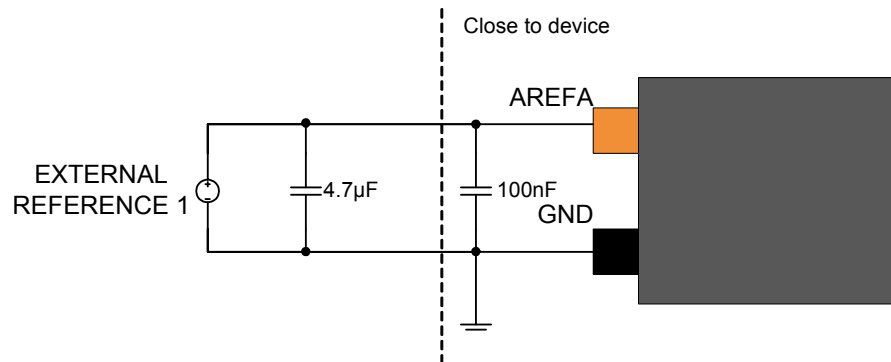
**Note:**

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.
3. An inductor should be added between the external power and the V<sub>DD</sub> for power filtering.
4. Ferrite bead has better filtering performance than the common inductor at high frequencies. It can be added between V<sub>DD</sub> and V<sub>DDANA</sub> for preventing digital noise from entering the analog power domain. The bead should provide enough impedance (e.g. 50 $\Omega$  at 20MHz and 220 $\Omega$  at 100MHz) for separating the digital power from the analog power domain. Be sure to select a ferrite bead designed for filtering applications with a low DC resistance to avoid a large voltage drop across the ferrite bead.

### 38.3 External Analog Reference Connections

The following schematic checklist is only necessary if the application is using external analog references. If the internal references are used instead, the following circuits are not necessary.

**Figure 38-2. External Analog Reference Schematic With One Reference**



**Table 38-2. External Analog Reference Connections**

Signal Name	Recommended Pin Connection	Description
AREFA	1.0V to V <sub>DDANA</sub> - 0.6V for ADC 1.0V to V <sub>DDANA</sub> - 0.6V for DAC	External reference from AREFA pin on the analog port

Signal Name	Recommended Pin Connection	Description
	Decoupling/filtering capacitors 100nF <sup>(1)(2)</sup> and 4.7μF <sup>(1)</sup>	
GND		Ground

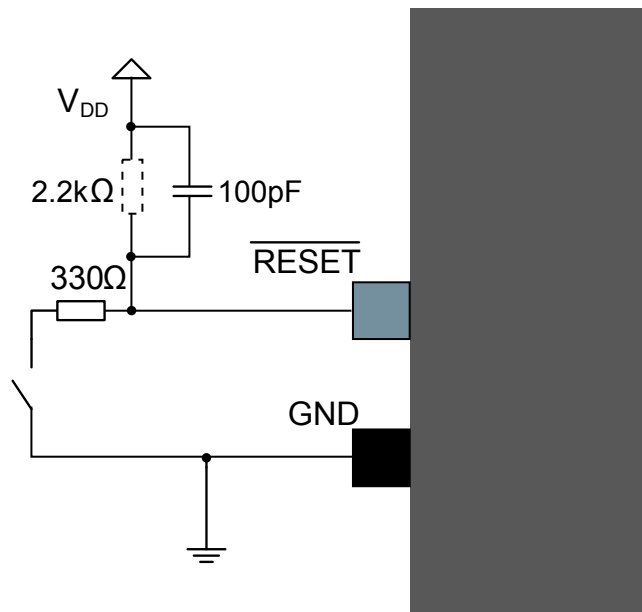
1. These values are given as a typical example.
2. Decoupling capacitor should be placed close to the device for each supply pin pair in the signal group.

### 38.4 External Reset Circuit

The external Reset circuit is connected to the  $\overline{\text{RESET}}$  pin when the external Reset function is used. The circuit is not necessary when the  $\overline{\text{RESET}}$  pin is not driven LOW externally by the application circuitry.

The reset switch can also be removed, if a manual reset is not desired. The  $\overline{\text{RESET}}$  pin itself has an internal pull-up resistor, hence it is optional to add any external pull-up resistor.

**Figure 38-3. External Reset Circuit Schematic**



A pull-up resistor makes sure that the reset does not go low and unintentionally cause 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 can cause a noise spike that can have a negative effect on the system.

**Table 38-3. Reset Circuit Connections**

Signal Name	Recommended Pin Connection	Description
$\overline{\text{RESET}}$	Reset low level threshold voltage  $V_{\text{DDIO}} = 2.7\text{V} - 3.6\text{V}$ : Below $0.36 * V_{\text{DDIO}}$ Decoupling/filter capacitor 100 pF <sup>(1)</sup> Pull-up resistor 2.2kΩ kΩ <sup>(1)(2)</sup> Resistor in series with the switch 330Ω <sup>(1)</sup>	Reset pin

1. These values are given as a typical example.



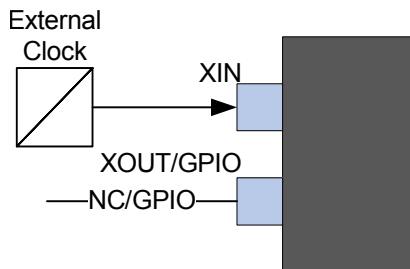
2. The SAMHA0/1 features an internal pull-up resistor on the  $\overline{\text{RESET}}$  pin, hence an external pull-up is optional.

### 38.5 Clocks and Crystal Oscillators

The SAMHA0/1 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).

#### 38.5.1 External Clock Source

**Figure 38-4. External Clock Source Example Schematic**

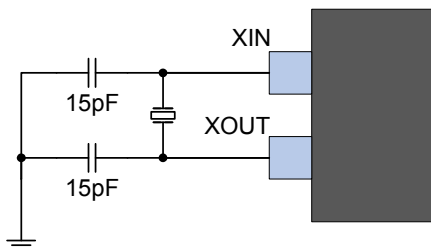


**Table 38-4. External Clock Source Connections**

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	

#### 38.5.2 Crystal Oscillator

**Figure 38-5. Crystal Oscillator Example Schematic**



The crystal should be located as close to the device as possible. Long signal lines may cause a load too high to operate the crystal, and cause crosstalk to other parts of the system.

**Table 38-5. Crystal Oscillator Checklist**

Signal Name	Recommended Pin Connection	Description
XIN	Load capacitor 15pF <sup>(1)(2)</sup>	External crystal between 0.4 to 30MHz
XOUT	Load capacitor 15pF <sup>(1)(2)</sup>	

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.

**38.5.3 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 SAMHA0/1 oscillator is optimized for very low power consumption, so pay close attention when selecting crystals. See the table below for maximum ESR recommendations on 9pF and 12.5pF crystals.

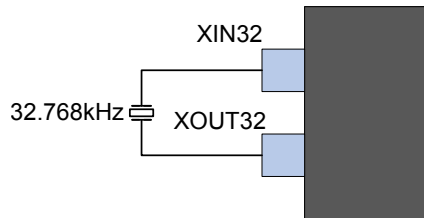
The low-frequency crystal oscillator provides an internal load capacitance of typical values available in Table , *32kHz Crystal Oscillator Characteristics*. This internal load capacitance and PCB capacitance can use a crystal inferior to 12.5pF load capacitance without external capacitors as shown in the following figure.

**Table 38-6. Maximum ESR Recommendation for 32.768kHz Crystal**

Crystal $C_L$ (pF)	Max ESR [kΩ]
12.5	313

Note: Maximum ESR is typical value based on characterization. These values are not covered by test limits in production.

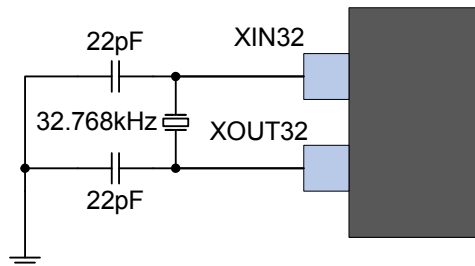
**Figure 38-6. External Real Time Oscillator without Load Capacitor**



However, to improve crystal accuracy and safety factor, the data sheet recommends adding external capacitors as shown in the next figure.

To find suitable load capacitance for a 32.768kHz crystal, consult the crystal datasheet.

**Figure 38-7. External Real Time Oscillator with Load Capacitor**



**Table 38-7. External Real Time Oscillator Checklist**

Signal Name	Recommended Pin Connection	Description
XIN32	Load capacitor 22pF <sup>(1)(2)</sup>	Timer oscillator input
XOUT32	Load capacitor 22pF <sup>(1)(2)</sup>	Timer oscillator output

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.

**Note:** In order to minimize the cycle-to-cycle jitter of the external oscillator, keep the neighboring pins as steady as possible. For neighboring pin details, refer to the Oscillator Pinout section.

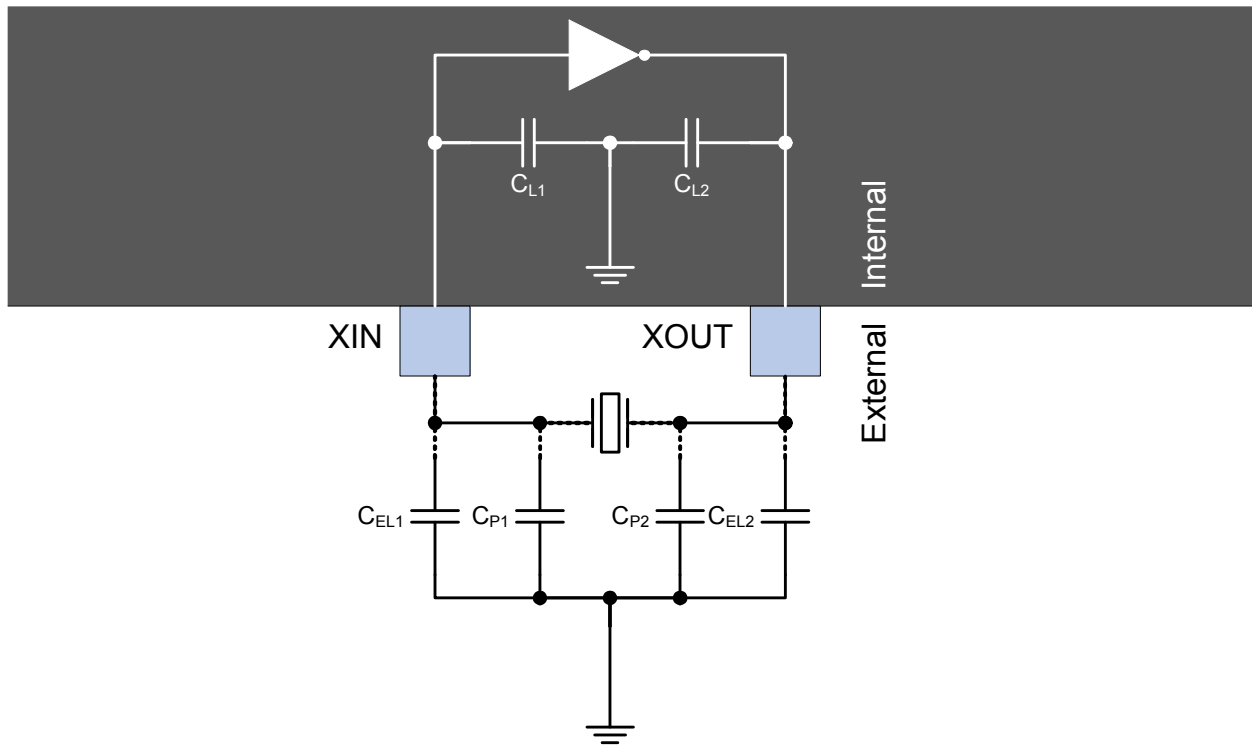
**Related Links**

[Oscillator Pinout](#)

### 38.5.4 Calculating the Correct Crystal Decoupling Capacitor

In order to calculate correct load capacitor for a given crystal, use the model shown in the next figure, which includes internal capacitors  $C_{Ln}$ , external parasitic capacitance  $C_{ELn}$  and external load capacitance  $C_{Pn}$ .

**Figure 38-8. 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 be disregarded in most applications as these are usually very small. If accounted for, the value is dependent on the PCB material and PCB layout.

For some crystals, 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:

$$\sum C_{tot} = \frac{C_L}{2}$$

The next table shows the device equivalent internal pin capacitance.

**Table 38-8. Equivalent Internal Pin Capacitance**

Symbol	Value	Description
$C_{XIN32}$	3.05pF	Equivalent internal pin capacitance
$C_{XOUT32}$	3.29pF	Equivalent internal pin capacitance

### 38.6 Unused or Unconnected Pins

For unused pins, the default state of the pins will provide the lowest current leakage. There is no need to do any configuration of the unused pins in order to lower the power consumption.

### 38.7 Programming and Debug Ports

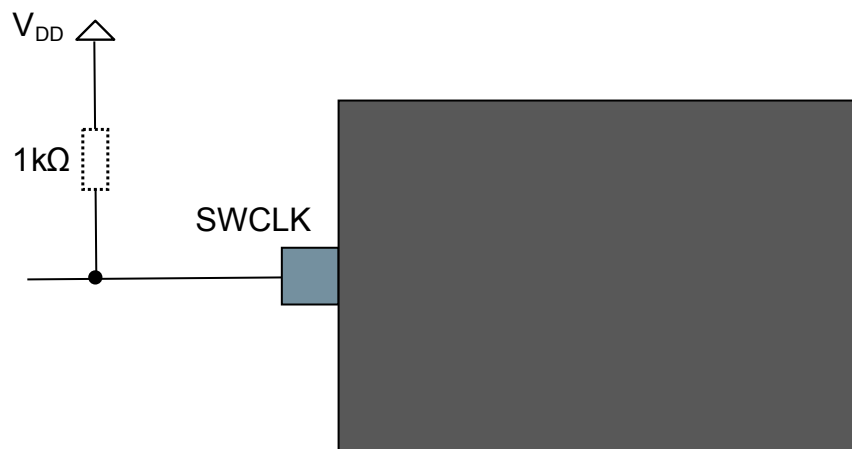
For programming and/or debugging the SAMHA0/1 the device should be connected using the Serial Wire Debug, SWD, interface. Currently the SWD interface is supported by several Microchip and third party programmers and debuggers, like the JTAGICE3, SAM-ICE, ATMEL\_ICE or SAMHA0/1 Xplained Pro ( SAMHA0/1 evaluation kit) Embedded Debugger.

Refer to the JTAGICE3, SAM-ICE, ATMEL\_ICE or SAMHA0/1 Xplained Pro user guides for details on debugging and programming connections and options. For connecting to any other programming or debugging tool, refer to that specific programmer or debugger's user guide.

The SAMHA0/1 Xplained Pro evaluation board for the SAMHA0/1 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 related link for more information.

**Figure 38-9. SWCLK Circuit Connections**



**Table 38-9. SWCLK Circuit Connections**

Pin Name	Description	Recommended Pin Connection
SWCLK	Serial wire clock pin	Pull-up resistor 1kΩ

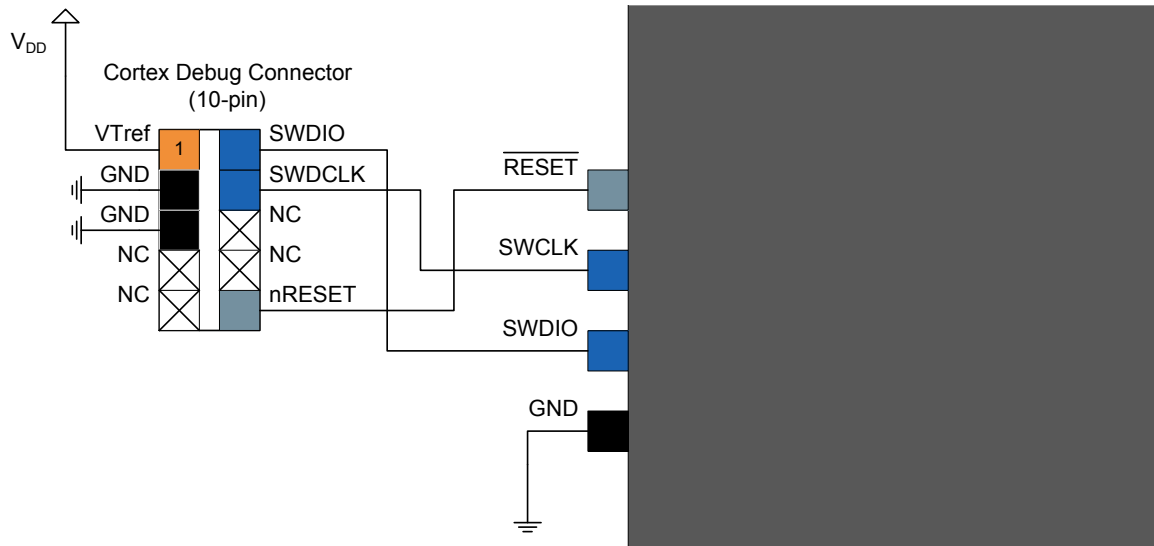
#### Related Links

[Operation in Noisy Environment](#)

### 38.7.1 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 figure below with details described in the next table.

**Figure 38-10. Cortex Debug Connector (10-pin)**



**Table 38-10. Cortex Debug Connector (10-pin)**

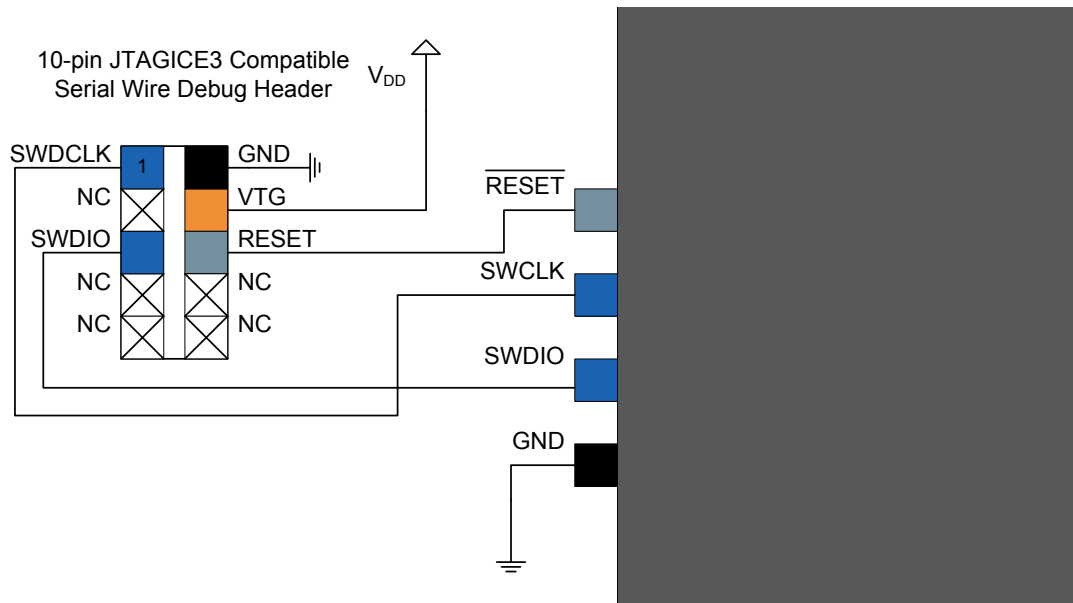
Header Signal Name	Description	Recommended Pin Connection
SWDCLK	Serial wire clock pin	Pull-up resistor 1k $\Omega$
SWDIO	Serial wire bidirectional data pin	
$\overline{\text{RESET}}$	Target device reset pin, active low Refer to <a href="#">External Reset Circuit</a> .	
Vtref	Target voltage sense, should be connected to the device $V_{DD}$	
GND	Ground	

### 38.7.2 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 SAMHA0/1 to the JTAGICE3, alternatively one can use the JTAGICE3 squid cable and manually match the signals between the JTAGICE3 and SAMHA0/1. The following figure describes how to connect a 10-pin header that support connecting the JTAGICE3 directly to the SAMHA0/1 without the need for a squid cable.

To connect the JTAGICE3 programmer and debugger to the SAMHA0/1, one can either use the JTAGICE3 squid cable, or use a 10-pin connector as shown in the figure below with details given in the next table to connect to the target using the JTAGICE3 50 mil cable directly.

**Figure 38-11. 10-pin JTAGICE3 Compatible Serial Wire Debug Interface**



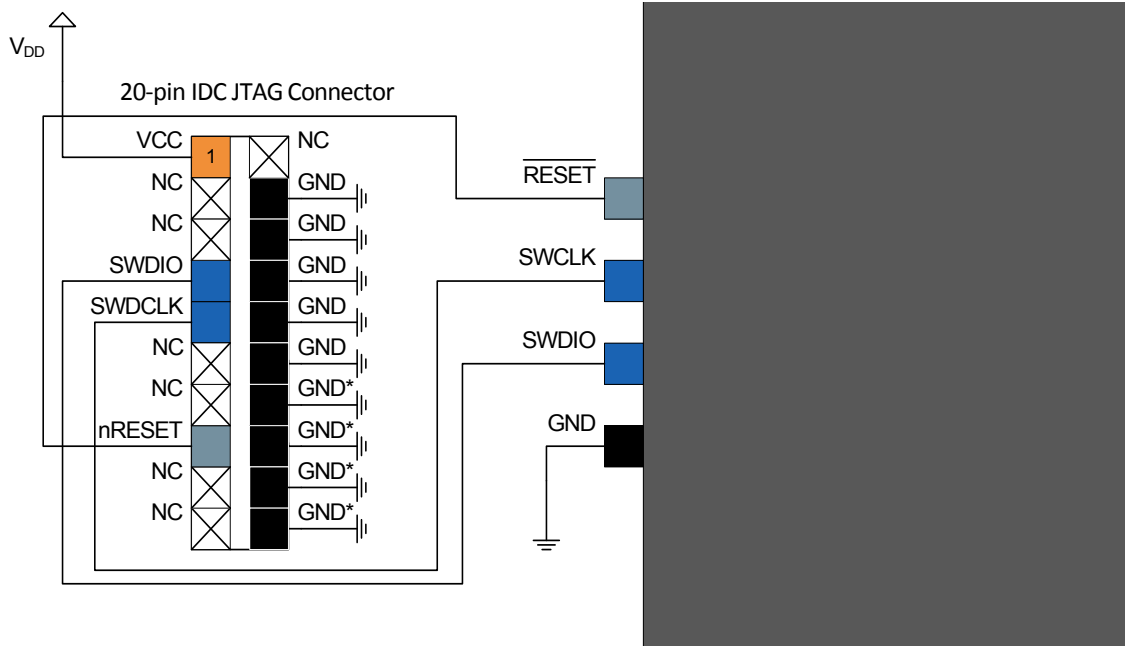
**Table 38-11. 10-pin JTAGICE3 Compatible Serial Wire Debug Interface**

Header 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

### 38.7.3 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 next figure with details described in the table.

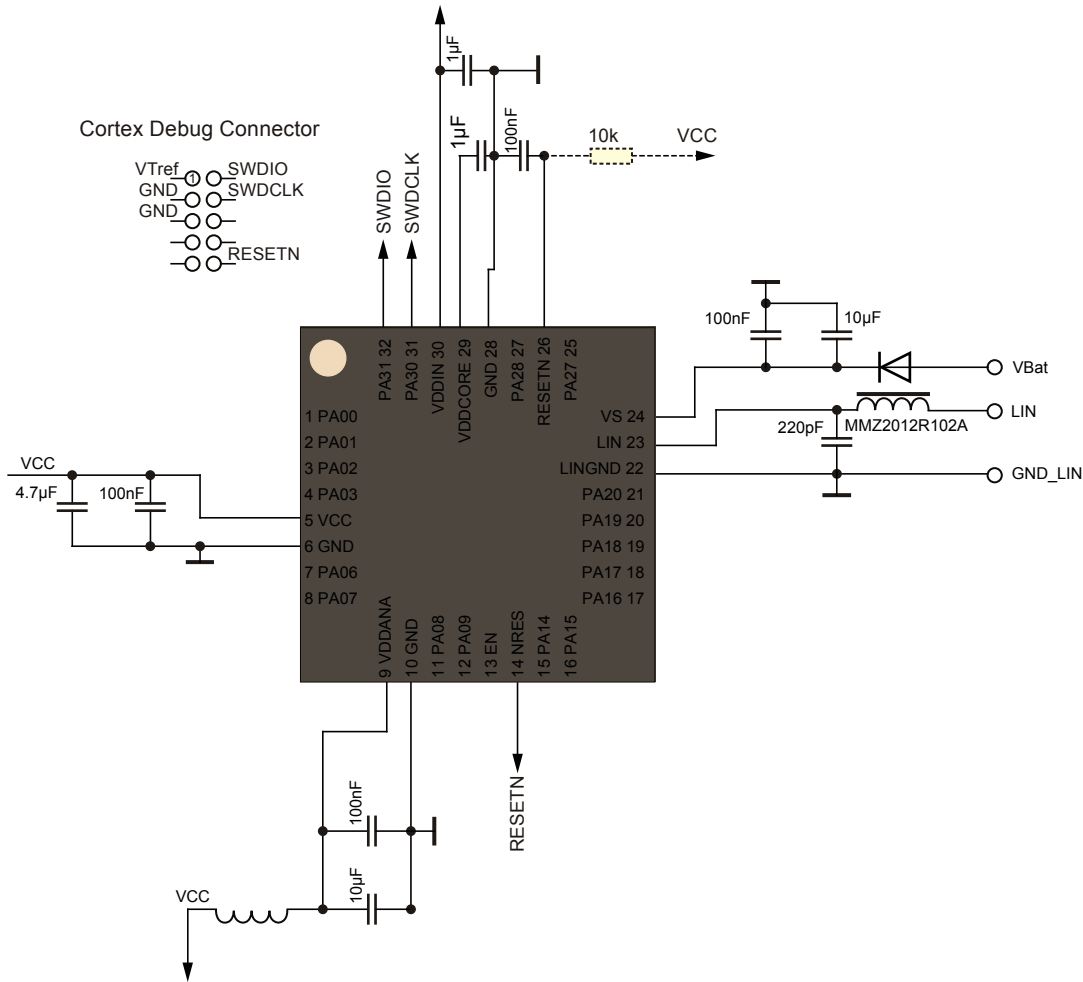
**Figure 38-12. 20-pin IDC JTAG Connector**



**Table 38-12. 20-pin IDC JTAG Connector**

Header Signal Name	Description
SWDCLK	Serial wire clock pin
SWDIO	Serial wire bidirectional data pin
$\overline{\text{RESET}}$	Target device reset pin, active low
VCC	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.

### 39. Typical Application Diagram



**Note:** These values are only given as typical examples.

**Note:** The decoupling capacitor (100nF) should be placed close to the device for each supply pin pair in the signal group and the low ESR caps should be used for better decoupling.



---

---

## 40. Errata

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.

### 40.1 Die Revision F

#### 40.1.1 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: DFLLCKC and DFLLCKF in the SYSCtrl Interrupt Flag Status and Clear register (INTFLAG) are both set before enabling the DFLL\_OOB interrupt.

#### 40.1.2 FDPLL

**1 – When changing on-the-fly the FDPLL ratio in DPLLnRATIO register, STATUS.DPLLnLDRTO will not be set when the ratio update will be completed.**

**Errata reference: 15753**

**Fix/Workaround:**

Wait for the interruption flag INTFLAG.DPLLnLDRTO instead.

#### 40.1.3 Device

**1 – The SYSTICK calibration value is incorrect.**

**Errata reference: 14155**

**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.

**2 – On pin PA24 and PA25 the pull-up and pull-down configuration is not disabled automatically when alternative pin function is enabled except for USB.**

**Errata reference: 12368**

**Fix/Workaround:**

For pin PA24 and PA25, the GPIO pull-up and pull-down must be disabled before enabling alternative functions on them.

**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 – 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.

### 40.1.4 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.

### 40.1.5 DMAC

**1 – When at least one channel using linked descriptors is already active, enabling another DMA channel (with or without linked descriptors) can result in a channel Fetch Error (FERR) or an incorrect descriptor fetch.**

**This happens if the channel number of the channel being enabled is lower than the channel already active.**

**Errata reference: 15683**

**Fix/Workaround:**

When enabling a DMA channel while other channels using linked descriptors are already active, the channel number of the new channel enabled must be greater than the other channel numbers.

**2 – 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.

### 40.1.6 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.

### 40.1.7 NVMCTRL

**1 – Default value of MANW in NVM.CTRLB is 0.**

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.

**Errata reference: 13134**

**Fix/Workaround:**

Set MANW in the NVM.CTRLB to 1 at startup

### 40.1.8 SERCOM

**1 – In USART autobaud mode, missing stop bits are not recognized as inconsistent sync (ISF) or framing (FERR) errors.**

**Errata reference: 13852**

**Fix/Workaround:**

None

### 40.1.9 TCC

**1 – FCTRLX.CAPTURE[CAPTMARK] does not work as described in the datasheet. CAPTMARK cannot be used to identify captured values triggered by fault inputs source A or B on the same channel.**

**Errata reference: 13316**

**Fix/Workaround:**

Use two different channels to timestamp FaultA and FaultB.

**2 – Using TCC in dithering mode with external retrigger events can lead to unexpected stretch of right aligned pulses, or shrink of left aligned pulses.**

**Errata reference: 15625**

**Fix/Workaround:**

Do not use retrigger events/actions when TCC is configured in dithering mode.

**3 – Advance capture mode (CAPTMIN CAPTMAX LOCMIN LOCMAX DERIV0) doesn't work if an upper channel is not in one of these mode.**

**Example: when CC[0]=CAPTMIN, CC[1]=CAPTMAX, CC[2]=CAPTEN, and CC[3]=CAPTEN, CAPTMIN and CAPTMAX won't work.**

**Errata reference: 14817**

**Fix/Workaround:**

Basic capture mode must be set in lower channel and advance capture mode in upper channel.

Example: CC[0]=CAPTEN , CC[1]=CAPTEN , CC[2]=CAPTMIN, CC[3]=CAPTMAX

All capture will be done as expected.

## 41. Conventions

### 41.1 Numerical Notation

**Table 41-1. Numerical Notation**

Symbol	Description
165	Decimal number
0b0101	Binary number (example 0b0101 = 5 decimal)
'0101'	Binary numbers are given without prefix 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

### 41.2 Memory Size and Type

**Table 41-2. 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
word	32 bit
half-word	16 bit

### 41.3 Frequency and Time

**Table 41-3. Frequency and Time**

Symbol	Description
kHz	1 kHz = $10^3$ Hz = 1,000 Hz
KHz	1 KHz = 1,024 Hz, 32 KHz = 32,768 Hz

Symbol	Description
MHz	1 MHz = $10^6$ Hz = 1,000,000 Hz
GHz	1 GHz = $10^9$ Hz = 1,000,000,000 Hz
s	second
ms	millisecond
$\mu$ s	microsecond
ns	nanosecond

### 41.4 Registers and Bits

**Table 41-4. Register and Bit Mnemonics**

Symbol	Description
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 ENABLE)
FIELD[n:m]	A set of bits from bit n down to m. (Example: PINA[3:0] = {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.  Reserved bit field values must not be written to a bit field. A reserved value won't be read from a read-only bit field.
PERIPHERAL <i>i</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. PERIPHERAL0 denotes one specific instance.
Reset	Value of a register after a power Reset. This is also the value of registers in a peripheral after performing a software Reset of the peripheral, except for the Debug Control registers.
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 '1' to a bit in the CLR register will clear the corresponding bit in both registers, while writing a '1' 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.

## 42. Acronyms and Abbreviations

The below table contains acronyms and abbreviations used in this document.

**Table 42-1. Acronyms and Abbreviations**

Abbreviation	Description
AC	Analog Comparator
ADC	Analog-to-Digital Converter
ADDR	Address
AES	Advanced Encryption Standard
AHB	AMBA Advanced High-performance Bus
AMBA <sup>®</sup>	Advanced Microcontroller Bus Architecture
APB	AMBA Advanced Peripheral Bus
AREF	Analog reference voltage
BLB	Boot Lock Bit
BOD	Brown-out detector
CAL	Calibration
CC	Compare/Capture
CCL	Configurable Custom Logic
CLK	Clock
CRC	Cyclic Redundancy Check
CTRL	Control
DAC	Digital-to-Analog Converter
DAP	Debug Access Port
DFLL	Digital Frequency Locked Loop
DPLL	Digital Phase Locked Loop
DMAC	DMA (Direct Memory Access) Controller
DSU	Device Service Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
EIC	External Interrupt Controller
EVSYS	Event System
FDPLL	Fractional Digital Phase Locked Loop, also DPLL
GCLK	Generic Clock Controller
GND	Ground

# ATSAMHAXEXXA

## Acronyms and Abbreviations

Abbreviation	Description
GPIO	General Purpose Input/Output
I <sup>2</sup> C	Inter-Integrated Circuit
IF	Interrupt flag
INT	Interrupt
MBIST	Memory built-in self-test
MEM-AP	Memory Access Port
MTB	Micro Trace Buffer
NMI	Non-maskable interrupt
NVIC	Nested Vector Interrupt Controller
NVM	Non-Volatile Memory
NVMCTRL	Non-Volatile Memory Controller
OSC	Oscillator
PAC	Peripheral Access Controller
PC	Program Counter
PER	Period
PM	Power Manager
POR	Power-on reset
PORT	I/O Pin Controller
PTC	Peripheral Touch Controller
PWM	Pulse Width Modulation
RAM	Random-Access Memory
REF	Reference
RTC	Real-Time Counter
RX	Receiver/Receive
SERCOM	Serial Communication Interface
SMBus™	System Management Bus
SP	Stack Pointer
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SUPC	Supply Controller
SWD	Serial Wire Debug
TC	Timer/Counter

# ATSAMHAXEXXA

## Acronyms and Abbreviations

Abbreviation	Description
TCC	Timer/Counter for Control Applications
TRNG	True Random Number Generator
TX	Transmitter/Transmit
ULP	Ultra-low power
USART	Universal Synchronous and Asynchronous Serial Receiver and Transmitter
V <sub>DD</sub>	Common voltage to be applied to VDDIO, VDDIN and VDDANA
V <sub>DDIN</sub>	Digital supply voltage
V <sub>DDIO</sub>	Digital supply voltage
V <sub>DDANA</sub>	Analog supply voltage
VREF	Voltage reference
WDT	Watchdog Timer
XOSC	Crystal Oscillator



**43. Data Sheet Revision History**

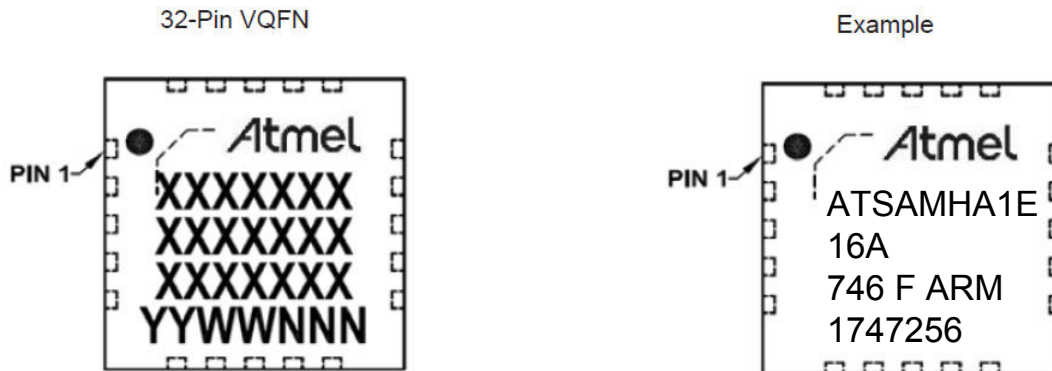
Page numbers listed in this section refer to this document. The revision listed in this section refers to the document revision.

**43.1 Revision A - 12/2017**

Original release of the document.

## 44. Packaging Information

### 44.1 Package Marking Information



Legend:

**XX...X** Customer-specific information

**Y** Year code (last digit of calendar year)

**YY** Year code (last 2 digits of calendar year)

**WW** Week code (week of January 1 is week '01')

**NNN** Alphanumeric traceability code

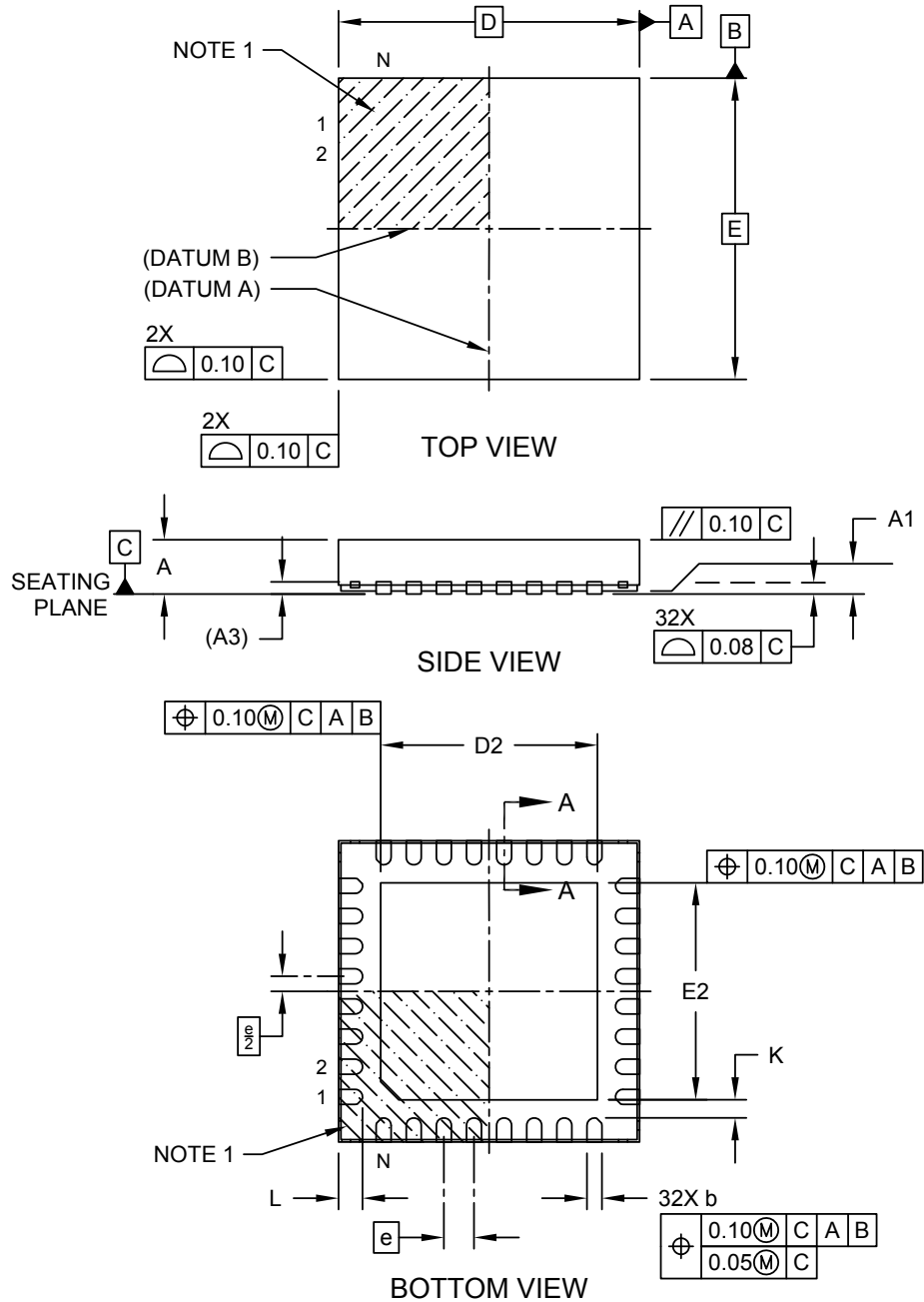
**e3** JEDEC® designator for Matte Tin (Sn)

\* This package is RoHS compliant. The JEDEC designator ( **e3** ) can be found on the outer packaging for this package.

**Note:** In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

### 32-Lead Very Thin Plastic Quad Flat, No Lead Package (RTB) - 5x5 mm Body [VQFN] With 3.6x3.6 mm Exposed Pad and Wettable Flanks

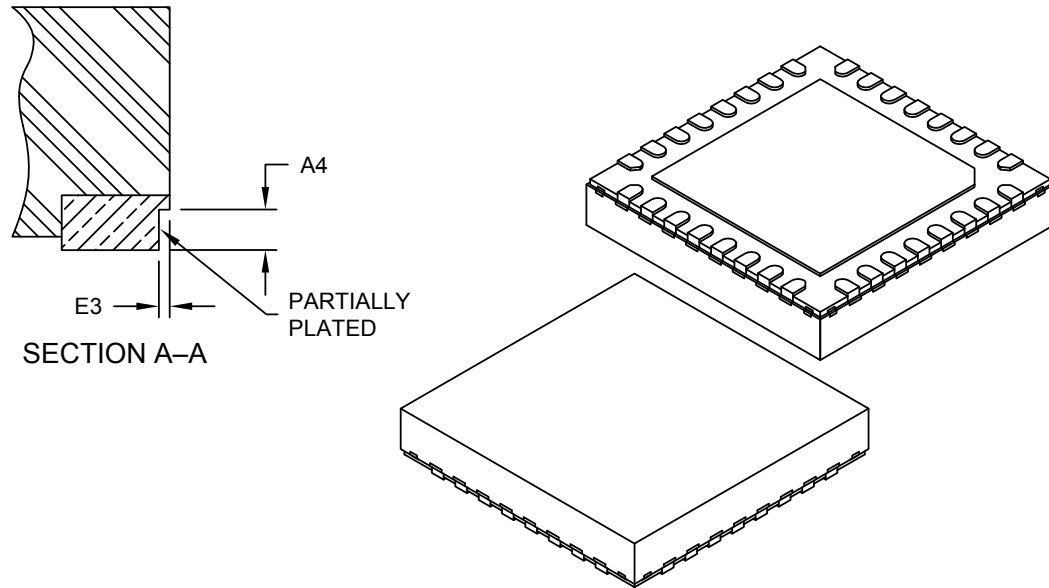
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing C04-21391 Rev A Sheet 1 of 2

### 32-Lead Very Thin Plastic Quad Flat, No Lead Package (RTB) - 5x5 mm Body [VQFN] With 3.6x3.6 mm Exposed Pad and Wettable Flanks

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Terminals	N	32		
Pitch	e	0.50 BSC		
Overall Height	A	0.80	0.85	0.90
Standoff	A1	0.00	0.035	0.05
Terminal Thickness	A3	0.203 REF		
Wettable Flank Step Cut Depth	A4	0.10	0.13	0.15
Overall Length	D	5.00 BSC		
Exposed Pad Length	D2	3.50	3.60	3.70
Overall Width	E	5.00 BSC		
Exposed Pad Width	E2	3.50	3.60	3.70
Wettable Flank Step Cut Width	E3	-	-	0.04
Terminal Width	b	0.20	0.25	0.30
Terminal Length	L	0.35	0.40	0.45
Terminal-to-Exposed-Pad	K	0.20	-	-

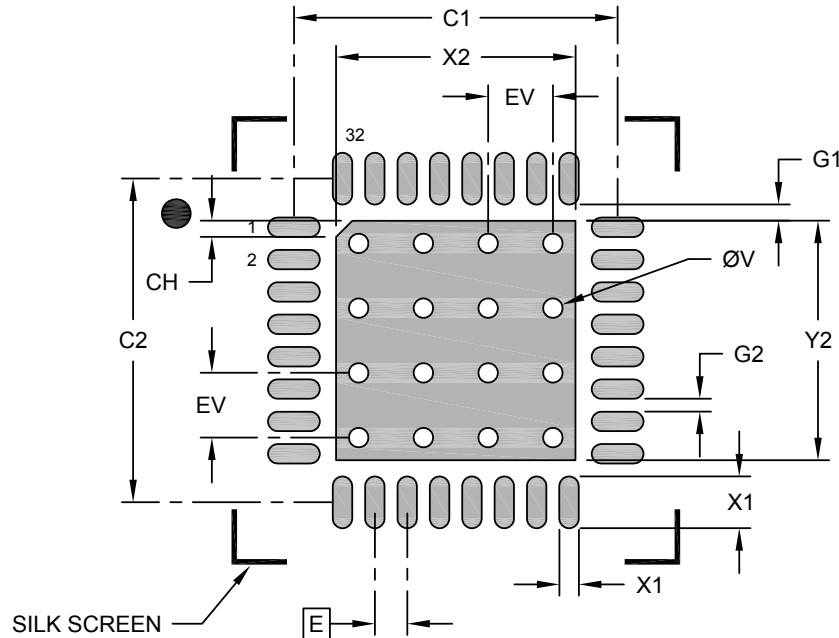
**Notes:**

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Package is saw singulated
- Dimensioning and tolerancing per ASME Y14.5M  
BSC: Basic Dimension. Theoretically exact value shown without tolerances.  
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21391 Rev A Sheet 2 of 2

### 32-Lead Very Thin Plastic Quad Flat, No Lead Package (RTB) - 5x5 mm Body [VQFN] With 3.6x3.6 mm Exposed Pad and Wettable Flanks

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E	0.50 BSC		
Optional Center Pad Width	X2			3.70
Optional Center Pad Length	Y2			3.70
Exposed Pad 45° Corner Chamfer	CH		0.25	
Contact Pad Spacing	C1		5.00	
Contact Pad Spacing	C2		5.00	
Contact Pad Width (X32)	X1			0.30
Contact Pad Length (X32)	Y1			0.80
Contact Pad to Center Pad (X32)	G1	0.25		
Contact Pad to Contact Pad (X28)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M  
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23391 Rev. A

## The Microchip Web Site

---

Microchip provides online support via our web site at <http://www.microchip.com/>. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## Customer Change Notification Service

---

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at <http://www.microchip.com/>. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## Customer Support

---

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: <http://www.microchip.com/support>

## Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

<u>PART NO.</u>	<u>XX</u>	<u>X</u>	<u>XX</u>	<u>X</u>	<u>-XX</u>	<u>X</u>	<u>X</u>	<u>-X</u>
Device	Product Series	Pin Count	Flash Memory Density	Device Version	Package	Temp. Range	Tape and Reel	Device Variant

Device:	ATSAMHA1E14A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 16 kB Flash ATSAMHA1E15A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 32 kB Flash ATSAMHA1E16A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 64 kB Flash ATSAMHA0E14A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 16 kB Flash, 115°C Tcase ATSAMHA0E15A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 32 kB Flash, 115°C Tcase ATSAMHA0E16A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 64 kB Flash, 115°C Tcase	
Part No.	ATSAMH	Automotive System in Package with LIN-SBC
Product Series	A1	Cortex M0+ DMA, PTC
	A0	Cortex M0+ DMA, 115°C Tc
Pin Count	E	32 pins
Flash Memory Density	14	16kB
	15	32kB
	16	64kB
Device version:	A	SiP with LIN TRX + VREG
Package:	M	= 32-Lead Very Thin Plastic Quad Flat, No Lead Package (RTB) - 5x5 mm Body [VQFN] With 3.6x3.6 mm Exposed Pad and Wettable Flanks
Temperature:	B	-40°C to +105°C Tcase
	Z	-40°C to +115°C Tcase
Tape and Reel	T	Tape and Reel option
Device Variant	B	Device variant B

Examples:

- ATSAMHA1E16A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 32L-VQFN package, 64 kB Flash
- ATSAMHA1E15A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 32L-VQFN package, 32 kB Flash
- ATSAMHA1E14A-MBT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 32L-VQFN package, 16 kB Flash
- ATSAMHA0E16A-MZT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 32L-VQFN package, 64 kB Flash, 115°C Tc
- ATSAMHA0E15A-MZT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 32L-VQFN package, 32 kB Flash, 115°C Tc
- ATSAMHA0E14A-MZT-B: Low-power SIP, 32 bit ARM Cortex M0+ processor, 32L-VQFN package, 16 kB Flash, 115°C Tc

**Note:**

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check <http://www.microchip.com/packaging> for small-form factor package availability, or contact your local Sales Office.

## Microchip Devices Code Protection Feature

---

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

## Legal Notice

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life



support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

---

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-2505-2

## Quality Management System Certified by DNV

---

### ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC<sup>®</sup> MCUs and dsPIC<sup>®</sup> DSCs, KEELOQ<sup>®</sup> code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

## Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
<p><b>Corporate Office</b> 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: <a href="http://www.microchip.com/support">http://www.microchip.com/support</a> Web Address: <a href="http://www.microchip.com">www.microchip.com</a></p> <p><b>Atlanta</b> Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455</p> <p><b>Austin, TX</b> Tel: 512-257-3370</p> <p><b>Boston</b> Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088</p> <p><b>Chicago</b> Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075</p> <p><b>Dallas</b> Addison, TX Tel: 972-818-7423 Fax: 972-818-2924</p> <p><b>Detroit</b> Novi, MI Tel: 248-848-4000</p> <p><b>Houston, TX</b> Tel: 281-894-5983</p> <p><b>Indianapolis</b> Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380</p> <p><b>Los Angeles</b> Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800</p> <p><b>Raleigh, NC</b> Tel: 919-844-7510</p> <p><b>New York, NY</b> Tel: 631-435-6000</p> <p><b>San Jose, CA</b> Tel: 408-735-9110 Tel: 408-436-4270</p> <p><b>Canada - Toronto</b> Tel: 905-695-1980 Fax: 905-695-2078</p>	<p><b>Australia - Sydney</b> Tel: 61-2-9868-6733</p> <p><b>China - Beijing</b> Tel: 86-10-8569-7000</p> <p><b>China - Chengdu</b> Tel: 86-28-8665-5511</p> <p><b>China - Chongqing</b> Tel: 86-23-8980-9588</p> <p><b>China - Dongguan</b> Tel: 86-769-8702-9880</p> <p><b>China - Guangzhou</b> Tel: 86-20-8755-8029</p> <p><b>China - Hangzhou</b> Tel: 86-571-8792-8115</p> <p><b>China - Hong Kong SAR</b> Tel: 852-2943-5100</p> <p><b>China - Nanjing</b> Tel: 86-25-8473-2460</p> <p><b>China - Qingdao</b> Tel: 86-532-8502-7355</p> <p><b>China - Shanghai</b> Tel: 86-21-3326-8000</p> <p><b>China - Shenyang</b> Tel: 86-24-2334-2829</p> <p><b>China - Shenzhen</b> Tel: 86-755-8864-2200</p> <p><b>China - Suzhou</b> Tel: 86-186-6233-1526</p> <p><b>China - Wuhan</b> Tel: 86-27-5980-5300</p> <p><b>China - Xian</b> Tel: 86-29-8833-7252</p> <p><b>China - Xiamen</b> Tel: 86-592-2388138</p> <p><b>China - Zhuhai</b> Tel: 86-756-3210040</p>	<p><b>India - Bangalore</b> Tel: 91-80-3090-4444</p> <p><b>India - New Delhi</b> Tel: 91-11-4160-8631</p> <p><b>India - Pune</b> Tel: 91-20-4121-0141</p> <p><b>Japan - Osaka</b> Tel: 81-6-6152-7160</p> <p><b>Japan - Tokyo</b> Tel: 81-3-6880-3770</p> <p><b>Korea - Daegu</b> Tel: 82-53-744-4301</p> <p><b>Korea - Seoul</b> Tel: 82-2-554-7200</p> <p><b>Malaysia - Kuala Lumpur</b> Tel: 60-3-7651-7906</p> <p><b>Malaysia - Penang</b> Tel: 60-4-227-8870</p> <p><b>Philippines - Manila</b> Tel: 63-2-634-9065</p> <p><b>Singapore</b> Tel: 65-6334-8870</p> <p><b>Taiwan - Hsin Chu</b> Tel: 886-3-577-8366</p> <p><b>Taiwan - Kaohsiung</b> Tel: 886-7-213-7830</p> <p><b>Taiwan - Taipei</b> Tel: 886-2-2508-8600</p> <p><b>Thailand - Bangkok</b> Tel: 66-2-694-1351</p> <p><b>Vietnam - Ho Chi Minh</b> Tel: 84-28-5448-2100</p>	<p><b>Austria - Wels</b> Tel: 43-7242-2244-39 Fax: 43-7242-2244-393</p> <p><b>Denmark - Copenhagen</b> Tel: 45-4450-2828 Fax: 45-4485-2829</p> <p><b>Finland - Espoo</b> Tel: 358-9-4520-820</p> <p><b>France - Paris</b> Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79</p> <p><b>Germany - Garching</b> Tel: 49-8931-9700</p> <p><b>Germany - Haan</b> Tel: 49-2129-3766400</p> <p><b>Germany - Heilbronn</b> Tel: 49-7131-67-3636</p> <p><b>Germany - Karlsruhe</b> Tel: 49-721-625370</p> <p><b>Germany - Munich</b> Tel: 49-89-627-144-0 Fax: 49-89-627-144-44</p> <p><b>Germany - Rosenheim</b> Tel: 49-8031-354-560</p> <p><b>Israel - Ra'anana</b> Tel: 972-9-744-7705</p> <p><b>Italy - Milan</b> Tel: 39-0331-742611 Fax: 39-0331-466781</p> <p><b>Italy - Padova</b> Tel: 39-049-7625286</p> <p><b>Netherlands - Drunen</b> Tel: 31-416-690399 Fax: 31-416-690340</p> <p><b>Norway - Trondheim</b> Tel: 47-7289-7561</p> <p><b>Poland - Warsaw</b> Tel: 48-22-3325737</p> <p><b>Romania - Bucharest</b> Tel: 40-21-407-87-50</p> <p><b>Spain - Madrid</b> Tel: 34-91-708-08-90 Fax: 34-91-708-08-91</p> <p><b>Sweden - Gothenberg</b> Tel: 46-31-704-60-40</p> <p><b>Sweden - Stockholm</b> Tel: 46-8-5090-4654</p> <p><b>UK - Wokingham</b> Tel: 44-118-921-5800 Fax: 44-118-921-5820</p>