

## Introduction

The Atmel® | SMART™ SAM C21 is a series of microcontrollers optimized for industrial automation, appliances and other 5V applications using the 32-bit ARM® Cortex®-M0+ processor, ranging from 32- to 64-pins with up to 256KB Flash and 32KB of SRAM and operate at a maximum frequency of 48MHz and reach 2.46 CoreMark®/MHz. The SAM C21 devices are designed for simple and intuitive migration with identical peripheral modules, hex compatible code, identical linear address map and pin compatible migration paths between all devices in the product series. All devices include intelligent and flexible peripherals, Atmel Event System for inter-peripheral signaling, and support for capacitive touch button, slider and wheel user interfaces.

SAM C21 devices are pin compatible to the SAM D family of general purpose microcontrollers.

## Features

- Processor
  - ARM Cortex-M0+ CPU running at up to 48MHz
    - Single-cycle hardware multiplier
    - Micro Trace Buffer
    - Memory Protection Unit (MPU)
- Memories
  - 32/64/128/256KB in-system self-programmable Flash
  - 1/2/4/8KB independent self-programmable Flash for EEPROM emulation
  - 4/8/16/32KB SRAM Main Memory
- System
  - Power-on reset (POR) and brown-out detection (BOD)
  - Internal and external clock options with 48MHz to 96MHz Fractional Digital Phase Locked Loop (FDPLL96M)
  - External Interrupt Controller (EIC)
  - 16 external interrupts
  - One non-maskable interrupt

- Two-pin Serial Wire Debug (SWD) programming, test and debugging interface
- Low Power
  - Idle, standby, and off sleep modes
  - SleepWalking peripherals
- Peripherals
  - Hardware Divide and Square Root Accelerator (DIVAS)
  - 12-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
  - Two 24-bit Timer/Counters and one 16-bit Timer/Counter 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 that increase 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 two Controller Area Network (CAN) interfaces:
    - CAN 2.0A/B
    - CAN-FD 1.0
      - Each CAN interface have two selectable pin locations to switch between two external CAN transceivers (without the need for an external switch)
  - Up to six 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
    - SPI
    - LIN master/slave
    - RS-485
  - One Configurable Custom Logic (CCL)
  - Two 12-bit, 1Msps Analog-to-Digital Converter (ADC) with up to 12 channels each (20 unique channels in total)
    - Differential and single-ended input
    - Automatic offset and gain error compensation
    - Oversampling and decimation in hardware to support 13-, 14-, 15- or 16-bit resolution
  - One 16-bit, 1ksps Sigma-Delta Analog-to-Digital Converter (SDADC) with up to 3 differential channels
  - 10-bit, 300ksps Digital-to-Analog Converter (DAC)
  - Four Analog Comparators (AC) with window compare function
  - Integrated Temperature Sensor with  $\pm 1^{\circ}\text{C}$  accuracy

- Peripheral Touch Controller (PTC)
    - 256-Channel capacitive touch and proximity sensing
- I/O
  - Up to 52 programmable I/O pins
- Drop in compatible with SAM D20 and SAM D21
- Packages
  - 64-pin TQFP, QFN
  - 48-pin TQFP, QFN
  - 32-pin TQFP, QFN
- Operating Voltage
  - 2.7V – 5.5V

## Table of Contents

---

Introduction.....	1
Features.....	1
1. Description.....	13
2. Configuration Summary.....	14
3. Ordering Information.....	16
3.1. SAM C21E.....	16
3.2. SAM C21G.....	17
3.3. SAM C21J.....	18
4. Block Diagram.....	20
5. Pinout.....	22
5.1. SAM C21E.....	22
5.2. SAM C21G.....	23
5.3. SAM C21J.....	24
6. Signal Descriptions List.....	25
7. I/O Multiplexing and Considerations.....	27
7.1. Multiplexed Signals.....	27
7.2. Other Functions.....	28
8. Power Supply and Start-Up Considerations.....	30
8.1. Power Domain Overview.....	30
8.2. Power Supply Considerations.....	30
8.3. Power-Up.....	32
8.4. Power-On Reset and Brown-Out Detector.....	33
9. Product Mapping.....	34
10. Memories.....	35
10.1. Embedded Memories.....	35
10.2. Physical Memory Map.....	35
10.3. NVM User Row Mapping.....	36
10.4. NVM Software Calibration Area Mapping.....	37
10.5. NVM Temperature Calibration Area Mapping.....	37
10.6. Serial Number.....	38
11. Processor and Architecture.....	39
11.1. Cortex M0+ Processor.....	39
11.2. Nested Vector Interrupt Controller.....	41
11.3. Micro Trace Buffer.....	42



11.4. High-Speed Bus System.....	43
12. PAC - Peripheral Access Controller.....	46
12.1. Overview.....	46
12.2. Features.....	46
12.3. Block Diagram.....	46
12.4. Product Dependencies.....	46
12.5. Functional Description.....	47
12.6. Register Summary.....	51
12.7. Register Description.....	51
13. Peripherals Configuration Summary.....	57
14. DSU - Device Service Unit.....	59
14.1. Overview.....	59
14.2. Features.....	59
14.3. Block Diagram.....	60
14.4. Signal Description.....	60
14.5. Product Dependencies.....	60
14.6. Debug Operation.....	61
14.7. Chip Erase.....	63
14.8. Programming.....	64
14.9. Intellectual Property Protection.....	64
14.10. Device Identification.....	66
14.11. Functional Description.....	67
14.12. Register Summary.....	72
14.13. Register Description.....	74
15. DIVAS – Divide and Square Root Accelerator.....	97
15.1. Overview.....	97
15.2. Features.....	97
15.3. Block Diagram.....	97
15.4. Signal Description.....	97
15.5. Product Dependencies.....	97
15.6. Functional Description.....	98
15.7. Register Summary.....	101
15.8. Register Description.....	101
16. Clock System.....	109
16.1. Clock Distribution.....	109
16.2. Synchronous and Asynchronous Clocks.....	110
16.3. Register Synchronization.....	110
16.4. Enabling a Peripheral.....	112
16.5. On-demand, Clock Requests.....	112
16.6. Power Consumption vs. Speed.....	113
16.7. Clocks after Reset.....	113
17. GCLK - Generic Clock Controller.....	114
17.1. Overview.....	114

17.2. Features.....	114
17.3. Block Diagram.....	114
17.4. Signal Description.....	115
17.5. Product Dependencies.....	115
17.6. Functional Description.....	116
17.7. Register Summary.....	121
17.8. Register Description.....	122
<b>18. MCLK – Main Clock.....</b>	<b>131</b>
18.1. Overview.....	131
18.2. Features.....	131
18.3. Block Diagram.....	131
18.4. Signal Description.....	131
18.5. Product Dependencies.....	131
18.6. Functional Description.....	133
18.7. Register Summary.....	138
18.8. Register Description.....	138
<b>19. RSTC – Reset Controller.....</b>	<b>156</b>
19.1. Overview.....	156
19.2. Features.....	156
19.3. Block Diagram.....	156
19.4. Signal Description.....	156
19.5. Product Dependencies.....	156
19.6. Functional Description.....	157
19.7. Register Summary.....	160
19.8. Register Description.....	160
<b>20. PM – Power Manager.....</b>	<b>162</b>
20.1. Overview.....	162
20.2. Features.....	162
20.3. Block Diagram.....	162
20.4. Signal Description.....	162
20.5. Product Dependencies.....	162
20.6. Functional Description.....	163
20.7. Register Summary.....	167
20.8. Register Description.....	167
<b>21. OSCCTRL – Oscillators Controller.....</b>	<b>170</b>
21.1. Overview.....	170
21.2. Features.....	170
21.3. Block Diagram.....	171
21.4. Signal Description.....	171
21.5. Product Dependencies.....	171
21.6. Functional Description.....	172
21.7. Register Summary.....	182
21.8. Register Description.....	183
<b>22. OSC32KCTRL – 32KHz Oscillators Controller.....</b>	<b>211</b>

22.1. Overview.....	211
22.2. Features.....	211
22.3. Block Diagram.....	212
22.4. Signal Description.....	212
22.5. Product Dependencies.....	212
22.6. Functional Description.....	214
22.7. Register Summary.....	220
22.8. Register Description.....	220
<b>23. SUPC – Supply Controller.....</b>	<b>237</b>
23.1. Overview.....	237
23.2. Features.....	237
23.3. Block Diagram.....	238
23.4. Signal Description.....	238
23.5. Product Dependencies.....	238
23.6. Functional Description.....	239
23.7. Register Summary.....	244
23.8. Register Description.....	244
<b>24. WDT – Watchdog Timer.....</b>	<b>263</b>
24.1. Overview.....	263
24.2. Features.....	263
24.3. Block Diagram.....	264
24.4. Signal Description.....	264
24.5. Product Dependencies.....	264
24.6. Functional Description.....	265
24.7. Register Summary.....	271
24.8. Register Description.....	271
<b>25. RTC – Real-Time Counter.....</b>	<b>283</b>
25.1. Overview.....	283
25.2. Features.....	283
25.3. Block Diagram.....	283
25.4. Signal Description.....	284
25.5. Product Dependencies.....	284
25.6. Functional Description.....	286
25.7. Register Summary - COUNT32.....	291
25.8. Register Description - COUNT32.....	291
25.9. Register Summary - COUNT16.....	303
25.10. Register Description - COUNT16.....	304
25.11. Register Summary - CLOCK.....	317
25.12. Register Description - CLOCK.....	318
<b>26. DMAC – Direct Memory Access Controller.....</b>	<b>334</b>
26.1. Overview.....	334
26.2. Features.....	334
26.3. Block Diagram.....	336
26.4. Signal Description.....	336
26.5. Product Dependencies.....	336

26.6. Functional Description.....	337
26.7. Register Summary.....	359
26.8. Register Description.....	360
26.9. Register Summary - SRAM.....	392
26.10. Register Description - SRAM.....	392
<b>27. EIC – External Interrupt Controller.....</b>	<b>400</b>
27.1. Overview.....	400
27.2. Features.....	400
27.3. Block Diagram.....	400
27.4. Signal Description.....	401
27.5. Product Dependencies.....	401
27.6. Functional Description.....	402
27.7. Register Summary.....	407
27.8. Register Description.....	408
<b>28. NVMCTRL – Non-Volatile Memory Controller.....</b>	<b>419</b>
28.1. Overview.....	419
28.2. Features.....	419
28.3. Block Diagram.....	419
28.4. Signal Description.....	420
28.5. Product Dependencies.....	420
28.6. Functional Description.....	421
28.7. Register Summary.....	428
28.8. Register Description.....	429
<b>29. PORT - I/O Pin Controller.....</b>	<b>444</b>
29.1. Overview.....	444
29.2. Features.....	444
29.3. Block Diagram.....	445
29.4. Signal Description.....	445
29.5. Product Dependencies.....	445
29.6. Functional Description.....	448
29.7. Register Summary.....	453
29.8. Register Description.....	454
<b>30. EVSYS – Event System.....</b>	<b>474</b>
30.1. Overview.....	474
30.2. Features.....	474
30.3. Block Diagram.....	474
30.4. Signal Description.....	475
30.5. Product Dependencies.....	475
30.6. Functional Description.....	476
30.7. Register Summary.....	480
30.8. Register Description.....	481
<b>31. SERCOM – Serial Communication Interface.....</b>	<b>497</b>
31.1. Overview.....	497
31.2. Features.....	497

31.3. Block Diagram.....	497
31.4. Signal Description.....	497
31.5. Product Dependencies.....	498
31.6. Functional Description.....	499
<b>32. SERCOM USART – SERCOM Universal Synchronous and Asynchronous Receiver and Transmitter.....</b>	<b>505</b>
32.1. Overview.....	505
32.2. USART Features.....	505
32.3. Block Diagram.....	506
32.4. Signal Description.....	506
32.5. Product Dependencies.....	506
32.6. Functional Description.....	508
32.7. Register Summary.....	520
32.8. Register Description.....	520
<b>33. SERCOM SPI – SERCOM Serial Peripheral Interface.....</b>	<b>545</b>
33.1. Overview.....	545
33.2. Features.....	545
33.3. Block Diagram.....	546
33.4. Signal Description.....	546
33.5. Product Dependencies.....	546
33.6. Functional Description.....	548
33.7. Register Summary.....	558
33.8. Register Description.....	559
<b>34. SERCOM I<sup>2</sup>C – SERCOM Inter-Integrated Circuit.....</b>	<b>579</b>
34.1. Overview.....	579
34.2. Features.....	579
34.3. Block Diagram.....	580
34.4. Signal Description.....	580
34.5. Product Dependencies.....	580
34.6. Functional Description.....	582
34.7. Register Summary - I2C Slave.....	601
34.8. Register Description - I2C Slave.....	601
34.9. Register Summary - I2C Master.....	621
34.10. Register Description - I2C Master.....	622
<b>35. CAN - Control Area Network.....</b>	<b>644</b>
35.1. Overview.....	644
35.2. Features.....	644
35.3. Block Diagram.....	644
35.4. Signal Description.....	645
35.5. Product Dependencies.....	645
35.6. Functional Description.....	646
35.7. Register Summary.....	667
35.8. Register Description.....	671
35.9. Message RAM.....	748

36. TC – Timer/Counter.....	758
36.1. Overview.....	758
36.2. Features.....	758
36.3. Block Diagram.....	759
36.4. Signal Description.....	759
36.5. Product Dependencies.....	760
36.6. Functional Description.....	761
36.7. Register Summary.....	775
36.8. Register Description.....	779
37. TCC – Timer/Counter for Control Applications.....	809
37.1. Overview.....	809
37.2. Features.....	809
37.3. Block Diagram.....	810
37.4. Signal Description.....	811
37.5. Product Dependencies.....	811
37.6. Functional Description.....	813
37.7. Register Summary.....	844
37.8. Register Description.....	847
38. CCL – Configurable Custom Logic.....	904
38.1. Overview.....	904
38.2. Features.....	904
38.3. Block Diagram.....	905
38.4. Signal Description.....	905
38.5. Product Dependencies.....	905
38.6. Functional Description.....	907
38.7. Register Summary.....	917
38.8. Register Description.....	918
39. ADC – Analog-to-Digital Converter.....	923
39.1. Overview.....	923
39.2. Features.....	923
39.3. Block Diagram.....	924
39.4. Signal Description.....	924
39.5. Product Dependencies.....	924
39.6. Functional Description.....	926
39.7. Register Summary.....	938
39.8. Register Description.....	939
40. SDADC – Sigma-Delta Analog-to-Digital Converter.....	967
40.1. Overview.....	967
40.2. Features.....	967
40.3. Block Diagram.....	968
40.4. Signal Description.....	968
40.5. Product Dependencies.....	969
40.6. Functional Description.....	970
40.7. Register Summary.....	979

40.8. Register Description.....	980
<b>41. AC – Analog Comparators.....</b>	<b>1006</b>
41.1. Overview.....	1006
41.2. Features.....	1006
41.3. Block Diagram.....	1007
41.4. Signal Description.....	1008
41.5. Product Dependencies.....	1008
41.6. Functional Description.....	1010
41.7. Register Summary.....	1019
41.8. Register Description.....	1019
<b>42. DAC – Digital-to-Analog Converter.....</b>	<b>1037</b>
42.1. Overview.....	1037
42.2. Features.....	1037
42.3. Block Diagram.....	1037
42.4. Signal Description.....	1037
42.5. Product Dependencies.....	1037
42.6. Functional Description.....	1039
42.7. Register Summary.....	1044
42.8. Register Description.....	1044
<b>43. PTC - Peripheral Touch Controller.....</b>	<b>1058</b>
43.1. Overview.....	1058
43.2. Features.....	1058
43.3. Block Diagram.....	1059
43.4. Signal Description.....	1060
43.5. Product Dependencies.....	1060
43.6. Functional Description.....	1061
<b>44. TSENS – Temperature Sensor.....</b>	<b>1063</b>
44.1. Overview.....	1063
44.2. Features.....	1063
44.3. Block Diagram.....	1063
44.4. Signal Description.....	1063
44.5. Product Dependencies.....	1064
44.6. Functional Description.....	1065
44.7. Register Summary.....	1069
44.8. Register Description.....	1069
<b>45. FREQM – Frequency Meter.....</b>	<b>1090</b>
45.1. Overview.....	1090
45.2. Features.....	1090
45.3. Block Diagram.....	1090
45.4. Signal Description.....	1090
45.5. Product Dependencies.....	1090
45.6. Functional Description.....	1092
45.7. Register Summary.....	1095
45.8. Register Description.....	1095

46. Electrical Characteristics.....	1105
46.1. Disclaimer.....	1105
46.2. Absolute Maximum Ratings.....	1105
46.3. General Operating Ratings.....	1106
46.4. Supply Characteristics.....	1106
46.5. Maximum Clock Frequencies.....	1107
46.6. Power Consumption.....	1109
46.7. I/O Pin Characteristics.....	1111
46.8. Analog Characteristics.....	1115
46.9. NVM Characteristics.....	1126
46.10. Oscillator Characteristics.....	1126
46.11. Timing Characteristics.....	1131
47. Packaging Information.....	1132
47.1. Thermal Considerations.....	1132
47.2. Package Drawings.....	1133
47.3. Soldering Profile.....	1139
48. Schematic Checklist.....	1140
48.1. Introduction.....	1140
48.2. Power Supply.....	1140
48.3. External Analog Reference Connections.....	1142
48.4. External Reset Circuit.....	1144
48.5. Unused or Unconnected Pins.....	1145
48.6. Clocks and Crystal Oscillators.....	1145
48.7. Programming and Debug Ports.....	1148
49. Errata.....	1152
49.1. Revision A.....	1152
49.2. Revision B.....	1152
50. Revision History.....	1164
50.1. Rev E - 12/2015.....	1164
50.2. Rev D - 09/2015.....	1164
50.3. Rev C - 09/2015.....	1164
50.4. Rev B - 06/2015.....	1165
50.5. Rev A - 04/2015.....	1165



## 1. Description

The Atmel SAM C21 devices provide the following features: In-system programmable Flash, twelve-channel direct memory access (DMA) controller, twelve-channel Event System, programmable interrupt controller, up to 52 programmable I/O pins, 32-bit real-time clock and calendar, up to five 16-bit Timer/Counters (TC) and three Timer/Counters for Control (TCC), where each TC can be configured to perform frequency and waveform generation, accurate program execution timing or input capture with time and frequency measurement of digital signals. The TCs can operate in 8- or 16-bit mode, selected TCs can be cascaded to form a 32-bit TC, and three timer/counters have extended functions optimized for motor, lighting and other control applications. Two TCC can operate in 24-bit mode, and the third TCC can operate in 16-bit mode. The series provide two Controller Area Network (CAN) modules supported CAN 2.0A/B and CAN-FD 1.0; up to six 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, SMBus, PMBus, RS-485 and LIN master/slave; two 12-bit, 1Msps ADCs with up to 12-channels each (20 unique channels total), three 24-bit, 1.5Msps Sigma-Delta ADC, one 10-bit 300ksps DAC, four analog comparators with window mode, Peripheral Touch Controller supporting up to 256 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 SAM C21 devices have three software-selectable sleep modes, idle, standby and off. 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. In this mode all RAMs and logic contents are retained. The device supports SleepWalking. This feature allows the peripheral to wake up from sleep based on predefined conditions, and thus allows some internal operation like DMA transfer and/or the CPU to wake up only when needed, e.g. when a threshold is crossed or a result is ready. The Event System supports synchronous and asynchronous events, allowing peripherals to receive, react to and send events even in standby mode.

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

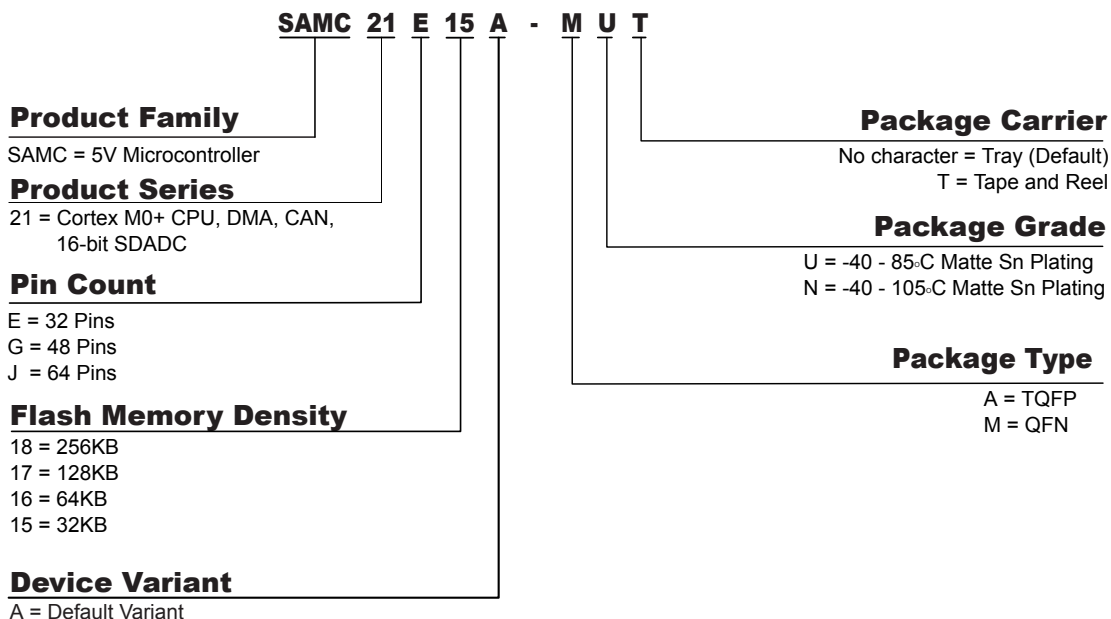
The Atmel SAM C21 devices are supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, programmers and evaluation kits.

## 2. Configuration Summary

	SAM C21J	SAM C21G	SAM C21E
Pins	64	48	32
General Purpose I/O-pins (GPIOs)	52	38	26
Flash	256/128/64/32KB	256/128/64/32KB	256/128/64/32KB
Flash RWW section	8/4/2/1KB	8/4/2/1KB	8/4/2/1KB
System SRAM	32/16/8/4KB	32/16/8/4KB	32/16/8/4KB
Timer Counter (TC) instances	5	5	5
Waveform output channels per TC instance	2	2	2
Timer Counter for Control (TCC) instances	3	3	3
Waveform output channels per TCC	8/4/2	8/4/2	6/4/2
DMA channels	12	12	12
CAN interface	2	2	1
Configurable Custom Logic (CCL) (LUTs)	4	4	4
Serial Communication Interface (SERCOM) instances	6	6	4
Analog-to-Digital Converter (ADC) channels	20	14	10
Sigma-Delta Analog-to-Digital Converter (SDADC) channels	3	2	1
Analog Comparators (AC)	4	4	3
Digital-to-Analog Converter (DAC) channels	1	1	1
Real-Time Counter (RTC)	Yes	Yes	Yes
RTC alarms	1	1	1
RTC compare values	One 32-bit value or two 16-bit values	One 32-bit value or two 16-bit values	One 32-bit value or two 16-bit values
External Interrupt lines	16	16	16
Peripheral Touch Controller (PTC) X and Y lines	16x16	10x12 12x10	6x10 10x6
Maximum CPU frequency	48MHz		

	SAM C21J	SAM C21G	SAM C21E
Packages	QFN TQFP	QFN TQFP	QFN TQFP
Oscillators	32.768kHz crystal oscillator (XOSC32K) 0.4-32MHz crystal oscillator (XOSC) 32.768kHz internal oscillator (OSC32K) 32KHz ultra-low-power internal oscillator (OSCULP32K) 48MHz high-accuracy internal oscillator (OSC48M) 96MHz Fractional Digital Phased Locked Loop (FDPLL96M)		
Event System channels	12	12	12
SW Debug Interface	Yes	Yes	Yes
Watchdog Timer (WDT)	Yes	Yes	Yes

### 3. Ordering Information



#### 3.1. SAM C21E

Table 3-1. SAM C21E15A Ordering Codes

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21E15A-AUT	32K	4K	TQFP32	Tape & Reel	85°C
ATSAM C21E15A-ANT	32K	4K	TQFP32	Tape & Reel	105°C
ATSAM C21E15A-MUT	32K	4K	QFN32	Tape & Reel	85°C
ATSAM C21E15A-MNT	32K	4K	QFN32	Tape & Reel	105°C

Table 3-2. SAM C21E16A Ordering Codes

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21E16A-AUT	64K	8K	TQFP32	Tape & Reel	85°C
ATSAM C21E16A-ANT	64K	8K	TQFP32	Tape & Reel	105°C
ATSAM C21E16A-MUT	64K	8K	QFN32	Tape & Reel	85°C
ATSAM C21E16A-MNT	64K	8K	QFN32	Tape & Reel	105°C

**Table 3-3. SAM C21E17A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21E17A-AUT	128K	16K	TQFP32	Tape & Reel	85°C
ATSAM C21E17A-ANT	128K	16K	TQFP32	Tape & Reel	105°C
ATSAM C21E17A-MUT	128K	16K	QFN32	Tape & Reel	85°C
ATSAM C21E17A-MNT	128K	16K	QFN32	Tape & Reel	105°C

**Table 3-4. SAM C21E18A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21E18A-AUT	256K	32K	TQFP32	Tape & Reel	85°C
ATSAM C21E18A-ANT	256K	32K	TQFP32	Tape & Reel	105°C
ATSAM C21E18A-MUT	256K	32K	QFN32	Tape & Reel	85°C
ATSAM C21E18A-MNT	256K	32K	QFN32	Tape & Reel	105°C

## 3.2. SAM C21G

**Table 3-5. SAM C21G15A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21G15A-AUT	32K	4K	TQFP48	Tape & Reel	85°C
ATSAM C21G15A-ANT	32K	4K	TQFP48	Tape & Reel	105°C
ATSAM C21G15A-MUT	32K	4K	QFN48	Tape & Reel	85°C
ATSAM C21G15A-MNT	32K	4K	QFN48	Tape & Reel	105°C

**Table 3-6. SAM C21G16A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21G16A-AUT	64K	8K	TQFP48	Tape & Reel	85°C
ATSAM C21G16A-ANT	64K	8K	TQFP48	Tape & Reel	105°C
ATSAM C21G16A-MUT	64K	8K	QFN48	Tape & Reel	85°C
ATSAM C21G16A-MNT	64K	8K	QFN48	Tape & Reel	105°C

**Table 3-7. SAM C21G17A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21G17A-AUT	128K	16K	TQFP48	Tape & Reel	85°C
ATSAM C21G17A-ANT	128K	16K	TQFP48	Tape & Reel	105°C
ATSAM C21G17A-MUT	128K	16K	QFN48	Tape & Reel	85°C
ATSAM C21G17A-MNT	128K	16K	QFN48	Tape & Reel	105°C

**Table 3-8. SAM C21G18A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21G18A-AUT	256K	32K	TQFP48	Tape & Reel	85°C
ATSAM C21G18A-ANT	256K	32K	TQFP48	Tape & Reel	105°C
ATSAM C21G18A-MUT	256K	32K	QFN48	Tape & Reel	85°C
ATSAM C21G18A-MNT	256K	32K	QFN48	Tape & Reel	105°C

### 3.3. SAM C21J

**Table 3-9. SAM C21J15A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21J15A-AUT	32K	4K	TQFP64	Tape & Reel	85°C
ATSAM C21J15A-ANT	32K	4K	TQFP64	Tape & Reel	105°C
ATSAM C21J15A-MUT	32K	4K	QFN64	Tape & Reel	85°C
ATSAM C21J15A-MNT	32K	4K	QFN64	Tape & Reel	105°C

**Table 3-10. SAM C21J16A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21J16A-AUT	64K	8K	TQFP64	Tape & Reel	85°C
ATSAM C21J16A-ANT	64K	8K	TQFP64	Tape & Reel	105°C
ATSAM C21J16A-MUT	64K	8K	QFN64	Tape & Reel	85°C
ATSAM C21J16A-MNT	64K	8K	QFN64	Tape & Reel	105°C

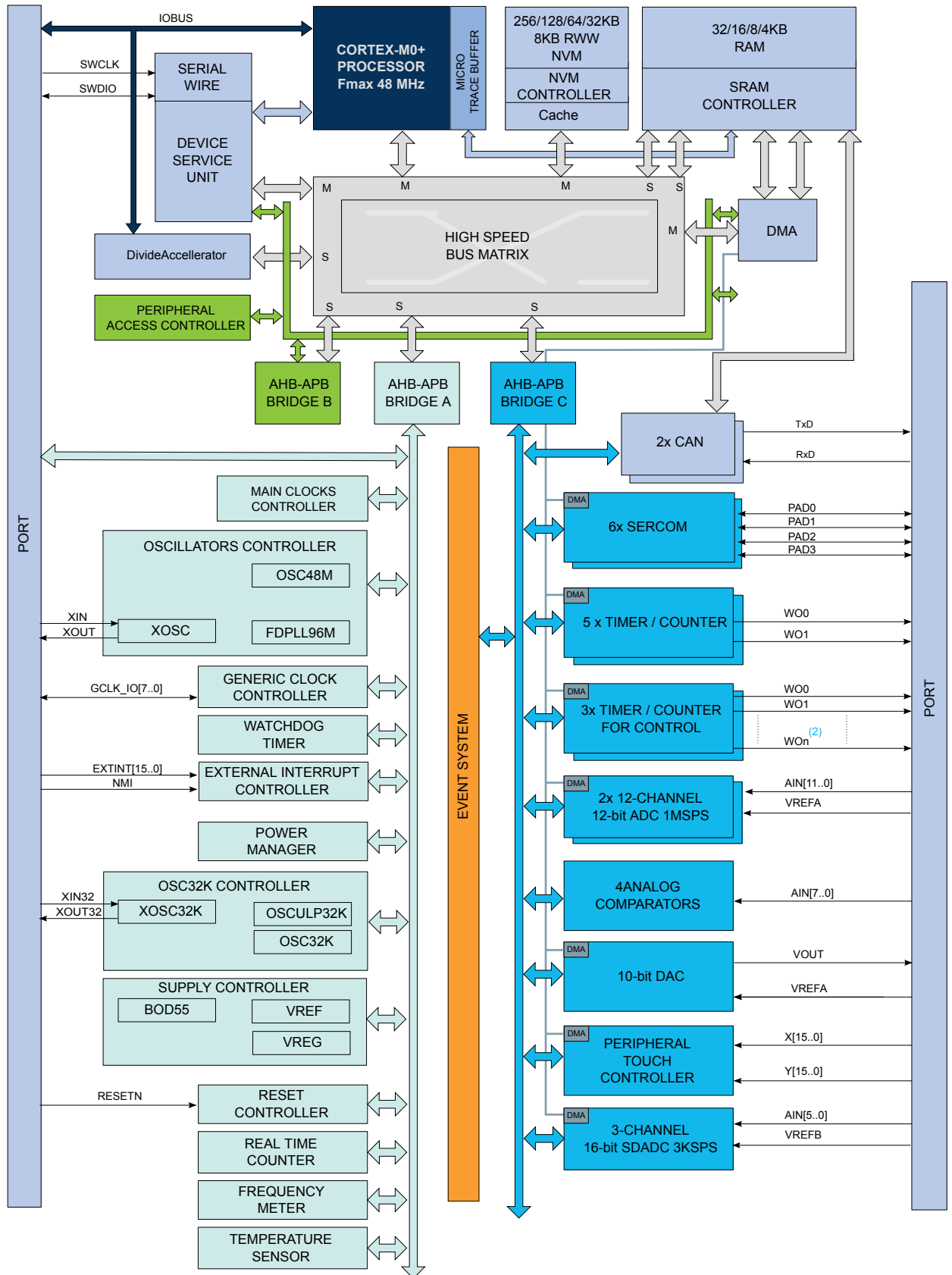
**Table 3-11. SAM C21J17A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21J17A-AUT	128K	16K	TQFP64	Tape & Reel	85°C
ATSAM C21J17A-ANT	128K	16K	TQFP64	Tape & Reel	105°C
ATSAM C21J17A-MUT	128K	16K	QFN64	Tape & Reel	85°C
ATSAM C21J17A-MNT	128K	16K	QFN64	Tape & Reel	105°C

**Table 3-12. SAM C21J18A Ordering Codes**

Ordering Code	FLASH (bytes)	SRAM (bytes)	Package	Carrier Type	Temp
ATSAM C21J18A-AUT	256K	32K	TQFP64	Tape & Reel	85°C
ATSAM C21J18A-ANT	256K	32K	TQFP64	Tape & Reel	105°C
ATSAM C21J18A-MUT	256K	32K	QFN64	Tape & Reel	85°C
ATSAM C21J18A-MNT	256K	32K	QFN64	Tape & Reel	105°C

## 4. Block Diagram



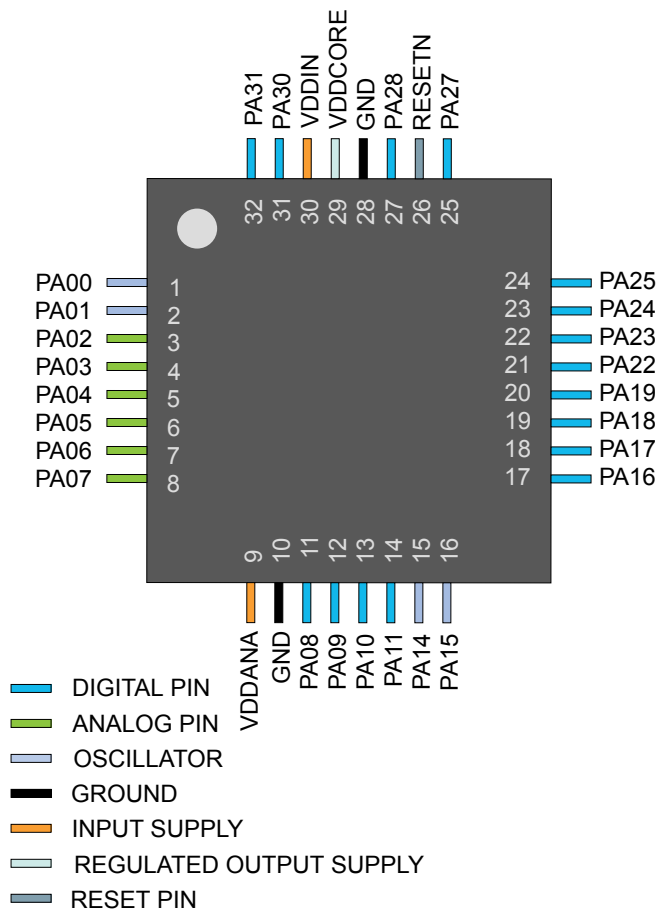


**Note:**

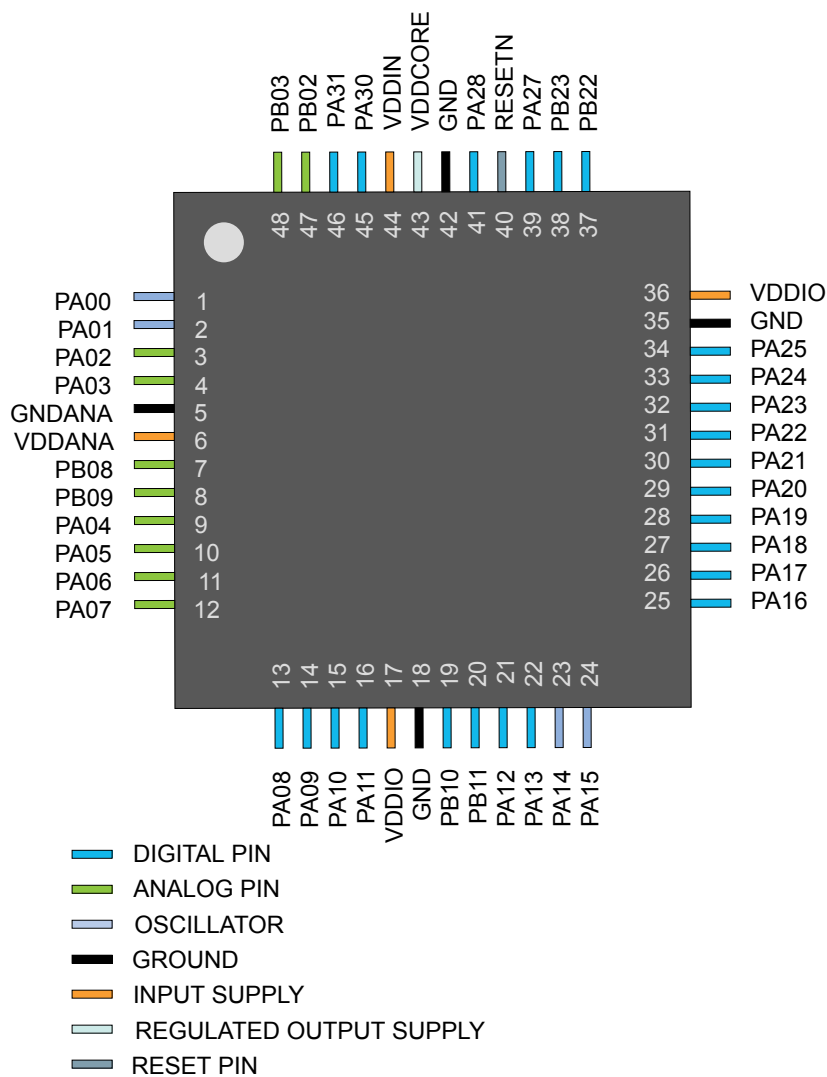
1. Some products have different number of SERCOM instances, Timer/Counter instances, PTC signals and ADC signals. Refer to [Ordering Information](#) on page 16 for details.
2. The three TCC instances have different configurations, including the number of Waveform Output (WO) lines. Refer to TCC CONFIGURATION TABLE for details.

## 5. Pinout

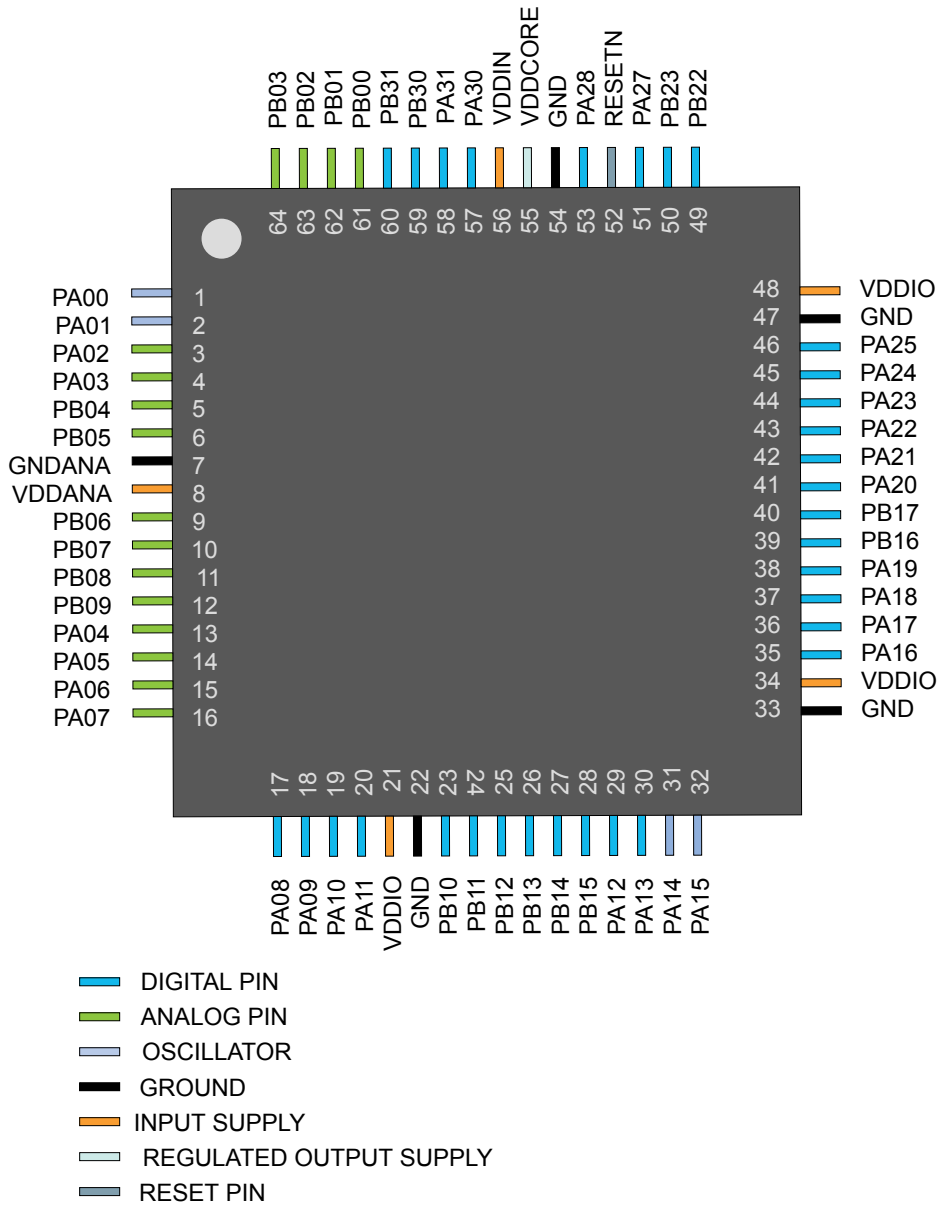
### 5.1. SAM C21E



## 5.2. SAM C21G



### 5.3. SAM C21J



## 6. Signal Descriptions List

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

**Table 6-1. Signal Descriptions List**

Signal Name	Function	Type	Active Level
<b>Analog Comparators - AC</b>			
AIN[7:0]	AC Analog Inputs	Analog	
CMP[2:0]	AC Comparator Outputs	Digital	
<b>Analog Digital Converter - ADCx</b>			
AIN[19:0]	ADC Analog Inputs	Analog	
VREFA	ADC Voltage External Reference A	Analog	
<b>Digital Analog Converter - DAC</b>			
VOUT[1:0]	DAC Voltage output	Analog	
VREFA	DAC Voltage External Reference	Analog	
<b>Sigma-Delta Analog Digital Converter - SDADC</b>			
INN[2:0]	SDADC Analog Negative Inputs	Analog	
INP[2:0]	SDADC Analog Positive Inputs	Analog	
VREFB	SDADC Voltage External Reference B	Analog	
<b>External Interrupt Controller - EIC</b>			
EXTINT[15:0]	External Interrupts inputs	Digital	
NMI	External Non-Maskable Interrupt input	Digital	
<b>Generic Clock Generator - GCLK</b>			
GCLK_IO[7:0]	Generic Clock (source clock inputs or generic clock generator output)	Digital	
<b>Custom Control Logic - CCL</b>			
IN[11:0]	Logic Inputs	Digital	
OUT[3:0]	Logic Outputs	Digital	
<b>Power Manager - PM</b>			
RESETN	Reset input	Digital	Low
<b>Serial Communication Interface - SERCOMx</b>			
PAD[3:0]	SERCOM Inputs/Outputs Pads	Digital	
<b>Oscillators Control - OSCCTRL</b>			
XIN	Crystal or external clock Input	Analog/Digital	
XOUT	Crystal Output	Analog	
<b>32KHz Oscillators Control - OSC32CTRL</b>			
XIN32	32KHz Crystal or external clock Input	Analog/Digital	
XOUT32	32KHz Crystal Output	Analog	
<b>Timer Counter - TCx</b>			
WO[1:0]	Waveform Outputs	Digital	

Signal Name	Function	Type	Active Level
<b>Timer Counter - TCCx</b>			
WO[1:0]	Waveform Outputs	Digital	
<b>Peripheral Touch Controller - PTC</b>			
X[15:0]	PTC Input	Analog	
Y[15:0]	PTC Input	Analog	
<b>General Purpose I/O - PORT</b>			
PA25 - PA00	Parallel I/O Controller I/O Port A	Digital	
PA28 - PA27	Parallel I/O Controller I/O Port A	Digital	
PA31 - PA30	Parallel I/O Controller I/O Port A	Digital	
PB17 - PB00	Parallel I/O Controller I/O Port B	Digital	
PB23 - PB22	Parallel I/O Controller I/O Port B	Digital	
PB31 - PB30	Parallel I/O Controller I/O Port B	Digital	
<b>Controller Area Network - CAN</b>			
TX	CAN Transmit Line	Digital	
RX	CAN Receive Line	Digital	

## 7. I/O Multiplexing and Considerations

### 7.1. Multiplexed Signals

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

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

	Pin		I/O Pin	Supply	Type	A			B <sup>(1)(2)</sup>				C	D	E	F	G	H	I		
SAM C21E	SAM C21G	SAM C21J				EIC	REF	ADC0	ADC1	AC	PTC	DAC	SDADC	SERCOM <sup>(1)(2)</sup> (3)	SERCOM-ALT <sup>(3)</sup>	TC <sup>(4)</sup> /TCC	TCC	COM	AC/GCLK	CCL	
1	1	1	PA00	VDDANA		EXTINT[0]									SERCOM1/ PAD[0]	TCC2/WO[0]				CMP[2]	
2	2	2	PA01	VDDANA		EXTINT[1]									SERCOM1/ PAD[1]	TCC2/WO[1]				CMP[3]	
3	3	3	PA02	VDDANA		EXTINT[2]		AIN[0]		AIN[4]	Y[0]	VOOUT									
4	4	4	PA03	VDDANA		EXTINT[3]	ADC/ VREFA DAC/ VREFA	AIN[1]		AIN[5]	Y[1]										
		5	PB04	VDDANA		EXTINT[4]			AIN[6]		Y[10]										
		6	PB05	VDDANA		EXTINT[5]			AIN[7]	AIN[6]	Y[11]										
		9	PB06	VDDANA		EXTINT[6]			AIN[8]	AIN[7]	Y[12]		INN[2]								CCL2/ IN[6]
		10	PB07	VDDANA		EXTINT[7]			AIN[9]		Y[13]		INP[2]								CCL2/ IN[7]
	7	11	PB08	VDDANA		EXTINT[8]		AIN[2]	AIN[4]		Y[14]		INN[1]		SERCOM4/ PAD[0]	TC0/WO[0]					CCL2/ IN[8]
	8	12	PB09	VDDANA		EXTINT[9]		AIN[3]	AIN[5]		Y[15]		INP[1]		SERCOM4/ PAD[1]	TC0WO[1]					CCL2/ OUT[2]
5	9	13	PA04	VDDANA		EXTINT[4]	SDADC VREFB	AIN[4]		AIN[0]	Y[2]				SERCOM0/ PAD[0]	TCC0/WO[0]					CCL0/ IN[0]
6	10	14	PA05	VDDANA		EXTINT[5]		AIN[5]		AIN[1]	Y[3]				SERCOM0/ PAD[1]	TCC0/WO[1]					CCL0/ IN[1]
7	11	15	PA06	VDDANA		EXTINT[6]		AIN[6]		AIN[2]	Y[4]		INN[0]		SERCOM0/ PAD[2]	TCC1/WO[0]					CCL0/ IN[2]
8	12	16	PA07	VDDANA		EXTINT[7]		AIN[7]		AIN[3]	Y[5]		INP[0]		SERCOM0/ PAD[3]	TCC1/WO[1]					CCL0/ OUT[0]
11	13	17	PA08	VDDIO	I2C	NMI		AIN[8]	AIN[10]		X[0]/Y[16]			SERCOM0/ PAD[0]	SERCOM2/ PAD[0]	TCC0/WO[0]	TCC1/ WO[2]				CCL1/ IN[3]
12	14	18	PA09	VDDIO	I2C	EXTINT[9]		AIN[9]	AIN[11]		X[1]/Y[17]			SERCOM0/ PAD[1]	SERCOM2/ PAD[1]	TCC0/WO[1]	TCC1/ WO[3]				CCL1/ IN[4]
13	15	19	PA10	VDDIO	HS	EXTINT[10]		AIN[10]			X[2]/Y[18]			SERCOM0/ PAD[2]	SERCOM2/ PAD[2]	TCC1/WO[0]	TCC0/ WO[2]		GCLK_IO[4]		CCL1/ IN[5]
14	16	20	PA11	VDDIO	HS	EXTINT[11]		AIN[11]			X[3]/Y[19]			SERCOM0/ PAD[3]	SERCOM2/ PAD[3]	TCC1/WO[1]	TCC0/ WO[3]		GCLK_IO[5]		CCL1/ OUT[1]
		19	PB10	VDDIO	HS	EXTINT[10]									SERCOM4/ PAD[2]	TC1/WO[0]	TCC0/ WO[4]	CAN1/TX	GCLK_IO[4]		CCL1/ IN[5]
		20	PB11	VDDIO	HS	EXTINT[11]									SERCOM4/ PAD[3]	TC1/WO[1]	TCC0/ WO[5]	CAN1/RX	GCLK_IO[5]		CCL1/ OUT[1]
		25	PB12	VDDIO	I2C	EXTINT[12]					X[12]/ Y[28]			SERCOM4/ PAD[0]		TC0/WO[0]	TCC0/ WO[6]			GCLK_IO[6]	
		26	PB13	VDDIO	I2C	EXTINT[13]					X[13]/ Y[29]			SERCOM4/ PAD[1]		TC0/WO[1]	TCC0/ WO[7]			GCLK_IO[7]	
		27	PB14	VDDIO		EXTINT[14]					X[14]/ Y[30]			SERCOM4/ PAD[2]		TC1/WO[0]		CAN1/TX	GCLK_IO[0]		CCL3/ IN[9]
		28	PB15	VDDIO		EXTINT[15]					X[15]/ Y[31]			SERCOM4/ PAD[3]		TC1/WO[1]		CAN1/RX	GCLK_IO[1]		CCL3/ IN[10]
	21	29	PA12	VDDIO	I2C	EXTINT[12]								SERCOM2/ PAD[0]	SERCOM4/ PAD[0]	TCC2/WO[0]	TCC0/ WO[6]			AC/CMP[0]	
	22	30	PA13	VDDIO	I2C	EXTINT[13]								SERCOM2/ PAD[1]	SERCOM4/ PAD[1]	TCC2/WO[1]	TCC0/ WO[7]			AC/CMP[1]	
15	23	31	PA14	VDDIO		EXTINT[14]								SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC4/WO[0]	TCC0/ WO[4]			GCLK_IO[0]	
16	24	32	PA15	VDDIO		EXTINT[15]								SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC4/WO[1]	TCC0/ WO[5]			GCLK_IO[1]	

SAM C21E	Pin	SAM C21G	SAM C21J	I/O Pin	Supply	Type	A							B <sup>(1)(2)</sup>			C		D		E		F		G		H		I	
							REF	ADC0	ADC1	AC	PTC	DAC	SDADC	SERCOM1/ PAD[0]	SERCOM3/ PAD[0]	TC4/TCC	TCC	COM	AC/GCLK	CCL	EIC	REF	ADC0	ADC1	AC	PTC	DAC	SDADC	SERCOM1/ PAD[2]	SERCOM3/ PAD[2]
17	25	35	PA16	VDDIO	I2C	EXTINT[0]							X[4]/Y[20]			SERCOM1/ PAD[0]	SERCOM3/ PAD[0]	TC2/WO[0]	TCC0/ WO[6]							GCLK_IO[2]	CCL0/ IN[0]			
18	26	36	PA17	VDDIO	I2C	EXTINT[1]							X[5]/Y[21]			SERCOM1/ PAD[1]	SERCOM3/ PAD[1]	TC2/WO[1]	TCC0/ WO[7]							GCLK_IO[3]	CCL0/ IN[1]			
19	27	37	PA18	VDDIO		EXTINT[2]							X[6]/Y[22]			SERCOM1/ PAD[2]	SERCOM3/ PAD[2]	TC4/WO[0]	TCC0/ WO[2]							AC/CMP[0]	CCL0/ IN[2]			
20	28	38	PA19	VDDIO		EXTINT[3]							X[7]/Y[23]			SERCOM1/ PAD[3]	SERCOM3/ PAD[3]	TC4/WO[1]	TCC0/ WO[3]							AC/CMP[1]	CCL0/ OUT[0]			
			39	PB16	VDDIO	I2C	EXTINT[0]									SERCOM5/ PAD[0]		TC2/WO[0]	TCC0/ WO[4]							GCLK_IO[2]	CCL3/ IN[11]			
			40	PB17	VDDIO	I2C	EXTINT[1]									SERCOM5/ PAD[1]		TC2/WO[1]	TCC0/ WO[5]							GCLK_IO[3]	CCL3/ OUT[3]			
	29	41	PA20	VDDIO		EXTINT[4]							X[8]/Y[24]			SERCOM5/ PAD[2]	SERCOM3/ PAD[2]	TC3/WO[0]	TCC0/ WO[6]							GCLK_IO[4]				
	30	42	PA21	VDDIO		EXTINT[5]							X[9]/Y[25]			SERCOM5/ PAD[3]	SERCOM3/ PAD[3]	TC3/WO[1]	TCC0/ WO[7]							GCLK_IO[5]				
21	31	43	PA22	VDDIO	I2C	EXTINT[6]							X[10]/Y[26]			SERCOM3/ PAD[0]	SERCOM5/ PAD[0]	TC0/WO[0]	TCC0/ WO[4]							GCLK_IO[6]	CCL2/ IN[0]			
22	32	44	PA23	VDDIO	I2C	EXTINT[7]							X[11]/Y[27]			SERCOM3/ PAD[1]	SERCOM5/ PAD[1]	TC0/WO[1]	TCC0/ WO[5]							GCLK_IO[7]	CCL2/ IN[1]			
23	33	45	PA24	VDDIO		EXTINT[12]										SERCOM3/ PAD[2]	SERCOM5/ PAD[2]	TC1/WO[0]	TCC1/ WO[2]		CAN0/TX		AC/CMP[2]			GCLK_IO[6]	CCL2/ IN[2]			
24	34	46	PA25	VDDIO		EXTINT[13]										SERCOM3/ PAD[3]	SERCOM5/ PAD[3]	TC1/WO[1]	TCC1/ WO[3]		CAN0/RX		AC/CMP[3]			GCLK_IO[7]	CCL2/ OUT			
	37	49	PB22	VDDIN		EXTINT[6]											SERCOM5/ PAD[2]	TC3/WO[0]				CAN0/TX			GCLK_IO[0]	CCL0/ IN[0]				
	38	50	PB23	VDDIN		EXTINT[7]											SERCOM5/ PAD[3]	TC3/WO[1]				CAN0/RX			GCLK_IO[1]	CCL0/ OUT				
25	39	51	PA27	VDDIN		EXTINT[15]																	BRK			GCLK_IO[0]				
27	41	53	PA28	VDDIN		EXTINT[8]																				GCLK_IO[0]				
31	45	57	PA30	VDDIN		EXTINT[10]												SERCOM1/ PAD[2]	TCC1/WO[0]				CORTEX_M0P/ SWCLK			GCLK_IO[0]	CCL1/ IN[0]			
32	46	58	PA31	VDDIN		EXTINT[11]												SERCOM1/ PAD[3]	TCC1/WO[1]				CORTEX_M0P/ SWDIO				CCL1/ OUT			
			59	PB30	VDDIN	I2C	EXTINT[14]											SERCOM5/ PAD[0]	TCC0/WO[0]	TCC1/ WO[2]						AC/CMP[2]				
			60	PB31	VDDIN	I2C	EXTINT[15]											SERCOM5/ PAD[1]	TCC0/WO[1]	TCC1/ WO[3]						AC/CMP[3]				
			61	PB00	VDDANA		EXTINT[0]												SERCOM5/ PAD[2]	TC3/WO[0]								CCL0/ IN[1]		
			62	PB01	VDDANA		EXTINT[1]												SERCOM5/ PAD[3]	TC3/WO[1]								CCL0/ IN[2]		
	47	63	PB02	VDDANA		EXTINT[2]													SERCOM5/ PAD[0]	TC2/WO[0]								CCL0/ OUT		
	48	64	PB03	VDDANA		EXTINT[3]													SERCOM5/ PAD[1]	TC2/WO[1]										

1. All analog pin functions are on peripheral function B. Peripheral function B must be selected to disable the digital control of the pin.
2. Only some pins can be used in SERCOM I2C mode. See the Type column for using a SERCOM pin in I2C mode.
3. SERCOM4 and SERCOM5 not supported on SAM C21E
4. TC2 and TC3 not supported on the SAM C21E/G.
5. Pins of type HS and I2C have higher sink capabilities.

## 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 Oscillators Controller (OSCCTRL) and in the 32K Oscillators Controller (OSC32CTRL).



**Table 7-2. 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**

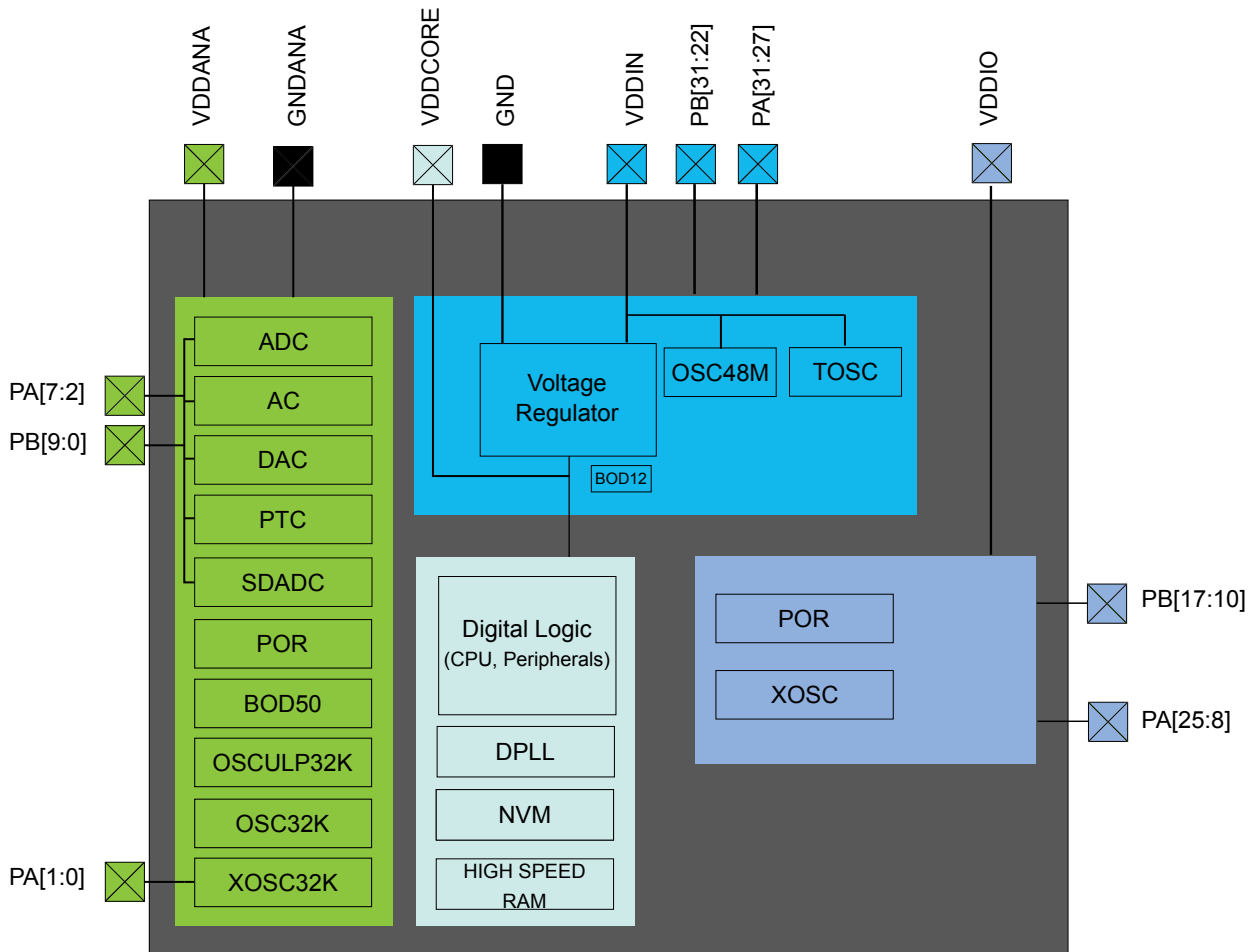
Only the SWCLK pin is mapped to the normal PORT functions. A debugger cold-plugging or hot-plugging detection will automatically switch the SWDIO port to the SWDIO function.

**Table 7-3. Serial Wire Debug Interface Pinout**

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

## 8. Power Supply and Start-Up Considerations

### 8.1. Power Domain Overview



## 8.2. Power Supply Considerations

### 8.2.1. Power Supplies

The SAM C21 has several different power supply pins:

- VDDIO: Powers I/O lines and XOSC. Voltage is 2.70V to 5.50V.
- VDDIN: Powers I/O lines and the OSC48M, TOSC and internal regulator. Voltage is 2.70V to 5.50V.
- VDDANA: Powers I/O lines and the ADC, AC, DAC, PTC, SDADC, OSCULP32K, OSC32K and XOSC32K. Voltage is 2.70V to 5.50V.
- VDDCORE: Internal regulated voltage output. Powers the core, memories, peripherals, and FDPLL96M. Voltage is 1.2V typical.

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

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

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

### 8.2.2. Voltage Regulator

The SAM C21 voltage regulator has two different modes:

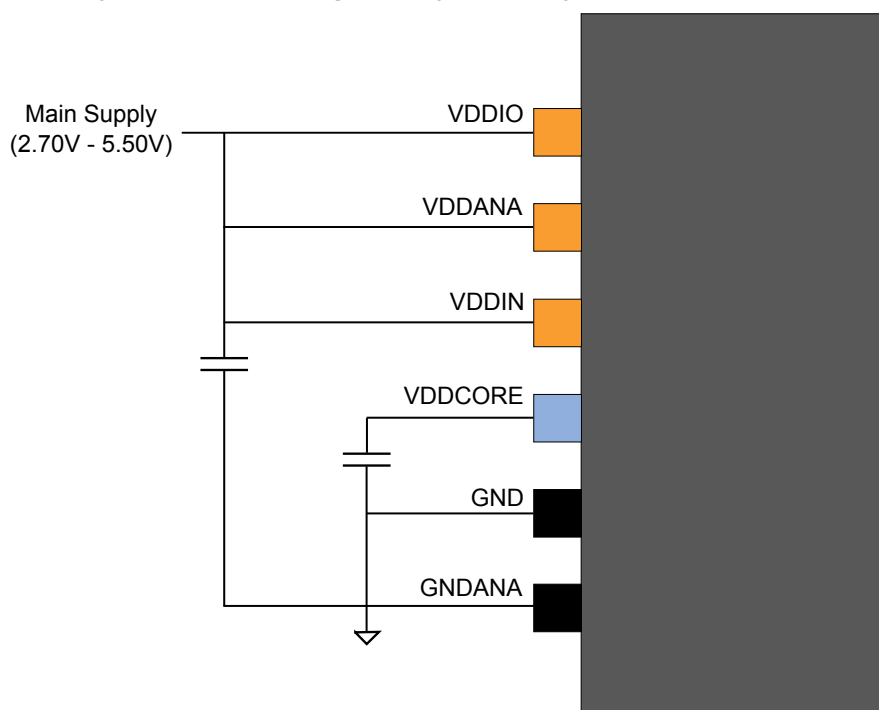
- Normal mode: This is the default mode when CPU and peripherals are running
- Low Power (LP) mode: This is the default mode used when the chip is in standby mode.

### 8.2.3. Typical Powering Schematics

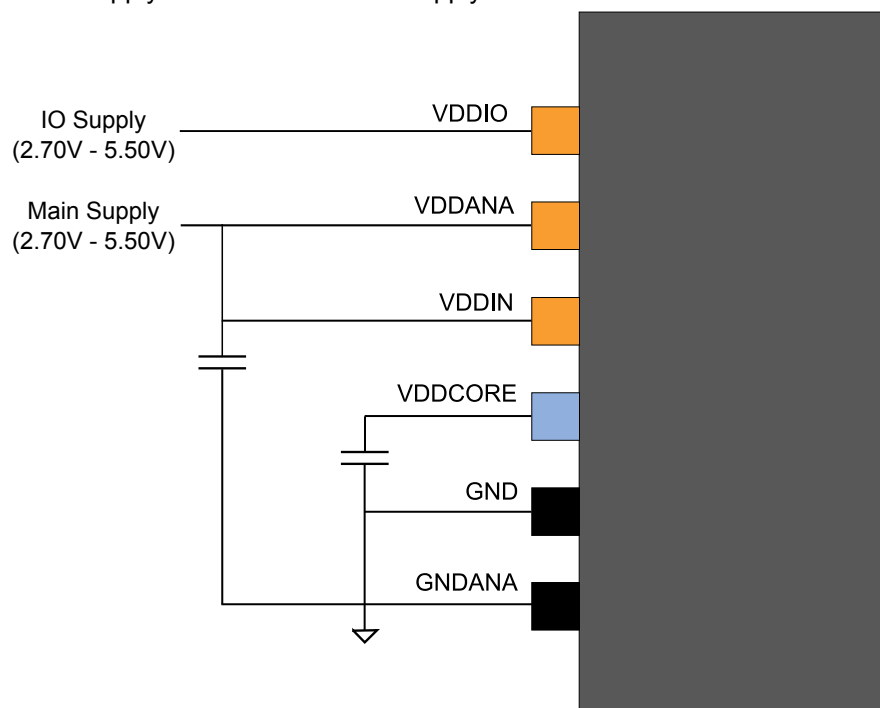
The SAM C21 uses a single supply from 2.70V to 5.50V or dual supply mode where VDDIO is supplied separately from VDDIN.

The following figures show the recommended power supply connections.

**Figure 8-1. Power Supply Connection for single supply mode only**



## Power Supply Connection for dual supply mode



### 8.2.4. Power-Up Sequence

#### 8.2.4.1. Minimum Rise Rate

The integrated power-on reset (POR) circuitry monitoring the VDDIN power supply requires a minimum rise rate.

#### 8.2.4.2. Maximum Rise Rate

The rise rate of the power supply must not exceed the values described in Electrical Characteristics.

## 8.3. Power-Up

This section summarizes the power-up sequence of the SAM C21. The behavior after power-up is controlled by the 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 4MHz clock. This clock is derived from the 48MHz Internal Oscillator (OSC48M), which is configured to provide a 4MHz clock 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” in the Power Manager for the list of default peripheral clocks running. Synchronous system clocks that are running are by default not divided and receive a 4MHz clock through generic clock generator 0. Other generic clocks are disabled.

### 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 the ARM Architecture Reference Manual for more information on CPU startup (<http://www.arm.com>).

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

The SAM C21 embeds three features to monitor, warn and/or reset the device:

- POR: Power-on reset on VDDIN and VDDIO
- BODVDD: Brown-out detector on VDDIN
- BODCORE: 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 BODCORE.

### 8.4.1. Power-On Reset on VDDIN

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

### 8.4.2. Power-On Reset on VDDIO

POR monitors VDDIO. It is always activated and monitors voltage at startup and also during all the sleep modes. If VDDIO goes below the threshold voltage, all IOs supplied by VDDIO are reset.

### 8.4.3. Brown-Out Detector on VDDIN

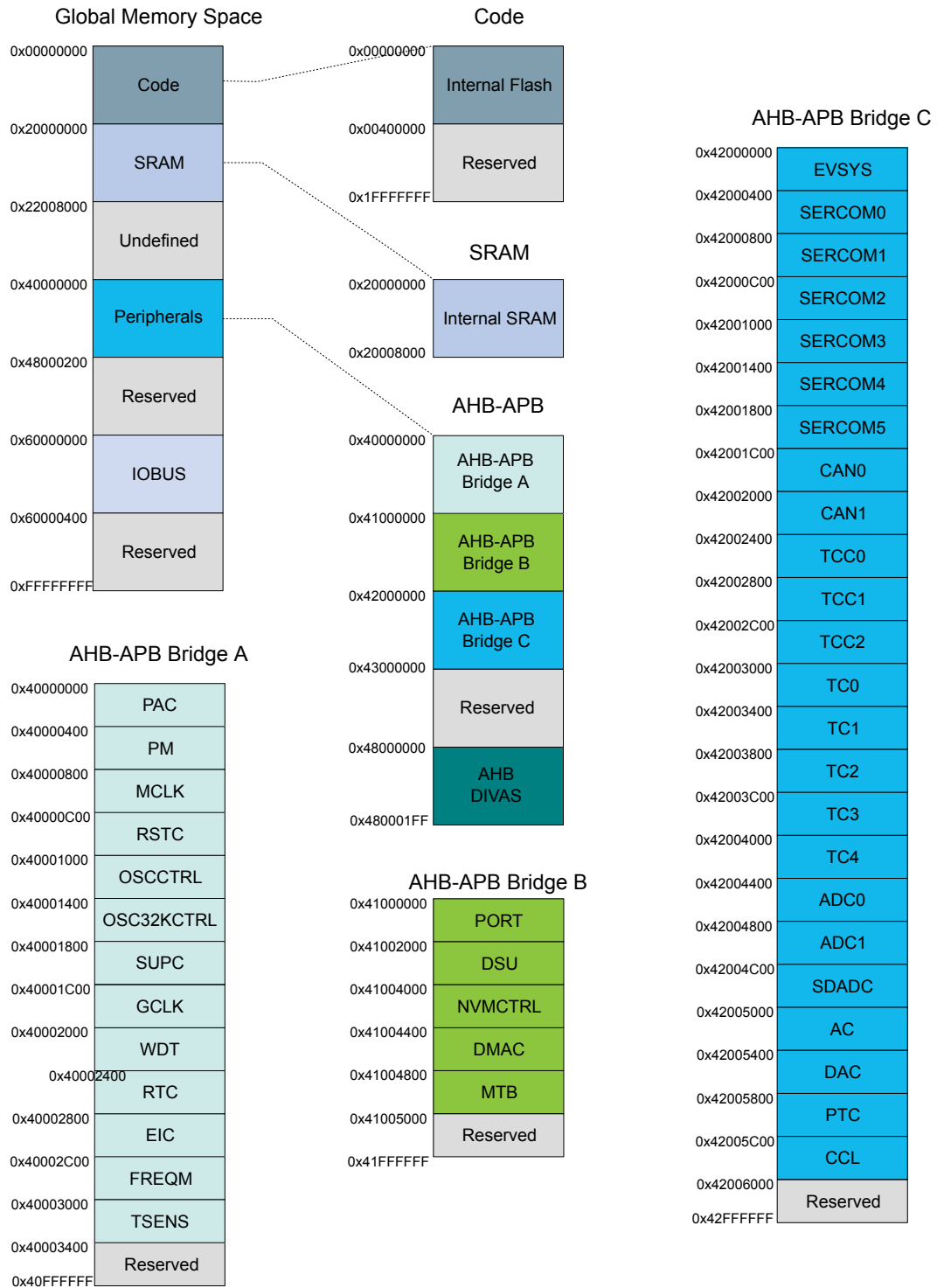
BODVDD monitors VDDIN.

### 8.4.4. Brown-Out Detector on VDDCORE

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

# 9. Product Mapping

Figure 9-1. SAM C21 Product Mapping



## 10. Memories

### 10.1. Embedded Memories

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

### 10.2. Physical Memory Map

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

**Table 10-1. SAM C21 physical memory map<sup>(1)</sup>**

Memory	Start address	Size			
		SAM C21x18	SAM C21x17	SAM C21x16	SAM C21x15
Embedded Flash	0x00000000	256Kbytes	128Kbytes	64Kbytes	32Kbytes
Embedded RWW section	0x00400000	8Kbytes	4Kbytes	2Kbytes	1Kbytes
Embedded high-speed SRAM	0x20000000	32Kbytes	16Kbytes	8Kbytes	4Kbytes
Peripheral Bridge A	0x40000000	64Kbytes	64Kbytes	64Kbytes	64Kbytes
Peripheral Bridge B	0x41000000	64Kbytes	64Kbytes	64Kbytes	64Kbytes
Peripheral Bridge C	0x42000000	64Kbytes	64Kbytes	64Kbytes	64Kbytes

Note: 1. x = G, J or E.

**Table 10-2. Flash memory parameters<sup>(1)</sup>**

Device	Flash size (FLASH_PM)	Number of pages (FLASH_P)	Page size (FLASH_W)
SAM C21x18	256Kbytes	4096	64 bytes
SAM C21x17	128Kbytes	2046	64 bytes
SAM C21x16	64Kbytes	1024	64 bytes
SAM C21x15	32Kbytes	512	64 bytes

Note: 1. x = G, J or E .

**Table 10-3. RWW section parameters<sup>(1)</sup>**

Device	Flash size (FLASH_PM)	Number of pages (FLASH_P)	Page size (FLASH_W)
SAM C21x18	8Kbytes	128	64 bytes
SAM C21x17	4Kbytes	64	64 bytes
SAM C21x16	2Kbytes	32	64 bytes
SAM C21x15	1Kbytes	16	64 bytes

Note: 1. x = G, J or E .

### 10.3. 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 then 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 10-4. NVM User Row Mapping**

Bit Position	Name	Usage
2:0	BOOTPROT	Used to select one of eight different bootloader sizes.
3	Reserved	
6:4	EEPROM	Used to select one of eight different EEPROM sizes.
7	Reserved	
13:8	BODVDD Level	BODVDD Threshold Level at power on.
14	BODVDD Enable	BODVDD Enable at power on.
16:15	BODVDD Action	BODVDD Action at power on.
22:17	BODCORE Level	BODCORE Threshold Level at power on.
23	BODCORE Enable	BODCORE Enable at power on.
25:24	BODCORE Action	BODCORE Action at power on.
26	WDT Enable	WDT Enable at power on.
27	WDT Always-On	WDT Always-On at power on.
31:28	WDT Period	WDT Period at power on.
35:32	WDT Window	WDT Window mode time-out at power on.
39:36	WDT EWOFFSET	WDT Early Warning Interrupt Time Offset at power on.
40	WDT WEN	WDT Timer Window Mode Enable at power on.
41	BODVDD Hysteresis	BODVDD Hysteresis configuration at power on.
42	BODCORE Hysteresis	BODCORE Hysteresis configuration at power on.
47:43	Reserved	
63:48	LOCK	NVM Region Lock Bits.

#### Related Links

[NVMCTRL – Non-Volatile Memory Controller](#) on page 419

[CTRLA](#) on page 272

[CONFIG](#) on page 274

[EWCTRL](#) on page 276



## 10.4. 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 10-5. NVM Software Calibration Area Mapping**

Bit Position	Name	Description
2:0	ADC0 LINEARITY	ADC0 Linearity Calibration. Should be written to the CALIB register.
5:3	ADC0 BIASCAL	ADC0 Bias Calibration. Should be written to the CALIB register.
8:6	ADC1 LINEARITY	ADC1 Linearity Calibration. Should be written to the CALIB register.
11:9	ADC1 BIASCAL	ADC1 Bias Calibration. Should be written to the CALIB register.
18:12	OSC32K CAL	OSC32K Calibration. Should be written to OSC32K register.
63:19	Reserved	

## 10.5. NVM Temperature Calibration Area Mapping

The NVM Temperature 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 Temperature Calibration Area can be read at address 0x806030.

The NVM Temperature Calibration Area can not be written.

**Table 10-6. NVM Software Calibration Area Mapping**

Bit Position	Name	Description
5:0	TSENS TCAL	TSENS Temperature Calibration. Should be written to the TSENS CAL register.
11:6	TSENS FCAL	TSENS Frequency Calibration. Should be written to the TSENS CAL register.
35:12	TSENS GAIN	TSENS Gain Calibration. Should be written to the TSENS GAIN register.
59:36	TSENS OFFSET	TSENS Offset Calibration. Should be written to TSENS OFFSET register.
63:60	Reserved	

## 10.6. Serial Number

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

Word 0: 0x0080A00C

Word 1: 0x0080A040

Word 2: 0x0080A044

Word 3: 0x0080A048

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

# 11. Processor and Architecture

## 11.1. Cortex M0+ Processor

The Atmel SAM C21 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 implemented ARM Cortex-M0+ is revision r0p1. For more information refer to <http://www.arm.com>.

### 11.1.1. Cortex M0+ Configuration

Table 11-1. Cortex M0+ Configuration

Features	Cortex-M0+ options	SAM C21 configuration
Interrupts	External interrupts 0-32	32
Data endianness	Little-endian or big-endian	Little-endian
SysTick timer	Present or absent	Present
Number of watchpoint comparators	0, 1, 2	2
Number of breakpoint comparators	0, 1, 2, 3, 4	4
Halting debug support	Present or absent	Present
Multiplier	Fast or small	Fast (single cycle)
Single-cycle I/O port	Present or absent	Present
Wake-up interrupt controller	Supported or not supported	Not supported
Vector Table Offset Register	Present or absent	Present
Unprivileged/Privileged support	Present or absent	Present
Memory Protection Unit	Not present or 8-region	8-region
Reset all registers	Present or absent	Absent
Instruction fetch width	16-bit only or mostly 32-bit	32-bit

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

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

### 11.1.2. 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 (<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](#) on page 41 and the Cortex-M0+ Technical Reference Manual for details (<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 (<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 (<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](#) on page 42 and the CoreSight MTB-M0+ Technical Reference Manual for details (<http://www.arm.com>).
- Memory Protection Unit (MPU)
  - The Memory Protection Unit divides the memory map into a number of regions, and defines the location, size, access permissions and memory attributes of each region. Refer to the Cortex-M0+ Devices Generic User Guide for details (<http://www.arm.com>).

### 11.1.3. Cortex-M0+ Address Map

**Table 11-2. 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)
0x41008000	Micro Trace Buffer (MTB)

#### Related Links

[Product Mapping](#) on page 34

### 11.1.4. I/O Interface

#### 11.1.4.1. 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.

#### Related Links

[CPU Local Bus](#) on page 447

#### 11.1.4.2. Description

Direct access to PORT registers and DIVAS registers.

## 11.2. Nested Vector Interrupt Controller

### 11.2.1. Overview

The Nested Vectored Interrupt Controller (NVIC) in the SAM C21 supports 32 interrupt lines with four different priority levels. For more details, refer to the Cortex-M0+ Technical Reference Manual (<http://www.arm.com>).

### 11.2.2. Interrupt Line Mapping

Each of the 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 11-3. Interrupt Line Mapping**

Peripheral Source	NVIC Line
EIC NMI – External Interrupt Controller	NMI
PM – Power Manager MCLK - Main Clock OSCCTRL - Oscillators Controller OSC32KCTRL - 32kHz Oscillators Controller SUPC - Supply Controller PAC - Protection Access Controller	0
WDT – Watchdog Timer	1
RTC – Real Time Clock	2
EIC – External Interrupt Controller	3
FREQM – Frequency Meter	4
TSENS – Temperature Sensor	5
NVMCTRL – Non-Volatile Memory Controller	6
DMAC - Direct Memory Access Controller	7

Peripheral Source	NVIC Line
EVSYS – Event System	8
SERCOM0 – Serial Communication Controller 0	9
SERCOM1 – Serial Communication Controller 1	10
SERCOM2 – Serial Communication Controller 2	11
SERCOM3 – Serial Communication Controller 3	12
SERCOM4 – Serial Communication Controller 4	13
SERCOM5 – Serial Communication Controller 5	14
CAN0 – Controller Area Network 0	15
CAN1 – Controller Area Network 1	16
TCC0 – Timer Counter for Control 0	17
TCC1 – Timer Counter for Control 1	18
TCC2 – Timer Counter for Control 2	19
TC0 – Timer Counter 0	20
TC1 – Timer Counter 2	21
TC2 – Timer Counter 2	22
TC3 – Timer Counter 3	23
TC4 – Timer Counter 4	24
ADC0 – Analog-to-Digital Converter 0	25
ADC1 – Analog-to-Digital Converter 1	26
AC – Analog Comparator	27
DAC – Digital-to-Analog Converter	28
SDADC – Sigma-Delta Analog-to-Digital Converter 1	29
PTC – Peripheral Touch Controller	30
Reserved	31

## 11.3. Micro Trace Buffer

### 11.3.1. 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

### 11.3.2. 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.

## 11.4. High-Speed Bus System

### 11.4.1. 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 1-to-1 clock frequency with the bus masters

## 11.4.2. Configuration

Figure 11-1. Master-Slave Relation High-Speed Bus Matrix

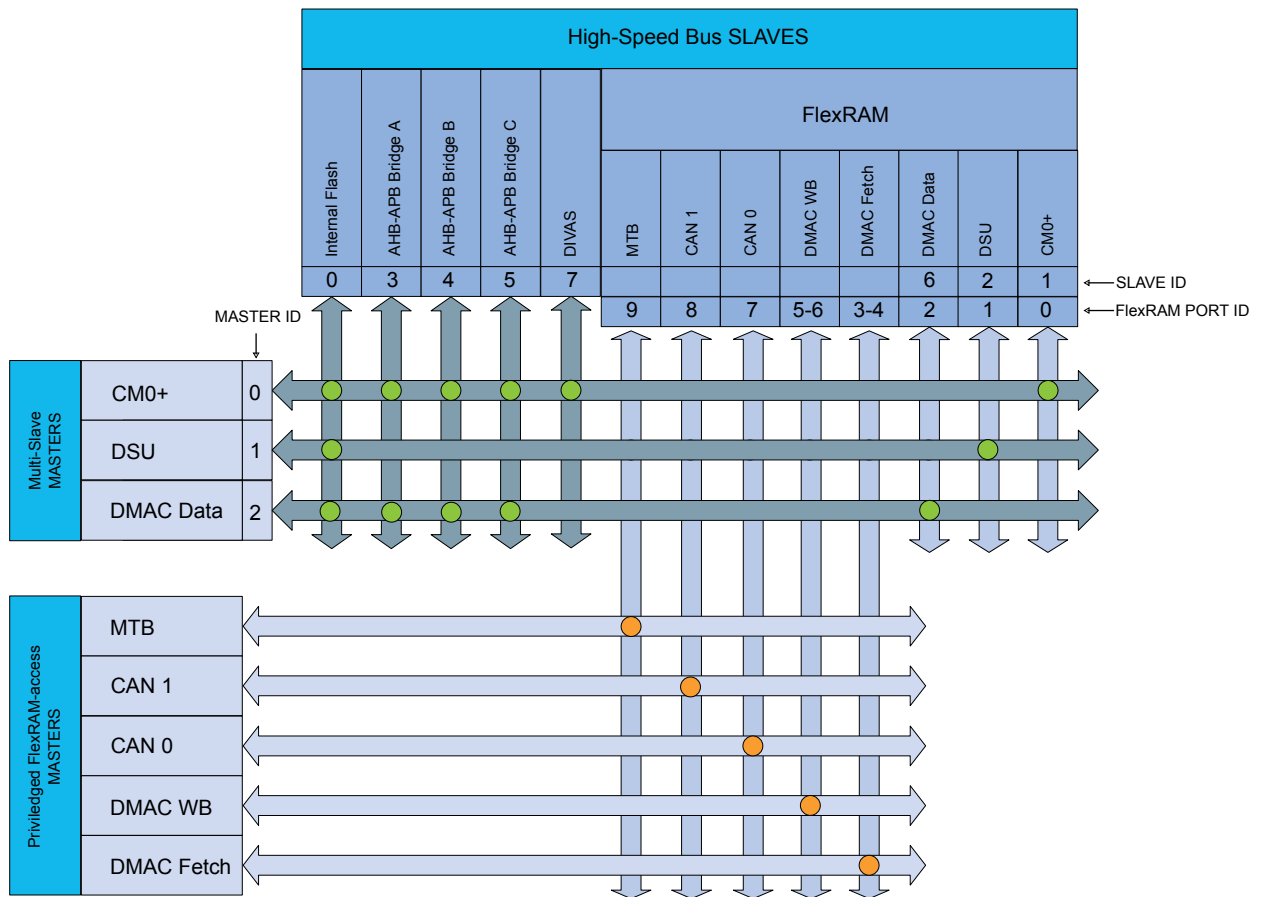


Table 11-4. Bus Matrix Masters

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

Table 11-5. Bus Matrix Slaves

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



Bus Matrix Slaves	Slave ID
SRAM Port 5 - DMAC Data Access	6
DIVAS - Divide Accelerator	7

**Table 11-6. SRAM Port Connections**

SRAM Port Connection	Port ID	Connection Type
CM0+ - Cortex M0+ Processor	0	Bus Matrix
DSU - Device Service Unit	1	Bus Matrix
DMAC - Direct Memory Access Controller - Data Access	2	Bus Matrix
DMAC - Direct Memory Access Controller - Fetch Access 0	3	Direct
DMAC - Direct Memory Access Controller - Fetch Access 1	4	Direct
DMAC - Direct Memory Access Controller - Write-Back Access 0	5	Direct
DMAC - Direct Memory Access Controller - Write-Back Access 1	6	Direct
CAN0 - Controller Area Network 0	7	Direct
CAN1 - Controller Area Network 1	8	Direct
MTB - Micro Trace Buffer	9	Direct

#### 11.4.3. 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 below.

**Table 11-7. Quality of Service Level Configuration**

Value	Name	Description
0x0	DISABLE	Background (no sensitive operation)
0x1	LOW	Sensitive Bandwidth
0x2	MEDIUM	Sensitive Latency
0x3	HIGH	Critical Latency

If a master is configured with QoS level DISABLE (0x0) or LOW (0x1) there will be minimum latency of one cycle for the RAM access.

The priority order for concurrent accesses are decided by two factors. First, the QoS level for the master and second, a static priority given by [Table 11-6 SRAM Port Connections](#) on page 45. The lowest port ID has the highest static priority.

The MTB has fixed QoS level HIGH (0x3) and the DSU has fixed QoS level LOW (0x1).

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 masters (CAN, DMAC).

## 12. PAC - Peripheral Access Controller

### 12.1. Overview

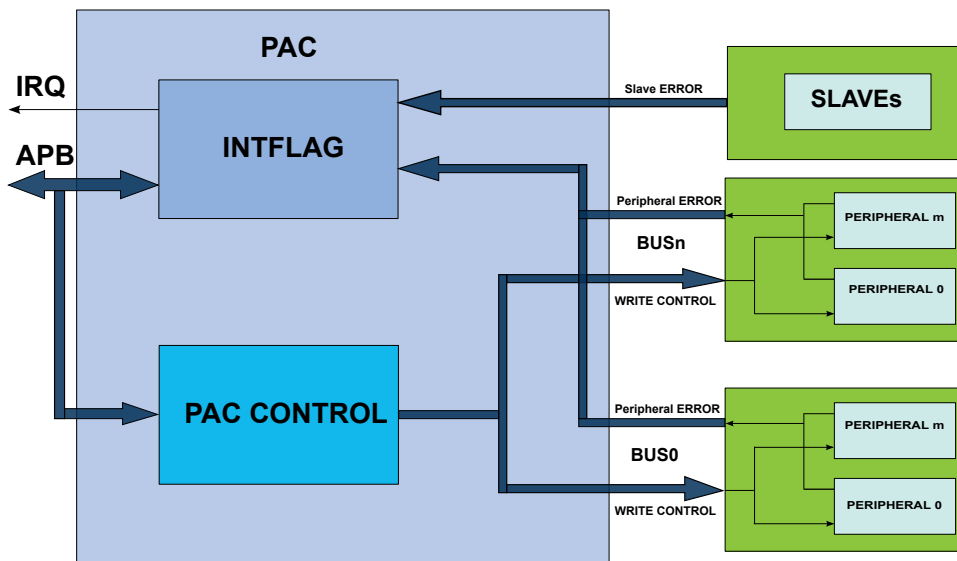
The Peripheral Access Controller provides an interface for the locking and unlocking of peripheral registers within the device. It reports all violations that could happen when accessing a peripheral: write protected access, illegal access, enable protected access, access when clock synchronization or software reset is on-going. These errors are reported in a unique interrupt flag for a peripheral. The PAC module also reports errors occurring at the slave bus level, when an access to a non-existing address is detected.

### 12.2. Features

- Manages write protection access and reports access errors for the peripheral modules or bridges

### 12.3. Block Diagram

Figure 12-1. PAC Block Diagram



### 12.4. Product Dependencies

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

#### 12.4.1. IO Lines

Not applicable.

#### 12.4.2. Power Management

The PAC can continue to operate in any sleep mode where the selected source clock is running. The PAC 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](#) on page 162

### 12.4.3. Clocks

The PAC bus clock (CLK\_PAC\_APB) can be enabled and disabled in the Main Clock module. The default state of CLK\_PAC\_APB can be found in the related links.

#### Related Links

[MCLK – Main Clock](#) on page 131

[Peripheral Clock Masking](#) on page 135

### 12.4.4. DMA

Not applicable.

### 12.4.5. Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the PAC interrupt requires the Interrupt Controller to be configured first.

**Table 12-1. Interrupt Lines**

Instances	NVIC Line
PAC	PACERR

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

### 12.4.6. Events

The events are connected to the Event System, which may need configuration.

#### Related Links

[EVSYS – Event System](#) on page 474

### 12.4.7. Debug Operation

When the CPU is halted in debug mode, write protection of all peripherals is disabled and the PAC continues normal operation.

### 12.4.8. Register Access Protection

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

- Write Control (WRCTRL) register
- AHB Slave Bus Interrupt Flag Status and Clear (INTFLAGAHB) register
- Peripheral Interrupt Flag Status and Clear n (INTFLAG A/B/C...) registers

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.

## 12.5. Functional Description

### 12.5.1. Principle of Operation

The Peripheral Access Control module allows the user to set a write protection on peripheral modules and generate an interrupt in case of a peripheral access violation. The peripheral's protection can be set,

cleared or locked for user convenience. A set of Interrupt Flag and Status registers informs the user on the status of the violation in the peripherals. In addition, slaves bus errors can be also reported in the cases where reserved area is accessed by the application.

## 12.5.2. Basic Operation

### 12.5.2.1. Initialization

After reset, the PAC is enabled.

### 12.5.2.2. Enabling and Resetting

The PAC is always enabled after reset.

Only a hardware reset will reset the PAC module.

### 12.5.2.3. Operations

The PAC module allows the user to set, clear or lock the write protection status of all peripherals on all Peripheral Bridges.

If a peripheral register violation occurs, the Peripheral Interrupt Flag *n* registers (INTFLAG<sub>*n*</sub>) are updated to inform the user on the status of the violation in the peripherals connected to the Peripheral Bridge *n* (*n* = A,B,C ...). The corresponding Peripheral Write Control Status *n* register (STATUS<sub>*n*</sub>) gives the state of the write protection for all peripherals connected to the corresponding Peripheral Bridge *n*. Refer to the [Peripheral Access Errors](#) on page 48 for details.

The PAC module reports also the errors occurring at slave bus level when an access to reserved area is detected. AHB Slave Bus Interrupt Flag register (INTFLAGAHB) informs the user on the status of the violation in the corresponding slave. Refer to the [AHB Slave Bus Errors](#) on page 49 for details.

### 12.5.2.4. Peripheral Access Errors

The following events will generate a Peripheral Access Error:

- Protected write: To avoid unexpected writes to a peripheral's registers, each peripheral can be write protected. Only the registers denoted as "Write Protected" in the module's datasheet will be protected. If a peripheral is not write protected, write data accesses are performed normally. If a peripheral is write protected and if a write access is attempted, data will not be written and peripheral returns an access error. The corresponding interrupt flag bit in the INTFLAG<sub>*n*</sub> register will be set.
- Illegal access: Access to an unimplemented register within the module.
- Synchronized write error: For write-synchronized registers an error will be reported if the register is written while a synchronization is ongoing.

When any of the INTFLAG<sub>*n*</sub> registers bit are set, an interrupt will be requested if the PAC interrupt enable bit is set.

### 12.5.2.5. Write Access Protection Management

Peripheral access control can be enabled or disabled by writing to the WRCTRL register.

The data written to the WRCTRL register is composed of two fields; WRCTRL.PERID and WRCTRL.KEY. The WRCTRL.PERID is an unique identifier corresponding to a peripheral. The WRCTRL.KEY is a key value that defines the operation to be done on the control access bit. These operations can be "clear protection", "set protection" and "set and lock protection bit".

The "clear protection" operation will remove the write access protection for the peripheral selected by WRCTRL.PERID. Write accesses are allowed for the registers in this peripheral.

The "set protection" operation will set the write access protection for the peripheral selected by WRCTRL.PERID. Write accesses are not allowed for the registers with write protection property in this peripheral.

The “set and lock protection” operation will permanently set the write access protection for the peripheral selected by WRCTRL.PERID. The write access protection will only be cleared by a hardware reset.

The peripheral access control status can be read from the corresponding STATUSn register.

#### 12.5.2.6. Write Access Protection Management Errors

Only word-wise writes to the WRCTRL register will effectively change the access protection. Other type of accesses will have no effect and will cause a PAC write access error. This error is reported in the INTFLAGn.PAC bit corresponding to the PAC module.

PAC also offers an additional safety feature for correct program execution with an interrupt generated on double write clear protection or double write set protection. If a peripheral is write protected and a subsequent set protection operation is detected then the PAC returns an error, and similarly for a double clear protection operation. In addition, an error is generated when writing a “set and lock” protection to a write-protected peripheral or when a write access is done to a locked set protection.

This can be used to ensure that the application follows the intended program flow by always following a write protect with an unprotect and conversely. However in applications where a write protected peripheral is used in several contexts, e.g. interrupt, care should be taken so that either the interrupt can not happen while the main application or other interrupt levels manipulates the write protection status or when the interrupt handler needs to unprotect the peripheral based on the current protection status by reading the STATUS register.

The errors generated while accessing the PAC module registers (eg. key error, double protect error...) will set the INTFLAGn.PAC flag.

#### 12.5.2.7. AHB Slave Bus Errors

The PAC module reports errors occurring at the Slave bus level. These errors are generated when an access is performed at an address where no slave (bridge or peripheral) is mapped. These errors are reported in the INTFLAGAHB register.

#### 12.5.2.8. Generating Events

The PAC module can also generate an event when any of the Interrupt Flag registers bit are set. To enable the PAC event generation, the control bit EVCTRL.ERREO must be set.

#### 12.5.3. DMA Operation

Not applicable.

#### 12.5.4. Interrupts

The PAC has the following interrupt source:

- Error (ERR): Indicates that a peripheral access violation occurred in one of the peripherals controlled by the PAC module, or a bridge error occurred in one of the bridges reported by the PAC
  - This interrupt is a synchronous wake-up source.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAGAHB and INTFLAGn) registers 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 PAC is reset. 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 INTFLAGAHB and INTFLAGn registers 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](#) on page 41

[Sleep Mode Controller](#) on page 164

### 12.5.5. Events

The PAC can generate the following output event:

- Error (ERR): Generated when one of the interrupt flag registers bits is set

Writing a one to an Event Output bit in the Event Control Register (EVCTRL.ERREO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event.

### 12.5.6. Sleep Mode Operation

In Sleep mode, the PAC is kept enabled if an available master (CPU, DMA) is running. The PAC will continue to catch access errors from module and generate interrupts or events.

### 12.5.7. Synchronization

Not applicable.

## 12.6. Register Summary

Offset	Name	Bit Pos.								
0x00	WRCTRL	7:0	PERID[7:0]							
0x01		15:8	PERID[15:8]							
0x02		23:16	KEY[7:0]							
0x03		31:24								
0x04	EVCTRL	7:0							ERREO	
0x05	Reserved									
...										
0x07										
0x08	INTENCLR	7:0							ERR	
0x09	INTENSET	7:0							ERR	

## 12.7. 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 the related links.

### 12.7.1. Write Control

**Name:** WRCTRL  
**Offset:** 0x0  
**Reset:** 0x00000000  
**Property:** –

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	KEY[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
	PERID[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
	PERID[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 – KEY[7:0]: Peripheral Access Control Key

These bits define the peripheral access control key:

Value	Name	Description
0x0	OFF	No action
0x1	CLEAR	Clear the peripheral write control
0x2	SET	Set the peripheral write control
0x3	LOCK	Set and lock until the next hardware reset the peripheral write control

#### Bits 15:0 – PERID[15:0]: Peripheral Identifier

The PERID represents the peripheral whose control is changed using the WRCTRL.KEY. The Peripheral Identifier is calculated following formula:

$$PERID = 32 * BridgeNumber + N$$

Where BridgeNumber represents the Peripheral Bridge Number (0 for Peripheral Bridge A, 1 for Peripheral Bridge B, etc). N represents the peripheral index from the respective Bridge Number:



**Table 12-2. PERID Values**

Periph. Bridge Name	BridgeNumber	PERID Values
A	0	0+N
B	1	32+N
C	2	64+N
D	3	96+N
E	4	128+N

## 12.7.2. Event Control

**Name:** EVCTRL  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

### Bit 0 – ERREO: Peripheral Access Error Event Output

This bit indicates if the Peripheral Access Error Event Output is enabled or not. When enabled, an event will be generated when one of the interrupt flag registers bits (INTFLAGAHB, INTFLAGn) is set:

Value	Description
0	Peripheral Access Error Event Output is disabled.
1	Peripheral Access Error Event Output is enabled.

### 12.7.3. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x08

**Reset:** 0x00

**Property:** PAC Write-Protection

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

#### Bit 0 – ERR: Peripheral Access Error Interrupt Enable

This bit indicates that the Peripheral Access Error Interrupt is enabled and an interrupt request will be generated when one of the interrupt flag registers bits (INTFLAGAHB, INTFLAGn) is set:

Writing a zero to this bit has no effect.

Writing a one to this bit will clear the Peripheral Access Error interrupt Enable bit and disables the corresponding interrupt request.

Value	Description
0	Peripheral Access Error interrupt is disabled.
1	Peripheral Access Error interrupt is enabled.

#### 12.7.4. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x09

**Reset:** 0x00

**Property:** PAC Write-Protection

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

##### Bit 0 – ERR: Peripheral Access Error Interrupt Enable

This bit indicates that the Peripheral Access Error Interrupt is enabled and an interrupt request will be generated when one of the interrupt flag registers bits (INTFLAGAHB, INTFLAGn) is set:

Writing a zero to this bit has no effect.

Writing a one to this bit will set the Peripheral Access Error interrupt Enable bit and enables the corresponding interrupt request.

Value	Description
0	Peripheral Access Error interrupt is disabled.
1	Peripheral Access Error interrupt is enabled.

## 13. Peripherals Configuration Summary

Table 13-1. Peripherals Configuration Summary SAMCxx J, G, E

Peripheral 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	
AHB-APB Bridge A	0x40000000		0	Y									N/A
PAC	0x44000000	0	10	Y	0	Y		0	N		85 : ACCERR		N/A
PM	0x40000400	0			1	Y		1	N				N/A
MCLK	0x40000800	0			2	Y		2	N				Y
RSTC	0x40000C00				3	Y		3	N				N/A
OSCCTRL	0x40001000	0			4	Y	0: FDPLL96M clk source 1: FDPLL96M 32kHz	4	N		0: XOSC_FAIL		Y
OSC32KCTRL	0x40001400	0			5	Y		5	N		1: XOSC32K_FAIL		Y
SUPC	0x40001800	0			6	Y		6	N				N/A
GCLK	0x40001C00				7	Y		7	N				N/A
WDT	0x40002000	1			8	Y		8	N				Y
RTC	0x40002400	2			9	Y		9	N		2: CMP0/ALARM0 3: CMP1 4: OVF 5-12: PER0-7		Y
EIC	0x40002800	3, NMI			10	Y	2	10	N		13-28: EXTINT0-15		Y
FREQM	0x40002C00	4			11	Y	3: Measure 4: Reference	11	N				N/A
TSENS	0x40003000	5			12	N	5	12	N	0: START	29: WINMON	1: RESRDY	A
AHB-APB Bridge B	0x41000000		1	Y									N/A
PORT	0x41000000				0	Y		0	N	1-4 : EV0-3			Y
DSU	0x41002000		3	Y	1	Y		1	Y				N/A
NVMCTRL	0x41004000	6	5	Y	2	Y	39	2	N				Y
DMAC	0x41006000	7	7	Y				3	N	5-8: CH0-3	30-33: CH0-3		Y
MTB	0x41008000								N	44: START 45: STOP			N/A
AHB-APB Bridge C	0x42000000		2	Y									N/A
EVSYS	0x42000000	8			0	N	6-17: one per CHANNEL	0	N				Y
SERCOM0	0x42000400	9			1	N	19: CORE 18: SLOW	1	N			2: RX 3: TX	Y
SERCOM1	0x42000800	10			2	N	20: CORE 18: SLOW	2	N			4: RX 5: TX	Y
SERCOM2	0x42000C00	11			3	N	21: CORE 18: SLOW	3	N			6: RX 7: TX	Y
SERCOM3	0x42001000	12			4	N	22: CORE 18: SLOW	4	N			8: RX 9: TX	Y
SERCOM4	0x42001400	13			5	N	23: CORE 18: SLOW	5	N			10: RX 11: TX	Y
SERCOM5	0x42001800	14			6	N	25: CORE 24: SLOW	6	N			12: RX 13: TX	Y
CAN0	0x42001C00	15	8	N			26					14: DEBUG	N/A
CAN1	0x42002000	16	9	N			27					15: DEBUG	N/A

Peripheral Name	Base Address	IRQ Line	AHB Clock		APB Clock		Generic Clock	PAC		Events		DMA	
			Index	Enabled at Reset	Index	Enabled at Reset	Index	Index	Prot at Reset	User	Generator	Index	Sleep Walking
TCC0	0x42002400	17			9	N	28	9	N	9-10: EV0-1 11-14: MC0-3	34: OVF 35: TRG 36: CNT 37-40: MC0-3	16: OVF 17-20: MC0-3	Y
TCC1	0x42002800	18			10	N	28	10	N	15-16: EV0-1 17-18: MC0-1	41: OVF 42: TRG 43: CNT 44-45: MC0-1	21: OVF 22-23: MC0-1	Y
TCC2	0x42002C00	19			11	N	29	11	N	19-20: EV0-1 21-22: MC0-1	46: OVF 47: TRG 48: CNT 49-50: MC0-1	24: OVF 25-26: MC0-1	Y
TC0	0x42003000	20			12	N	30	12	N	23: EVU	51: OVF 52-53: MC0-1	27: OVF 28-29: MC0-1	Y
TC1	0x42003400	21			13	N	30	13	N	24: EVU	54: OVF 55-56: MC0-1	30: OVF 21-32: MC0-1	Y
TC2	0x42003800	22			14	N	31	14	N	25: EVU	57: OVF 58-59: MC0-1	33: OVF 23-35: MC0-1	Y
TC3	0x42003C00	23			15	N	31	15	N	26: EVU	60: OVF 61-62: MC0-1	36: OVF 37-38: MC0-1	Y
TC4	0x42004000	24			16	N	32	16	N	27: EVU	63: OVF 64-65: MC0-1	39: OVF 40-41: MC0-1	Y
ADC0	0x42004400	25			17	N	33	17	N	28: START 29: SYNC	66: RESRDY 67: WINMON	42: RESRDY	Y
ADC1	0x42004800	26			18	N	34	18	N	30: START 31: SYNC	68: RESRDY 69: WINMON	43: RESRDY	Y
SDADC	0x42004C00	29			19	N	35	19	N	32: START 33: FLUSH	70: RESRDY 71: WINMON	44: RESRDY	Y
AC	0x42005000	27			20	N	34	20	N	34-37: SOC0-3	72-75: COMP0-3 76-77: WIN0-1		Y
DAC	0x42005400	28			21	N	36	21	N	38: START	78: EMPTY	45: EMPTY	Y
PTC	0x42005800	30			22	N	37	22	N	39: STCONV	79: EOC 80: WCOMP	EOC: 46 WCOMP: 47 SEQ: 48	
CCL	0x42005C00				23	N	38	23	N	40-43 : LUTIN0-3	781-84: LUTOUT0-3		Y
DIVAS	0x48000000		12	Y									N/A

## 14. DSU - Device Service Unit

### 14.1. Overview

The Device Service Unit (DSU) provides a means to detect debugger probes. This enables the ARM Debug Access Port (DAP) to have control over multiplexed debug pads and CPU reset. The DSU also provides system-level services to debug adapters in an ARM debug system. It implements a CoreSight Debug ROM that provides device identification as well as identification of other debug components 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

[NVMCTRL – Non-Volatile Memory Controller](#) on page 419

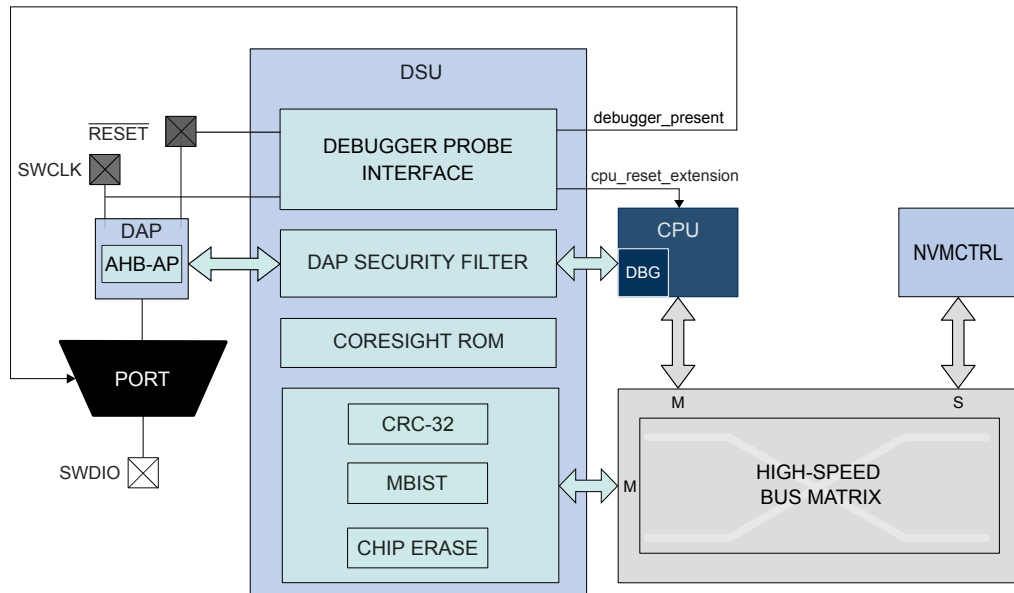
[Security Bit](#) on page 427

### 14.2. 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)

### 14.3. Block Diagram

Figure 14-1. DSU Block Diagram



### 14.4. 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

### 14.5. Product Dependencies

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

#### 14.5.1. IO 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](#) on page 62. 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.

#### 14.5.2. Power Management

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

#### Related Links

[PM – Power Manager](#) on page 162



### 14.5.3. Clocks

The DSU bus clocks (CLK\_DSU\_APB and CLK\_DSU\_AHB) can be enabled and disabled by the Main Clock Controller.

#### Related Links

[PM – Power Manager](#) on page 162

[MCLK – Main Clock](#) on page 131

[Peripheral Clock Masking](#) on page 135

### 14.5.4. DMA

Not applicable.

### 14.5.5. Interrupts

Not applicable.

### 14.5.6. Events

Not applicable.

### 14.5.7. Register Access Protection

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

When the CPU is halted in debug mode, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger.

#### Related Links

[PAC - Peripheral Access Controller](#) on page 46

### 14.5.8. Analog Connections

Not applicable.

## 14.6. Debug Operation

### 14.6.1. 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

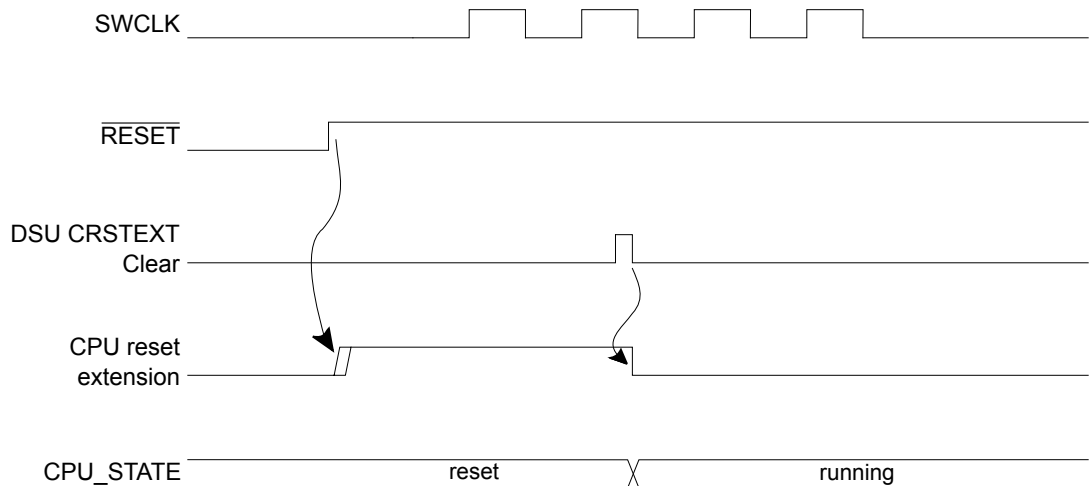
For more details on the ARM debug components, refer to the ARM Debug Interface v5 Architecture Specification.

### 14.6.2. CPU Reset Extension

“CPU reset extension” refers to the extension of the reset phase of the CPU core after the external reset is released. This ensures that the CPU is not executing code at startup while a debugger connects to the

system. It is detected on a  $\overline{\text{RESET}}$  release event when SWCLK is low. At startup, SWCLK is internally pulled up to avoid false detection of a debugger if SWCLK is left unconnected. When the CPU is held in the reset extension phase, the CPU Reset Extension bit 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 zero. 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 14-2. Typical CPU Reset Extension Set and Clear Timing Diagram**



**Related Links**

[NVMCTRL – Non-Volatile Memory Controller](#) on page 419

[Security Bit](#) on page 427

**14.6.3. Debugger Probe Detection**

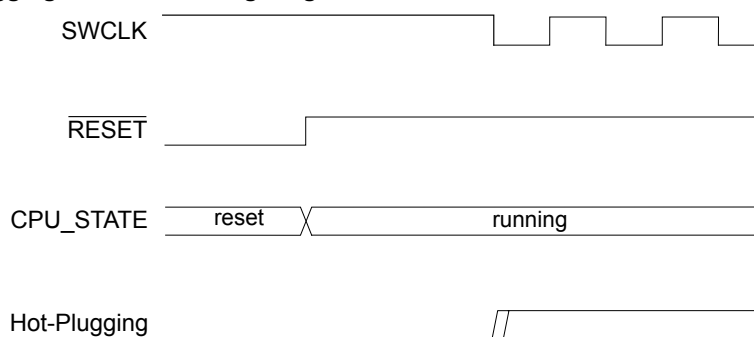
**14.6.3.1. Cold Plugging**

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.

**14.6.3.2. 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 14-3. 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 – Non-Volatile Memory Controller](#) on page 419

[Security Bit](#) on page 427

## 14.7. Chip Erase

Chip-Erase consists of removing all sensitive information stored in the chip and clearing the NVMCTRL security bit (refer to [Security Bit](#) on page 427). 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 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](#) on page 62). 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 one when completed).
4. Reset the device to let the NVMCTRL update fuses.

## 14.8. 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. 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. 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  $\overline{\text{RESET}}$ , toggling power, or writing a '1' to the Status A register CPU Reset Phase Extension bit (STATUSA.CRSTEXT). Make sure that the SWCLK pin is high when releasing  $\overline{\text{RESET}}$  to prevent extending the CPU reset.

### Related Links

[Electrical Characteristics](#) on page 1105

[NVMCTRL – Non-Volatile Memory Controller](#) on page 419

[Security Bit](#) on page 427

## 14.9. 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](#) on page 63). 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 generated by the ARM AHB-AP inside the DAP. If the device is protected, then AHB-AP read/write accesses outside the DSU external address range are discarded, causing an error response that sets the ARM AHB-AP sticky error bits (refer to the ARM Debug Interface v5 Architecture Specification on <http://www.arm.com>).

The DSU is intended to be accessed either:

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

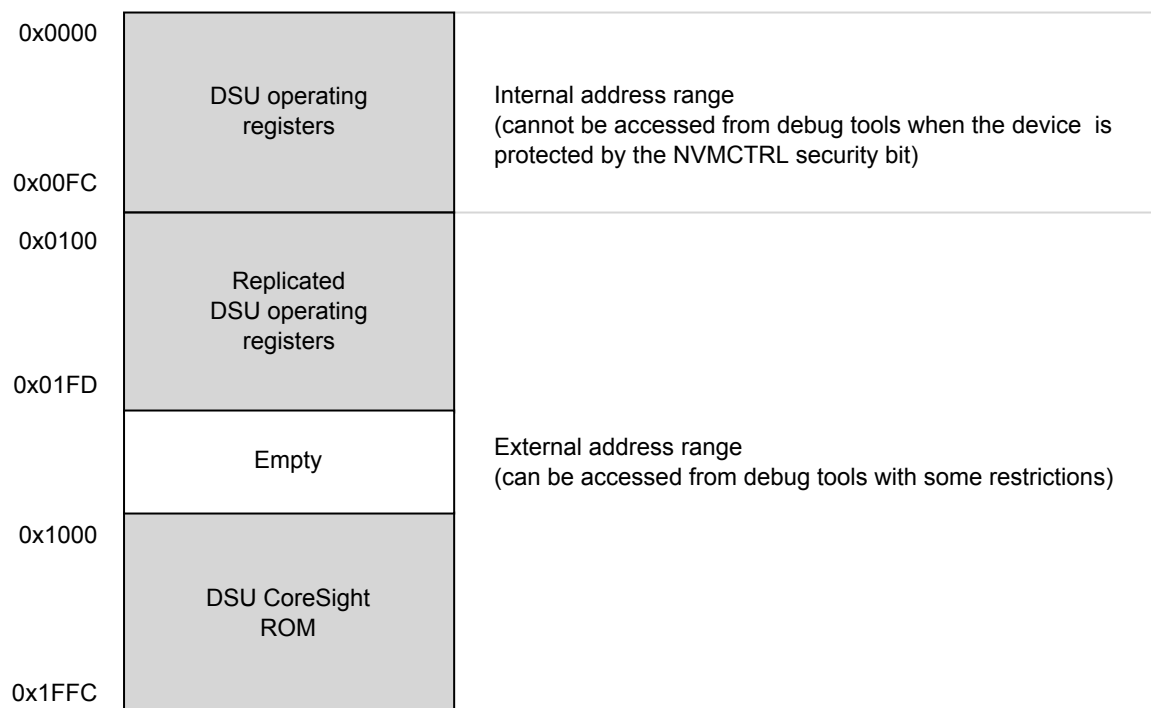
For security reasons, DSU features have limitations when used from a debug adapter. To differentiate external accesses from internal ones, the first 0x100 bytes of the DSU register map have been replicated at offset 0x100:

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

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

The DSU operating registers are located in the 0x00-0xFF area and remapped in 0x100-0x1FF to differentiate accesses coming from a debugger and the CPU. If the device is protected and an access is issued in the region 0x100-0x1FF, it is subject to security restrictions. For more information, refer to the [Table 14-1 Feature Availability Under Protection](#) on page 65.

**Figure 14-4. APB Memory Mapping**



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

**Table 14-1. 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**

[NVMCTRL – Non-Volatile Memory Controller](#) on page 419

[Security Bit](#) on page 427

## 14.10. Device Identification

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

### 14.10.1. 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 14-5. Conceptual 64-bit Peripheral ID

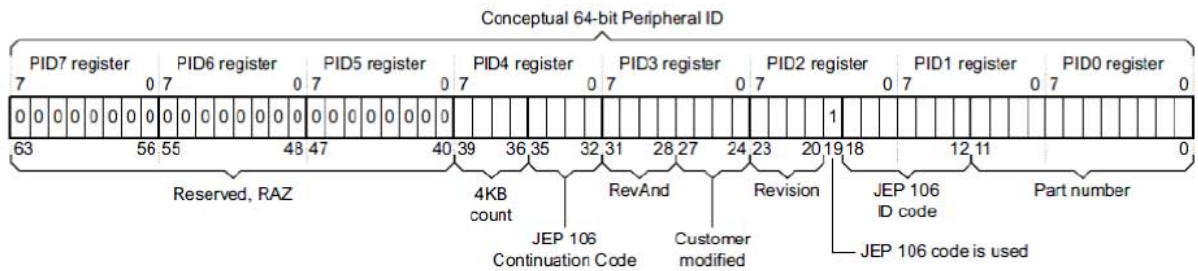


Table 14-2. Conceptual 64-Bit Peripheral ID Bit Descriptions

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

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

### 14.10.2. 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

## 14.11. Functional Description

### 14.11.1. 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.

### 14.11.2. Basic Operation

#### 14.11.2.1. Initialization

The module is enabled by enabling its clocks. For more details, refer to [Clocks](#) on page 61. The DSU registers can be PAC write-protected.

#### Related Links

[PAC - Peripheral Access Controller](#) on page 46

#### 14.11.2.2. 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](#) on page 64.

#### Related Links

[NVMCTRL – Non-Volatile Memory Controller](#) on page 419

[Security Bit](#) on page 427

#### 14.11.2.3. Operation From the CPU

There are no restrictions when accessing DSU registers from the CPU. However, the user should access DSU registers in the internal address range (0x0 – 0x100) to avoid external security restrictions. Refer to ["Intellectual Property Protection](#) on page 64.

### 14.11.3. 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 14-3. 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).

#### 14.11.3.1. 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 whole flash array when operated from the external address space. In most cases, this area will be the entire onboard non-volatile memory. The Address, Length and Data registers will be forced to predefined values once the CRC32 operation is started, and values written by the user are ignored. This allows the user to verify the contents of a protected device.

The actual test is started by writing a '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 – Non-Volatile Memory Controller](#) on page 419

[Security Bit](#) on page 427

#### 14.11.3.2. 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.

#### 14.11.4. 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 – Non-Volatile Memory Controller](#) on page 419

[Security Bit](#) on page 427

#### 14.11.5. 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 for bit read '0', write '1', in descending order.
- 1.3. Bit for bit read '1', write '0', read '0', write '1', in ascending order.
- 1.4. Bit for bit read '1', write '0', in ascending order.
- 1.5. Bit for 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 has a run time of  $O(14n)$ , where  $n$  is the number of bits 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 three different modes:

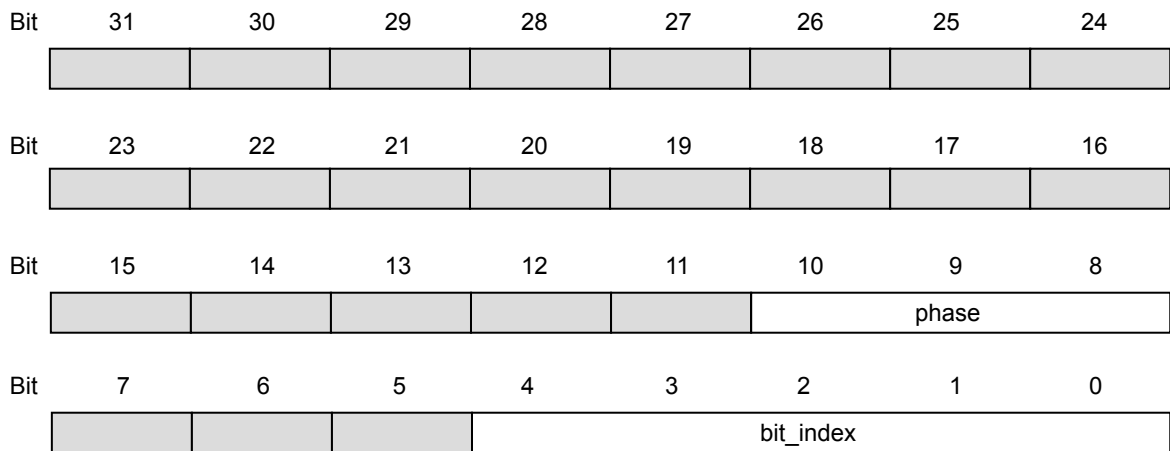
- ADDR.AMOD=0: exit-on-error (default)  
In this mode, the algorithm terminates either when a fault is detected or on successful completion. In both cases, STATUSA.DONE is set. If an error was detected, STATUSA.FAIL will be set. User then can read the DATA and ADDR registers to locate the fault.
- ADDR.AMOD=1: 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 14-6. 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 14-4. 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'
6	Read '1', write '0', decrement address
7	Read all zeros. bit_index is not used

## Related Links

[NVMCTRL – Non-Volatile Memory Controller](#) on page 419

[Security Bit](#) on page 427

[Product Mapping](#) on page 34

### 14.11.6. System Services Availability when Accessed Externally

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

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

**Table 14-5. 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)	Yes
STATUSA.CRSTEXT clearing	No (STATUSA.PERR is set when attempting to do so)

## 14.12. Register Summary

Offset	Name	Bit Pos.									
0x00	CTRL	7:0				CE	MBIST		CRC	SWRST	
0x01	STATUSA	7:0				PERR	FAIL	BERR	CRSTEXT	DONE	
0x02	STATUSB	7:0				HPE	DCCD1	DCCD0	DBGPRES	PROT	
0x03	Reserved										
0x04	ADDR	7:0	ADDR[5:0]								
0x05		15:8	ADDR[13:6]								
0x06		23:16	ADDR[21:14]								
0x07		31:24	ADDR[29:22]								
0x08	LENGTH	7:0	LENGTH[5:0]								
0x09		15:8	LENGTH[13:6]								
0x0A		23:16	LENGTH[21:14]								
0x0B		31:24	LENGTH[29:22]								
0x0C	DATA	7:0	DATA[7:0]								
0x0D		15:8	DATA[15:8]								
0x0E		23:16	DATA[23:16]								
0x0F		31:24	DATA[31:24]								
0x10	DCC0	7:0	DATA[7:0]								
0x11		15:8	DATA[15:8]								
0x12		23:16	DATA[23:16]								
0x13		31:24	DATA[31:24]								
0x14	DCC1	7:0	DATA[7:0]								
0x15		15:8	DATA[15:8]								
0x16		23:16	DATA[23:16]								
0x17		31:24	DATA[31:24]								
0x18	DID	7:0	DSEL[7:0]								
0x19		15:8	DIE[3:0]				REVISION[3:0]				
0x1A		23:16	FAMILY[0:0]		SERIES[5:0]						
0x1B		31:24	PROCESSOR[3:0]				FAMILY[4:1]				
0x1C ... 0x0FFF	Reserved										
0x1000	ENTRY0	7:0						FMT	EPRES		
0x1001		15:8	ADDOFF[3:0]								
0x1002		23:16	ADDOFF[11:4]								
0x1003		31:24	ADDOFF[19:12]								
0x1004	ENTRY1	7:0						FMT	EPRES		
0x1005		15:8	ADDOFF[3:0]								
0x1006		23:16	ADDOFF[11:4]								
0x1007		31:24	ADDOFF[19:12]								
0x1008	END	7:0	END[7:0]								
0x1009		15:8	END[15:8]								
0x100A		23:16	END[23:16]								
0x100B		31:24	END[31:24]								

Offset	Name	Bit Pos.								
0x100C ...	Reserved									
0x1FCB										
0x1FCC	MEMTYPE	7:0								SMEMP
0x1FCD		15:8								
0x1FCE		23:16								
0x1FCF		31:24								
0x1FD0	PID4	7:0	FKBC[3:0]			JEPCC[3:0]				
0x1FD1		15:8								
0x1FD2		23:16								
0x1FD3		31:24								
0x1FD4 ...	Reserved									
0x1FDF										
0x1FE0	PID0	7:0	PARTNBL[7:0]							
0x1FE1		15:8								
0x1FE2		23:16								
0x1FE3		31:24								
0x1FE4	PID1	7:0	JEPIDCL[3:0]			PARTNBH[3:0]				
0x1FE5		15:8								
0x1FE6		23:16								
0x1FE7		31:24								
0x1FE8	PID2	7:0	REVISION[3:0]		JEPU	JEPIDCH[2:0]				
0x1FE9		15:8								
0x1FEA		23:16								
0x1FEB		31:24								
0x1FEC	PID3	7:0	REVAND[3:0]			CUSMOD[3:0]				
0x1FED		15:8								
0x1FEE		23:16								
0x1FEF		31:24								
0x1FF0	CID0	7:0	PREAMBLE0[7:0]							
0x1FF1		15:8								
0x1FF2		23:16								
0x1FF3		31:24								
0x1FF4	CID1	7:0	CCLASS[3:0]			PREAMBLE[3:0]				
0x1FF5		15:8								
0x1FF6		23:16								
0x1FF7		31:24								
0x1FF8	CID2	7:0	PREAMBLE2[7:0]							
0x1FF9		15:8								
0x1FFA		23:16								
0x1FFB		31:24								
0x1FFC	CID3	7:0	PREAMBLE3[7:0]							
0x1FFD		15:8								
0x1FFE		23:16								
0x1FFF		31:24								

### 14.13. 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](#) on page 61.

### 14.13.1. Control

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

Bit	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 1 – 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.

### 14.13.2. 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.



### 14.13.3. Status B

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

Bit	7	6	5	4	3	2	1	0
				HPE	DCCD1	DCCD0	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.

#### 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.

#### 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.

### 14.13.4. 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]								
Access	R/W	R/W	R/W	R/W	R/W	R/W			
Reset	0	0	0	0	0	0			

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

### 14.13.5. 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.

### 14.13.6. 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.

### 14.13.7. 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.

### 14.13.8. 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.

### 14.13.9. Device Identification

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

**Name:** DID

**Offset:** 0x0018

**Reset:** see related links

**Property:** PAC Write-Protection

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
	DSEL[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.

#### Bits 7:0 – DSEL[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.

### 14.13.10. CoreSight ROM Table Entry 0

**Name:** ENTRY0  
**Offset:** 0x1000  
**Reset:** 0XXXXXXXX00X  
**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.



### 14.13.11. CoreSight ROM Table Entry 1

**Name:** ENTRY1  
**Offset:** 0x1004  
**Reset:** 0XXXXXXXX00X  
**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.

### 14.13.12. CoreSight ROM Table End

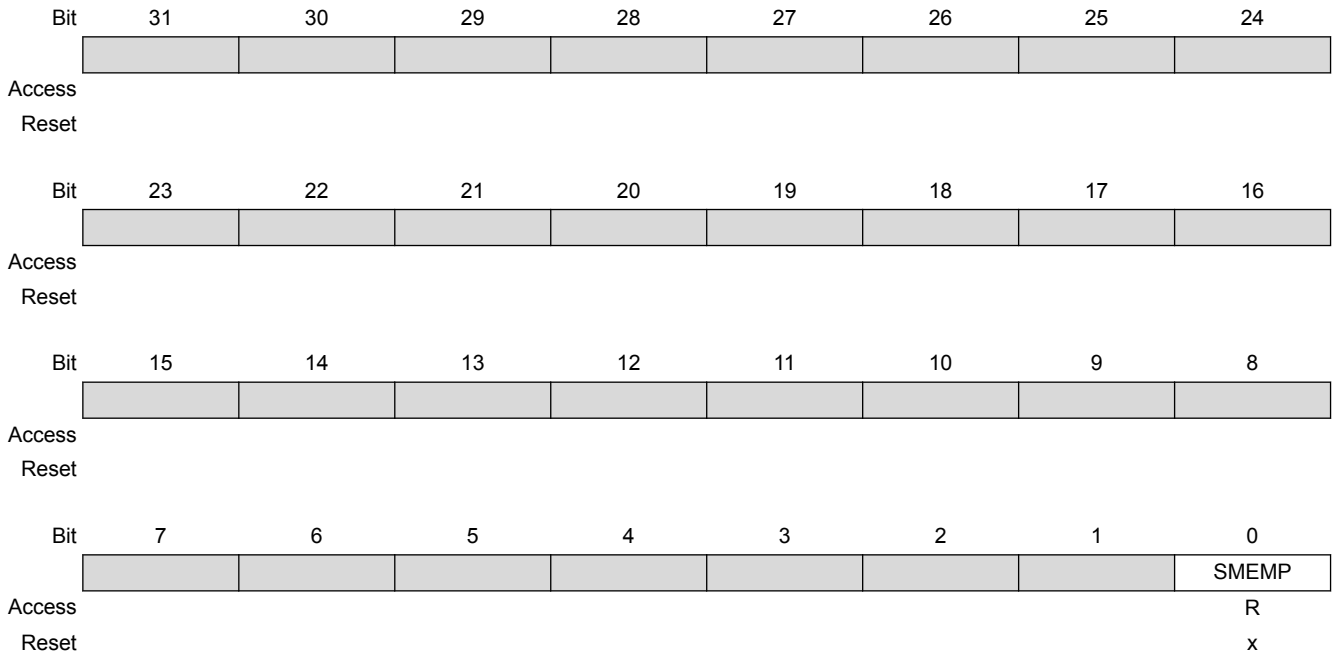
**Name:** END  
**Offset:** 0x1008  
**Reset:** 0x00000000  
**Property:** -

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.

### 14.13.13. CoreSight ROM Table Memory Type

**Name:** MEMTYPE  
**Offset:** 0x1FCC  
**Reset:** 0x0000000X  
**Property:** -



#### Bit 0 – SMEMP: System Memory Present

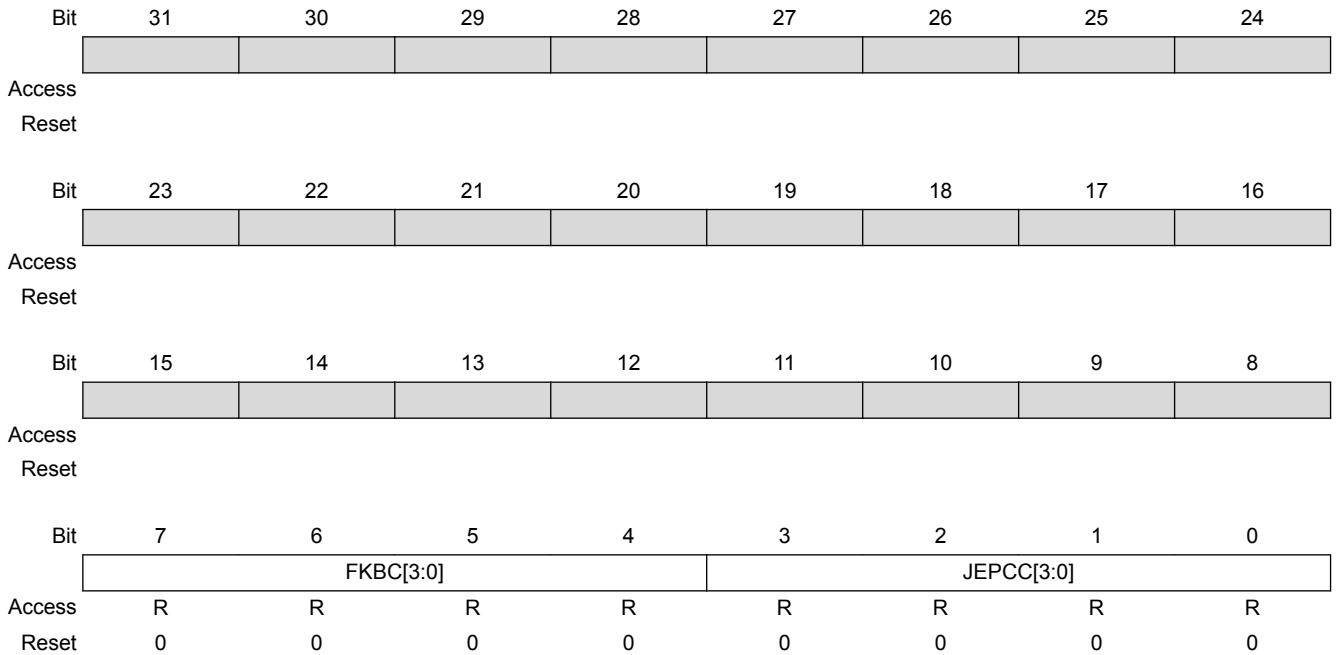
This bit indicates whether system memory is present on the bus that connects to the ROM table.

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.

### 14.13.14. Peripheral Identification 4

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



**Bits 7:4 – FKBC[3:0]: 4KB Count**

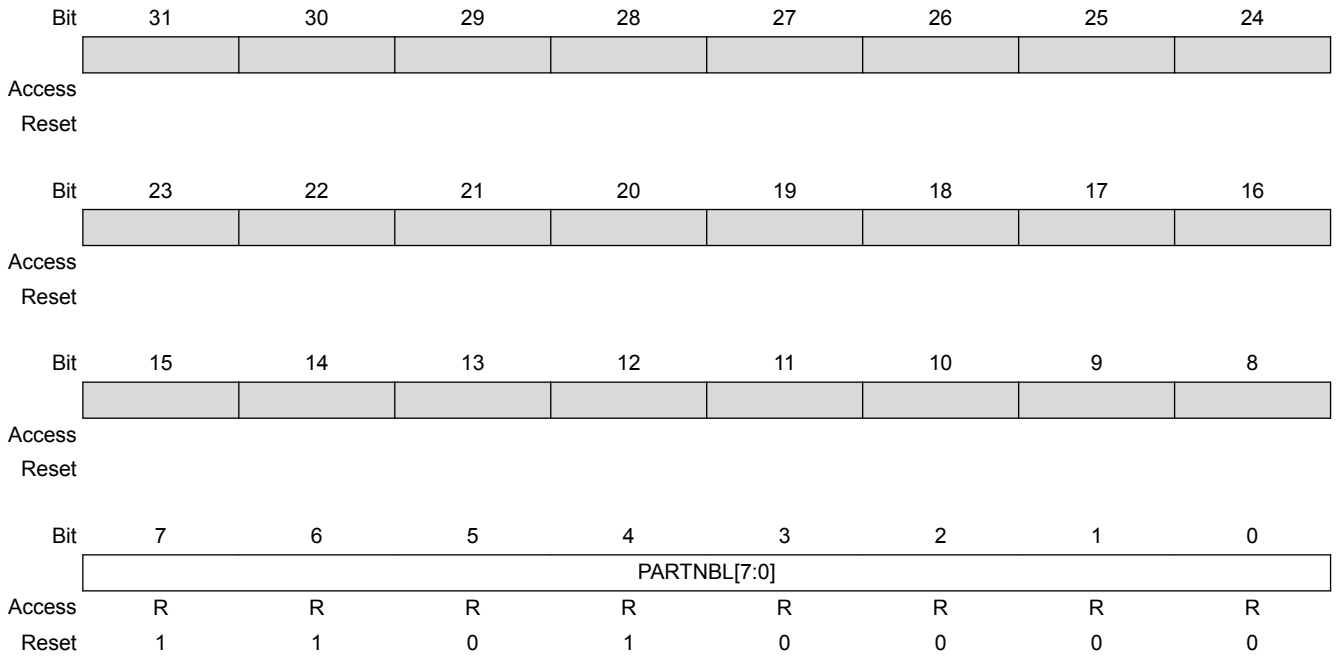
These bits will always return zero when read, indicating that this debug component occupies one 4KB block.

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

These bits will always return zero when read, indicating an Atmel device.

### 14.13.15. Peripheral Identification 0

**Name:** PID0  
**Offset:** 0x1FE0  
**Reset:** 0x000000D0  
**Property:** -

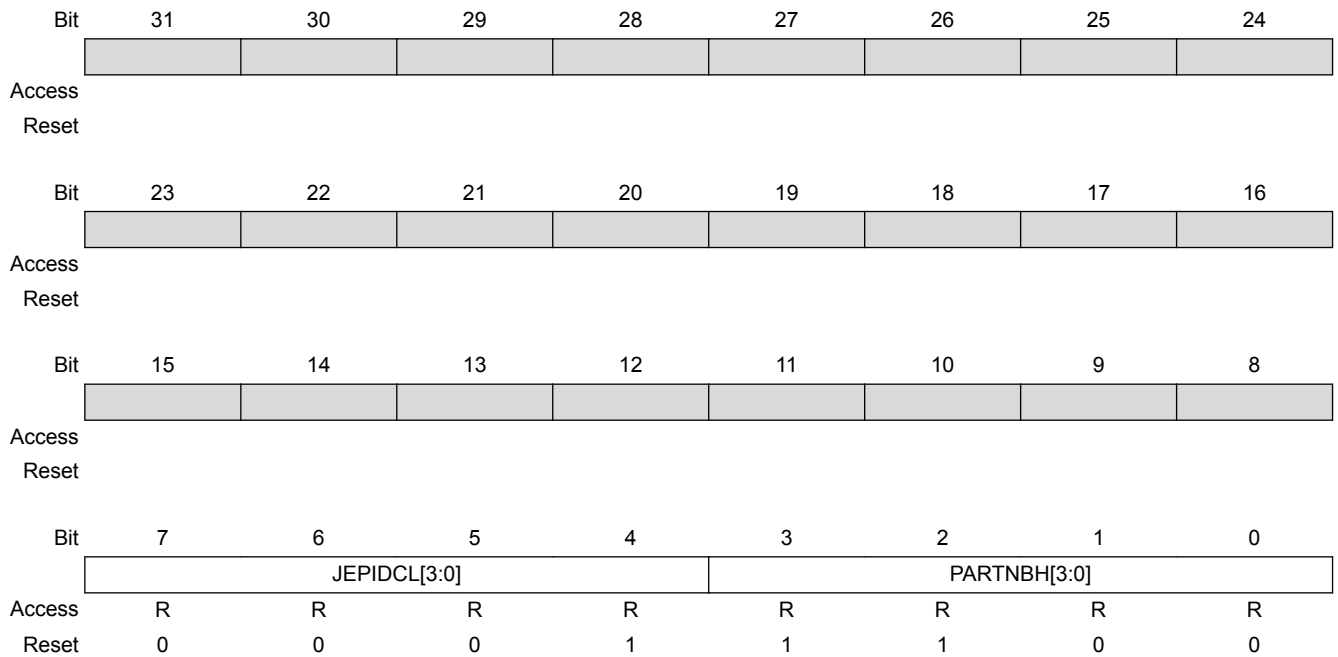


**Bits 7:0 – PARTNBL[7:0]: Part Number Low**

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

### 14.13.16. Peripheral Identification 1

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



**Bits 7:4 – JEPIDCL[3:0]: Low part of the JEP-106 Identity Code**

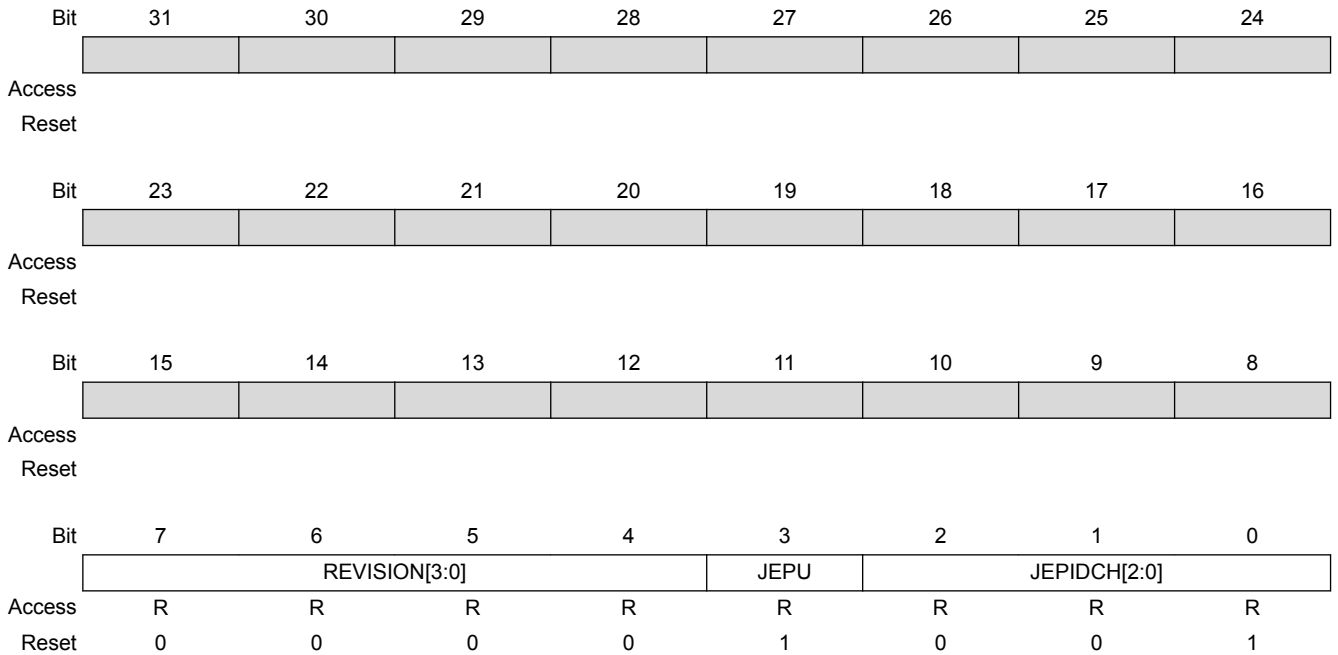
These bits will always return 0xF when read, indicating a Atmel device (Atmel JEP-106 identity code is 0x1F).

**Bits 3:0 – PARTNBH[3:0]: Part Number High**

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

### 14.13.17. 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**

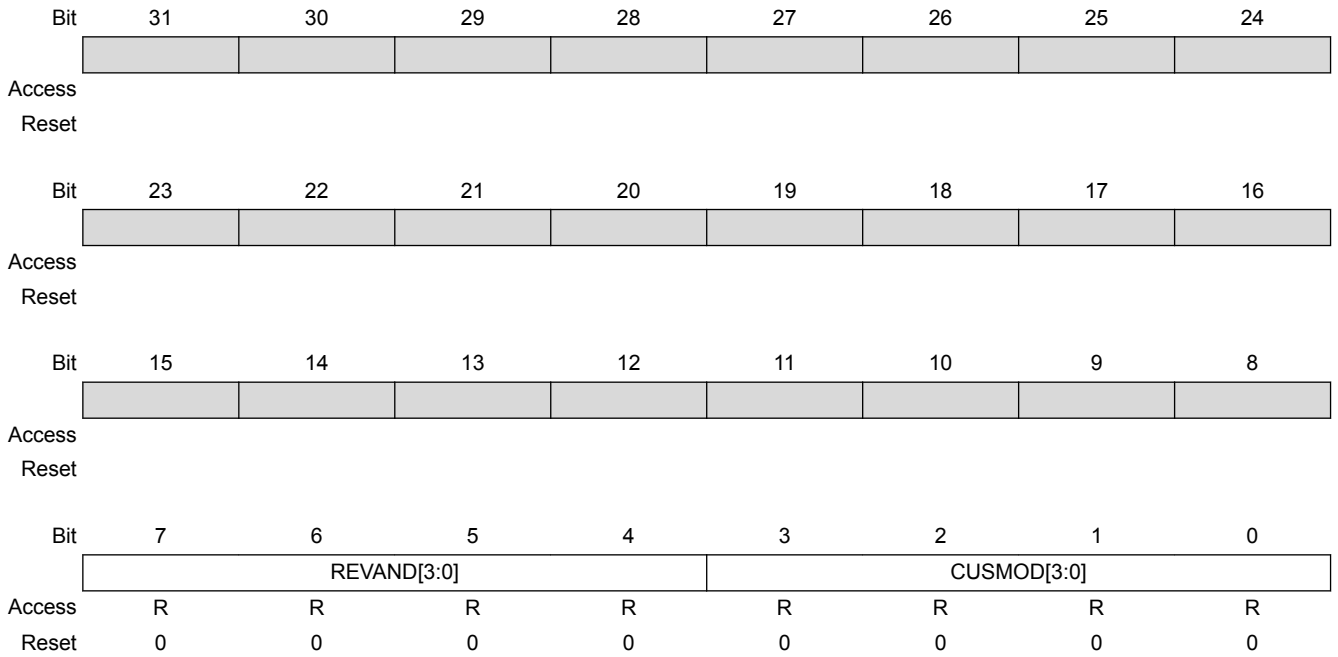
This bit will always return one when read, indicating that JEP-106 code is used.

**Bits 2:0 – JEPIDCH[2:0]: JEP-106 Identity Code High**

These bits will always return 0x1 when read, indicating an Atmel device (Atmel JEP-106 identity code is 0x1F).

### 14.13.18. Peripheral Identification 3

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



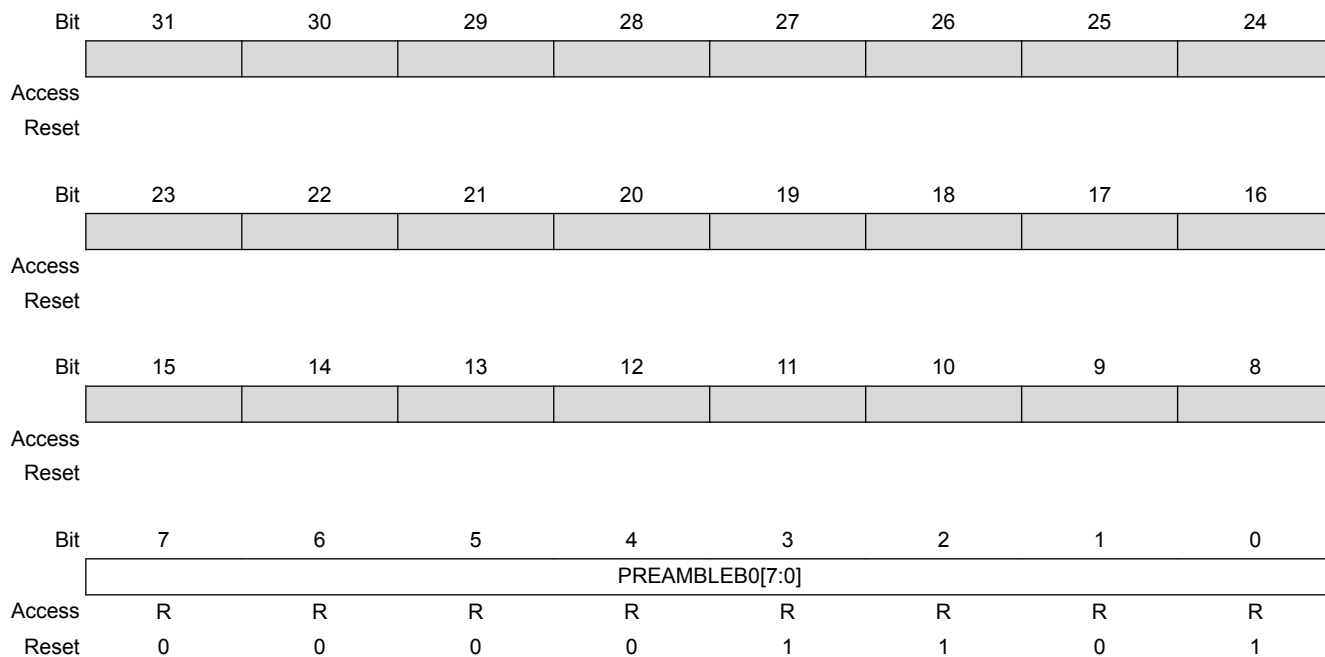
**Bits 7:4 – REVAND[3:0]: Revision Number**  
 These bits will always return 0x0 when read.

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



### 14.13.19. Component Identification 0

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

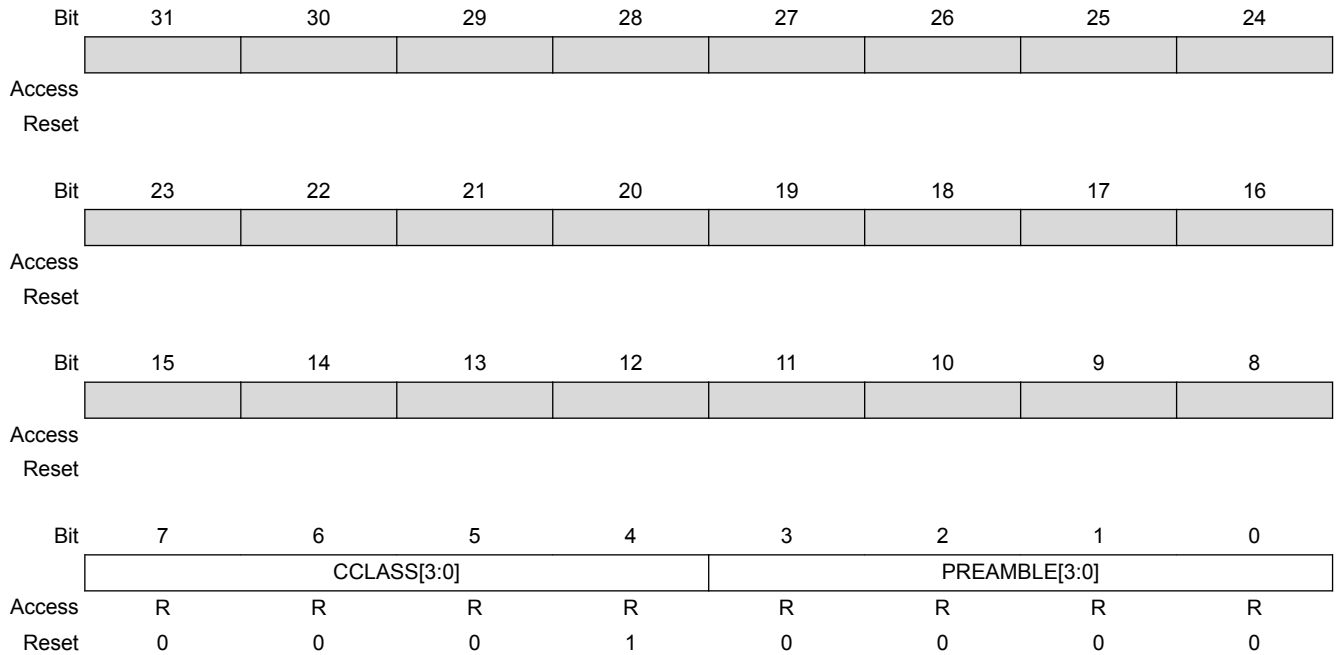


**Bits 7:0 – PREAMBLEB0[7:0]: Preamble Byte 0**

These bits will always return 0xD when read.

### 14.13.20. Component Identification 1

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



**Bits 7:4 – CCLASS[3:0]: Component Class**

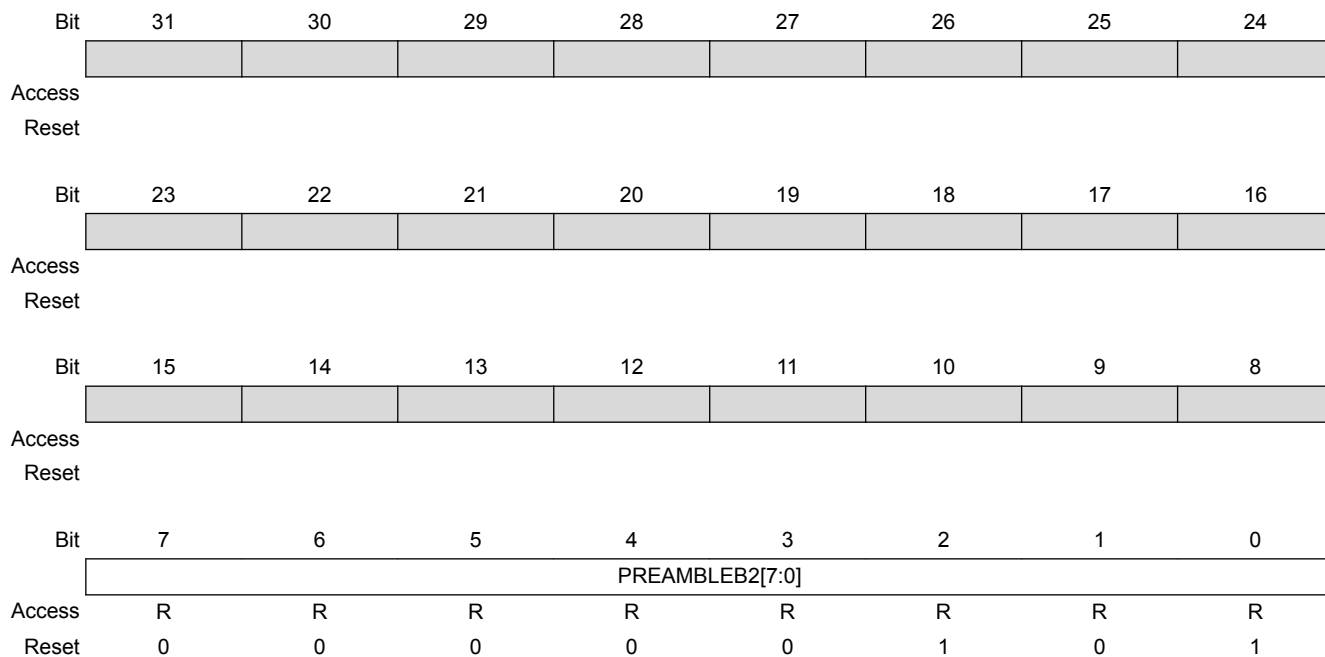
These bits will always return 0x1 when read indicating that this ARM CoreSight component is ROM table (refer to the ARM Debug Interface v5 Architecture Specification at <http://www.arm.com>).

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

These bits will always return 0x0 when read.

### 14.13.21. Component Identification 2

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

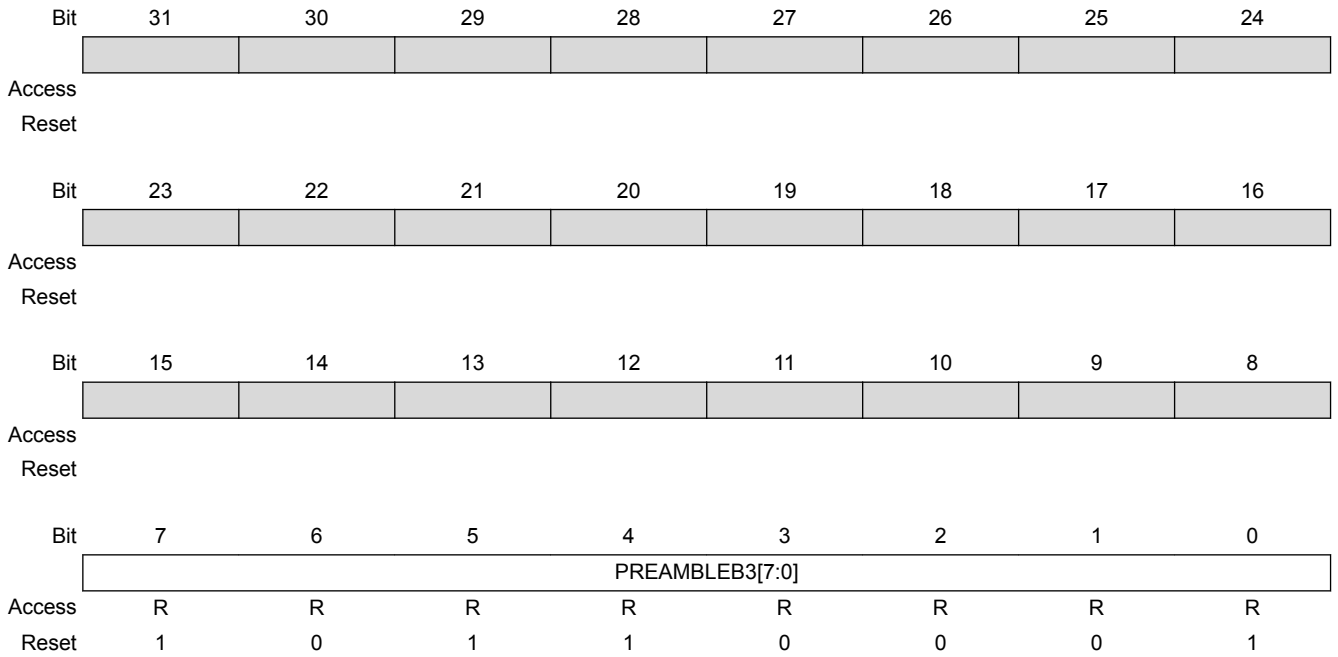


#### Bits 7:0 – PREAMBLEB2[7:0]: Preamble Byte 2

These bits will always return 0x05 when read.

### 14.13.22. Component Identification 3

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



**Bits 7:0 – PREAMBLEB3[7:0]: Preamble Byte 3**

These bits will always return 0xB1 when read.

## 15. DIVAS – Divide and Square Root Accelerator

### 15.1. Overview

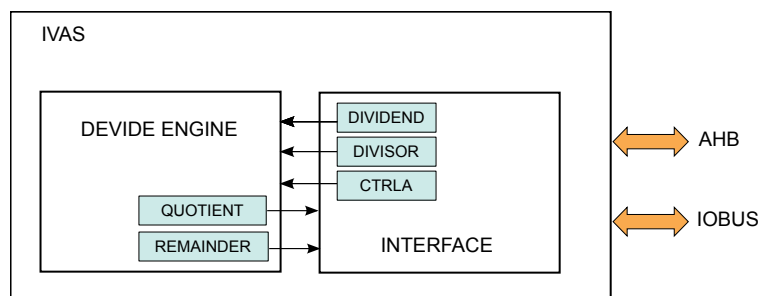
The Divide and Square Root Accelerator (DIVAS) is a programmable 32-bit signed or unsigned hardware divider and a 32-bit unsigned square root hardware engine. The DIVAS is connected to the high-speed bus matrix and may also be accessed using the low-latency CPU local bus (IOBUS; ARM® single-cycle I/O port). The DIVAS takes dividend and divisor values and returns the quotient and remainder when it is used as divider. The DIVAS takes unsigned input value and returns its square root and remainder when it is used as square root function.

### 15.2. Features

- Division accelerator for Cortex-M0+ systems
- 32-bit signed or unsigned integer division
- 32-bit unsigned square root
- 32-bit division in 2-16 cycles
- Programmable leading zero optimization
- Result includes quotient and remainder
- Result includes square root and remainder
- Busy and Divide-by-zero status
- Automatic start of operation when divisor or square root input is loaded

### 15.3. Block Diagram

Figure 15-1. DIVAS Block Diagram



### 15.4. Signal Description

Not applicable

### 15.5. Product Dependencies

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

### 15.5.1. I/O Lines

Not applicable

### 15.5.2. Power Management

The DIVAS will not operate in any sleep mode .

### 15.5.3. Clocks

The DIVAS bus clock (CLK\_DIVAS\_AHB) can be enabled and disabled in the power manager, and the default state of CLK\_DIVAS\_AHB can be found in the Peripheral Clock Masking section in the Power Manager chapter.

### 15.5.4. DMA

Not applicable

### 15.5.5. Interrupts

Not applicable

### 15.5.6. Events

Not applicable

### 15.5.7. Debug Operation

Not applicable

### 15.5.8. Register Access Protection

Certain registers cannot be modified while DIVAS is busy. The following registers are write-protected while busy:

- Control A ([CTRLA](#) on page 102)
- Dividend ([DIVIDEND](#) on page 104)
- Divisor ([DIVISOR](#) on page 105)
- Square Root Input ([SQRNUM](#) on page 108)

Accessing these registers while protected will result in an error.

### 15.5.9. Analog Connections

Not applicable

### 15.5.10. CPU Local Bus

The CPU local bus (IOBUS) is an interface that connects the CPU directly to the DIVAS. It is a single-cycle bus interface, and does not support wait states. It supports byte, half word and word sizes. This bus is generally used for low latency. All registers can be read and written using this bus.

Since the IOBUS cannot wait for DIVAS to complete operation, the Quotient and Remainder registers must be only be read via the IOBUS while the Busy bit in the Status register (STATUS.BUSY) is zero to prevent incorrect data from being read.

## 15.6. Functional Description

### 15.6.1. Principle of Operation

The Divide and Square Root Accelerator (DIVAS) supports signed or unsigned hardware division of 32-bit values and unsigned square root of 32-bit value. It is accessible from the CPU via both the AHB bus and

IOPUS. When the dividend and divide registers are programmed, the division starts and the result will be stored in the Result and Remainder registers. The Busy and Divide-by-zero status can be read from STATUS register.

When the square root input register ([SQRNUM](#) on page 108) is programmed, the square root function starts and the result will be stored in the Result and Remainder registers. The Busy status can be read from STATUS register.

## 15.6.2. Basic Operation

### 15.6.2.1. Initialization

The DIVAS configuration cannot be modified while a divide operation is ongoing. The following bits must be written prior to starting a division:

- Sign selection bit in Control A register ([CTRLA](#) on page 102.SIGNED)
- Leading zero mode bit in Control A register ([CTRLA](#) on page 102.DLZ)

### 15.6.2.2. Performing Division

First write the dividend to DIVIDEND register. Writing the divisor to DIVISOR register starts the division and sets the busy bit in the Status register (STATUS.BUSY). When the division has completed, the STATUS.BUSY bit is cleared and the result will be stored in RESULT and REMAINDER registers.

The RESULT and REMAINDER registers can be read directly via the high-speed bus without checking first STATUS.BUSY. Wait states will be inserted on the high-speed bus until the operation is complete. The IOPUS does not support wait states. For accesses via the IOPUS, the STATUS.BUSY bit must be polled before reading the result from the RESULT and REMAINDER registers.

### 15.6.2.3. Operand Size

#### Divide

The DIVAS can perform 32-bit signed and unsigned division and the operation follows the equation as below.

$$RESULT[31:0] = DIVIDEND[31:0] / DIVISOR[31:0]$$

$$REMAINDER[31:0] = DIVIDEND[31:0] \% DIVISOR[31:0]$$

DIVAS completes 32-bit division in 2-16 cycles.

#### Square Root

The DIVAS can perform 32-bit unsigned division and the operation follows the equation as below.

$$REMAINDER[31:0] = \sqrt{SQRNUM[31:0]}$$

$$REMAINDER[31:0] = SQRNUM[31:0] - RESULT[31:0]^2$$

### 15.6.2.4. Signed Division

When CTRLA.SIGNED is one, both the input and the result will be in 2's complement format. The results of signed division are such that the remainder and dividend have the same sign and the quotient is negative if the dividend and divisor have opposite signs. 16-bit results are sign extended to 32-bits. Note that when the maximum negative number is divided by the minimum negative number, the resulting quotient overflows the signed integer range and will return the maximum negative number with no indication of the overflow. This occurs for 0x80000000 / 0xFFFFFFFF in 32-bit operation and 0x8000 / 0xFFFF in 16-bit operation.

#### 15.6.2.5. Divide By Zero

A divide by zero fault occurs if the DIVISOR is programmed to zero. QUOTIENT will be zero and the REMAINDER is equal to DIVIDEND. Divide by zero sets the Divide-by-zero bit in the Status register (STATUS.DBZ) to one. STATUS.DBZ must be cleared by writing a one to it.

#### 15.6.2.6. Leading Zero Optimization

Leading zero optimization can reduce the time it takes to complete a division by skipping leading zeros in the DIVIDEND (or leading ones in signed mode). Leading zero optimization is enabled by default and can be disabled by the Disable Leading Zero bit in the Control A register (CTRLA.DLZ). When CTRLA.DLZ is zero, 16-bit division completes in 2-8 cycles and 32-bit division completes in 2-16 cycles, depending on the dividend value. If deterministic timing is required, setting CTRLA.DLZ to one forces 16-bit division to always take 8 cycles and 32-bit division to always take 16 cycles.

#### 15.6.2.7. Unsigned Square Root

When the square root input register ([SQRNUM](#) on page 108) is programmed, the square root function starts and the result will be stored in the Result and Remainder registers. The Busy status can be read from STATUS register.



## 15.7. Register Summary

Offset	Name	Bit Pos.							
0x00	CTRLA	7:0						DLZ	SIGNED
0x01	Reserved								
...									
0x03									
0x04	STATUS	7:0						DBZ	BUSY
0x05	Reserved								
...									
0x07									
0x08	DIVIDEND	7:0	DIVIDEND[7:0]						
0x09		15:8	DIVIDEND[15:8]						
0x0A		23:16	DIVIDEND[23:16]						
0x0B		31:24	DIVIDEND[31:24]						
0x0C	DIVISOR	7:0	DIVISOR[7:0]						
0x0D		15:8	DIVISOR[15:8]						
0x0E		23:16	DIVISOR[23:16]						
0x0F		31:24	DIVISOR[31:24]						
0x10	RESULT	7:0	RESULT[7:0]						
0x11		15:8	RESULT[15:8]						
0x12		23:16	RESULT[23:16]						
0x13		31:24	RESULT[31:24]						
0x14	REMAINDER	7:0	REMAINDER[7:0]						
0x15		15:8	REMAINDER[15:8]						
0x16		23:16	REMAINDER[23:16]						
0x17		31:24	REMAINDER[31:24]						
0x18	SQRNUM	7:0	SQRNUM[7:0]						
0x19		15:8	SQRNUM[15:8]						
0x1A		23:16	SQRNUM[23:16]						
0x1B		31:24	SQRNUM[31:24]						

## 15.8. Register Description

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

### 15.8.1. Control A

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x00

**Property:** -

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

#### Bit 1 – DLZ: Disable Leading Zero Optimization

Value	Description
0	Enable leading zero optimization; 32-bit division takes 2-16 cycles.
1	Disable leading zero optimization; 32-bit division takes 16 cycles.

#### Bit 0 – SIGNED: Signed Division Enable

Value	Description
0	Unsigned division.
1	Signed division.

## 15.8.2. Status

**Name:** STATUS  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** -

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

### Bit 1 – DBZ: Disable-By-Zero

Writing a zero to this bit has no effect.

Writing a one to this bit clears DBZ to zero.

Value	Description
0	A divide-by-zero fault has not occurred
1	A divide-by-zero fault has occurred

### Bit 0 – BUSY: DIVAS Accelerator Busy

This bit is set when a value is written to the [DIVISOR](#) on page 105 or [SQRNUM](#) on page 108 registers.

This bit is cleared when either division or square root function completes and results are ready in the [RESULT](#) on page 106 and [REMAINDER](#) on page 107 registers.

Value	Description
0	DIVAS is idle
1	DIVAS is busy with an ongoing division

### 15.8.3. Dividend

**Name:** DIVIDEND  
**Offset:** 0x08  
**Reset:** 0x0000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	DIVIDEND[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
	DIVIDEND[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
	DIVIDEND[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
	DIVIDEND[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 – DIVIDEND[31:0]: Dividend Value**

Holds the 32-bit dividend for the divide operation. If the Signed bit in Control A register (CTRLA.SIGNED) is zero, DIVIDEND is unsigned. If CTRLA.SIGNED = 1, DIVIDEND is signed two's complement. Refer to [Performing Division](#) on page 99, [Operand Size](#) on page 99 and [Signed Division](#) on page 99.

### 15.8.4. Divisor

**Name:** DIVISOR  
**Offset:** 0x0C  
**Reset:** 0x0000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
DIVISOR[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
DIVISOR[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
DIVISOR[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
DIVISOR[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 – DIVISOR[31:0]: Divisor Value

Holds the 32-bit divisor for the divide operation. If the Signed bit in Control A register (CTRLA.SIGNED) is zero, DIVISOR is unsigned. If CTRLA.SIGNED = 1, DIVISOR is signed two's complement. Writing the DIVISOR register will start the divide function. Refer to [Performing Division](#) on page 99, [Operand Size](#) on page 99 and [Signed Division](#) on page 99.

## 15.8.5. Result

**Name:** RESULT

**Offset:** 0x10

**Reset:** 0x0000

**Property:** -

Bit	31	30	29	28	27	26	25	24
	RESULT[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
	RESULT[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
	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 31:0 – RESULT[31:0]: Result of Operation

Holds the 32-bit result of the last performed operation. For a divide operation this is the quotient. If the Signed bit in Control A register (CTRLA.SIGNED) is zero, the quotient is unsigned. If CTRLA.SIGNED = 1, the quotient is signed two's complement. For a square root operation this is the square root. Refer to [Performing Division](#) on page 99, [Operand Size](#) on page 99 and [Signed Division](#) on page 99.

## 15.8.6. Remainder

**Name:** REMAINDER

**Offset:** 0x14

**Reset:** 0x0000

**Property:** -

Bit	31	30	29	28	27	26	25	24
	REMAINDER[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
	REMAINDER[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
	REMAINDER[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
	REMAINDER[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

### Bits 31:0 – REMAINDER[31:0]: Remainder of Operation

Holds the 32-bit remainder of the last performed operation. For a divide operation this is the division remainder. If the Signed bit in Control A register (CTRLA.SIGNED) is zero, the quotient is unsigned. If CTRLA.SIGNED = 1, the quotient is signed two's complement. For a square root operation this is the square root remainder. Refer to [Performing Division](#) on page 99, [Operand Size](#) on page 99 and [Signed Division](#) on page 99.

## 15.8.7. Square Root Input

**Name:** SQRNUM  
**Offset:** 0x18  
**Reset:** 0x0000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	SQRNUM[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
	SQRNUM[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
	SQRNUM[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
	SQRNUM[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 – SQRNUM[31:0]: Square Root Input

Holds the 32-bit unsigned input for the square root operation. Writing the SQRNUM register will start the square root function. Refer to [Unsigned Square Root](#) on page 100.

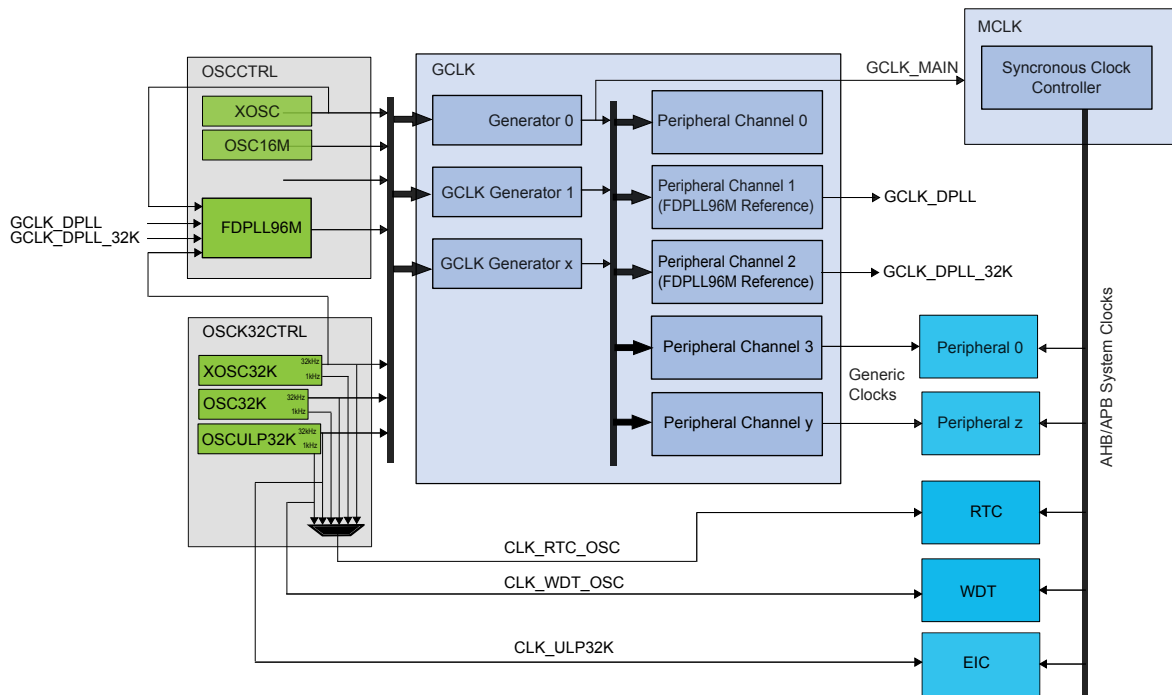


## 16. Clock System

This chapter only aims to summarize the clock distribution and terminology in the SAM C21 device. It will not explain every detail of its configuration. For in-depth documentation, see the referenced module chapters.

### 16.1. Clock Distribution

Figure 16-1. Clock distribution

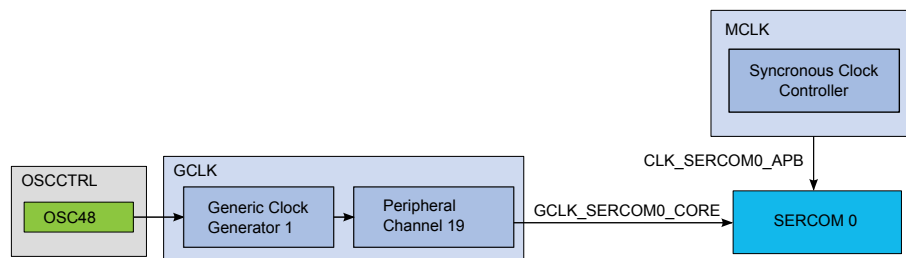


The clock system on the SAM C21 consists of:

- Clock sources, controlled by OSCCTRL and OSC32CTRL
  - A Clock source is the base clock signal used in the system. Example clock sources are the internal 48MHz oscillator (OSC48M), External crystal oscillator (XOSC) and the Digital phase locked loop (FDPLL96M).
- Generic Clock Controller (GCLK) which controls the clock distribution system, made up of:
  - Generic Clock generators: A programmable prescaler, that can use any of the system clock sources as its source clock. The Generic Clock Generator 0, also called GCLK\_MAIN, is the clock feeding the Power Manager used to generate synchronous clocks.
  - Generic Clocks: Typically the clock input of a peripheral on the system. The generic clocks, through the Generic Clock Multiplexer, can use any of the Generic Clock generators as its clock source. Multiple instances of a peripheral will typically have a separate generic clock for each instance.
- Main Clock controller (MCLK)
  - The MCLK controls synchronous clocks on the system. This includes the CPU, bus clocks (APB, AHB) as well as the synchronous (to the CPU) user interfaces of the peripherals. It contains clock masks that can turn on/off the user interface of a peripheral as well as prescalers for the CPU and bus clocks.

The figure below shows an example where SERCOM0 is clocked by the OSC48M. The OSC48M is enabled, the Generic Clock Generator 1 uses the OSC48M as its clock source, and the generic clock 19, also called GCLK\_SERCOM0\_CORE, that is connected to SERCOM0 uses generator 1 as its source. The SERCOM0 interface, clocked by CLK\_SERCOM0\_APB, has been unmasked in the APBC Mask register in the MCLK.

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



## 16.2. Synchronous and Asynchronous Clocks

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

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

## 16.3. Register Synchronization

### 16.3.1. Overview

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

### 16.3.2. General Write-Synchronization

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

However a second write access to the same core register, while synchronization is on going, is discarded and an error is reported through the PAC. To write again to the same core register in the same module, user must wait for the end of synchronization.

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

**Example:**

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

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

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

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

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

### 16.3.3. General Read-Synchronization

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

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

### 16.3.4. Completion of Synchronization

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

### 16.3.5. Enable Write-Synchronization

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

### 16.3.6. Software Reset Write-Synchronization

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

### 16.3.7. Synchronization Delay

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

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

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

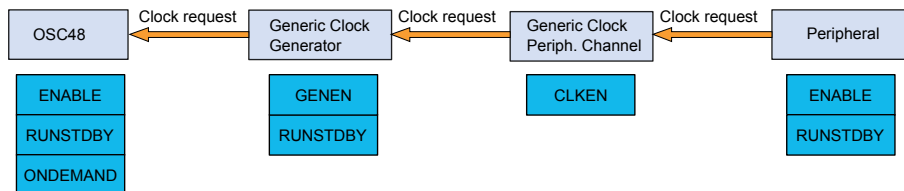
## 16.4. Enabling a Peripheral

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

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

## 16.5. On-demand, Clock Requests

Figure 16-3. Clock request routing



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

```
Delay_start_max = Clock source startup time + 2 * clock source periods + 2 *
divided clock source periods
Delay_start_min = Clock source startup time + 1 * clock source period + 1 * divided
clock source period
Delay_start_min = Clock source startup time + 1 * clock source
period + 1 * divided clock source period
```

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

```
Delay_stop_min = 1 * divided clock source period + 1 * clock source period
Delay_stop_max = 2 * divided clock source periods + 2 * clock source periods
```

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

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

## 16.6. Power Consumption vs. Speed

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

## 16.7. Clocks after Reset

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

- OSC48M is enabled and divided by 12
- GCLK\_MAIN uses OSC48M as source
- CPU and BUS clocks are undivided

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

- All generic clock generators disabled except:
  - The generator 0 (GCLK\_MAIN) using OSC48M as source, with no division
- All generic clocks disabled

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

- Generic clocks that are write-locked (WRTLOCK is written to one prior to reset)

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 Peripheral Channels, which provide the Generic Clock (GCLK\_PERIPH) to the peripheral modules, as shown in [Figure 17-2 Generic Clock Controller Block Diagram](#) on page 115. 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.

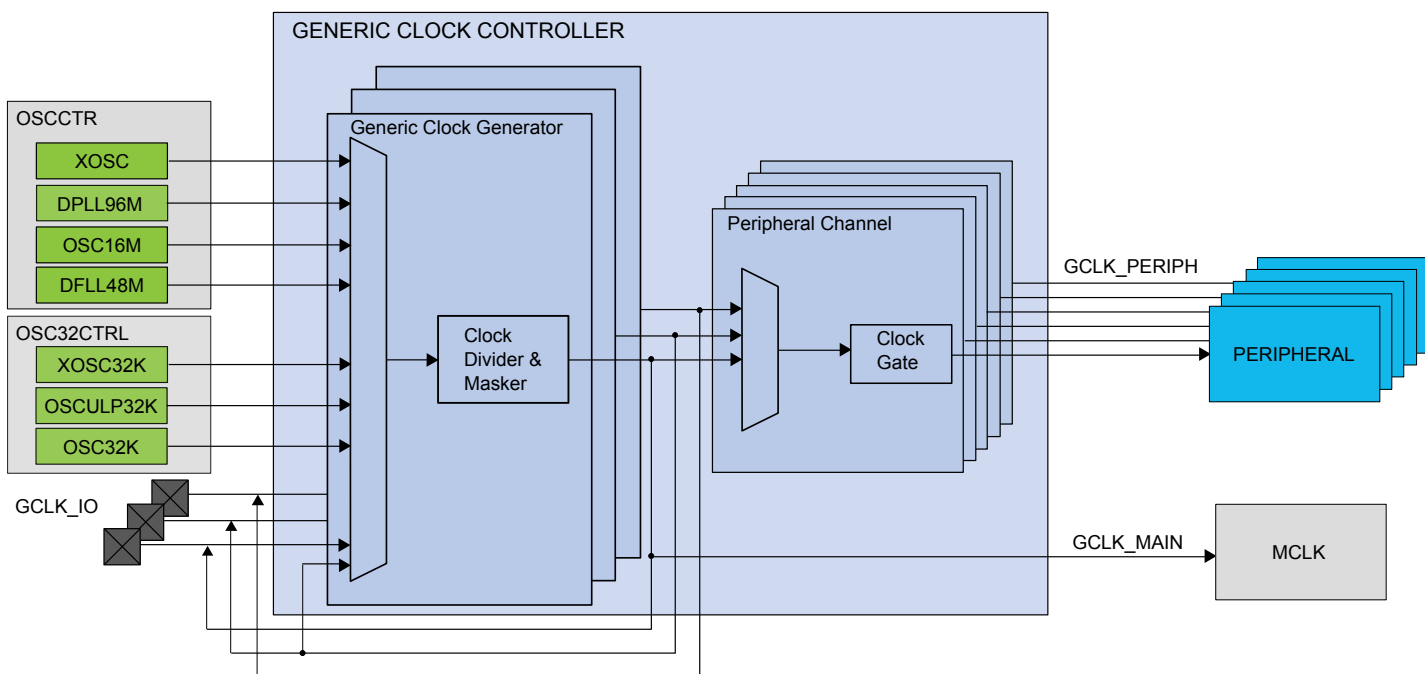
### 17.2. Features

- Provides a device-defined, configurable number of Peripheral Channel clocks
- Wide frequency range

### 17.3. Block Diagram

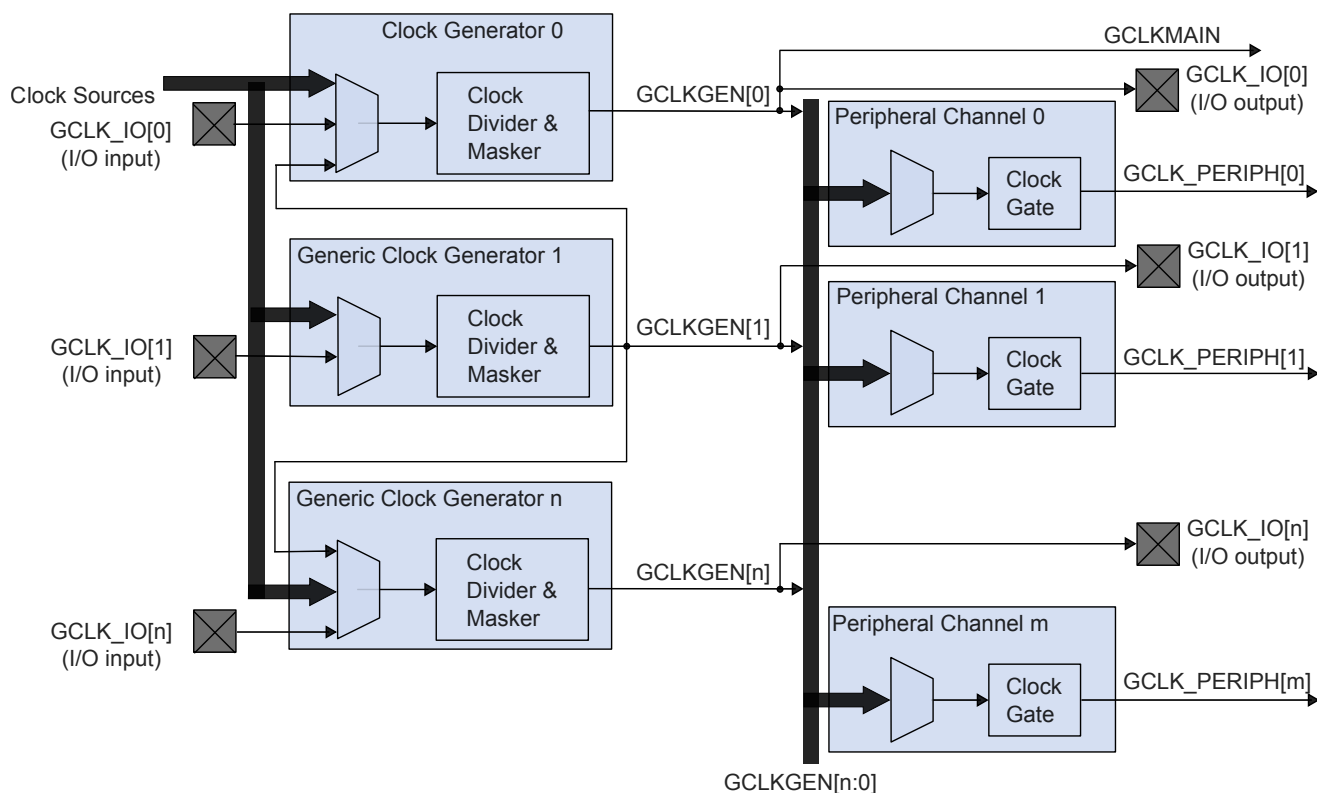
The generation of Peripheral Clock signals (GCLK\_PERIPH) and the Main Clock (GCLK\_MAIN) can be seen in [Device Clocking Diagram](#).

Figure 17-1. Device Clocking Diagram



The GCLK block diagram is shown below:

**Figure 17-2. Generic Clock Controller Block Diagram**



## 17.4. Signal Description

**Table 17-1. GCLK Signal Description**

Signal name	Type	Description
GCLK_IO[n..0]	Digital I/O	Clock source for Generators when input Generic Clock signal when output

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

## 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](#) on page 444

### 17.5.2. Power Management

The GCLK can operate in all sleep modes, if required.

#### Related Links

[PM – Power Manager](#) on page 162

### 17.5.3. Clocks

The GCLK bus clock (CLK\_GCLK\_APB) can be enabled and disabled in the Main Clock Controller.

#### Related Links

[Peripheral Clock Masking](#) on page 135

[OSC32KCTRL – 32KHz Oscillators Controller](#) on page 211

### 17.5.4. DMA

Not applicable.

### 17.5.5. Interrupts

Not applicable.

### 17.5.6. Events

Not applicable.

### 17.5.7. Debug Operation

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

### 17.5.8. 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.

When the CPU is halted in debug mode, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger.

#### Related Links

[PAC - Peripheral Access Controller](#) on page 46

### 17.5.9. Analog Connections

Not applicable.

## 17.6. Functional Description

### 17.6.1. Principle of Operation

The GCLK module is comprised of nine Generic Clock Generators (Generators) sourcing up to 64 Peripheral Channels and the Main Clock signal GCLK\_MAIN.

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 by one or more Peripheral Channels to provide a peripheral generic clock signal (GCLK\_PERIPH) to the 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 Generator must be enabled ( $GENCTRLn.GENEN=1$ ) and the division factor must be set ( $GENCTRLn.DIVSEL$  and  $GENCTRLn.DIV$ ) by performing a single 32-bit write to the Generator Control register ( $GENCTRLn$ ).
2. The Generic Clock for a peripheral must be configured by writing to the respective Peripheral Channel Control register ( $PCHCTRLm$ ). The Generator used as the source for the Peripheral Clock must be written to the GEN bit field in the Peripheral Channel Control register ( $PCHCTRLm.GEN$ ).

**Note:** Each Generator  $n$  is configured by one dedicated register  $GENCTRLn$ .

**Note:** Each Peripheral Channel  $m$  is configured by one dedicated register  $PCHCTRLm$ .

### 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 A register ( $CTRLA.SWRST$ ) to 1. All registers in the GCLK will be reset to their initial state, except for Peripheral Channels and associated Generators that have their Write Lock bit set to 1 ( $PCHCTRLm.WRTLOCK$ ). For further details, refer to [Configuration Lock](#) on page 119.

### 17.6.2.3. Generic Clock Generator

Each Generator ( $GCLK\_GEN$ ) can be set to run from one of nine different clock sources except  $GCLK\_GEN[1]$ , which can be set to run from one of eight sources.  $GCLK\_GEN[1]$  is the only Generator that can be selected as source to others Generators.

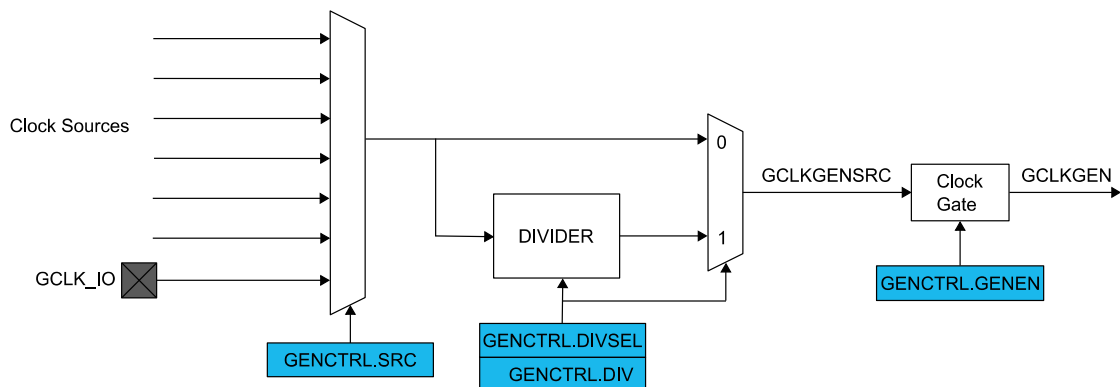
Each generator  $GCLK\_GEN[x]$  can be connected to one specific pin ( $GCLK\_IO[y]$ ). The  $GCLK\_IO[y]$  can be set to act as source to  $GCLK\_GEN[x]$  or to output the clock signal generated by  $GCLK\_GEN[x]$ .

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

Each  $GCLK\_GEN$  clock signal can then be used as clock source for Peripheral Channels. Each Generator output is allocated to one or several Peripherals.

$GCLK\_GEN[0]$  is used as  $GCLK\_MAIN$  for the synchronous clock controller inside the Main Clock Controller. Refer to the Main Clock Controller description for details on the synchronous clock generation.

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



#### Related Links

[MCLK – Main Clock](#) on page 131

#### 17.6.2.4. Enabling a Generator

A Generator is enabled by writing a '1' to the Generator Enable bit in the Generator Control register (GENCTRLn.GENEN=1).

#### 17.6.2.5. Disabling a Generator

A Generator is disabled by writing a '0' to GENCTRLn.GENEN. When GENCTRLn.GENEN=0, the GCLK\_GEN[n] clock is disabled and gated.

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

Each Generator can individually select a clock source by setting the Source Select bit group in the Generator Control register (GENCTRLn.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 using clock source A. As soon as source B is ready, the Generator will switch to it. During the switching operation, the Generator maintains clock requests to both clock sources A and B, and will release source A as soon as the switch is done. The according bit in SYNCBUSY register (SYNCBUSY.GENCTRLn) will remain '1' until the switch operation is completed.

The available clock sources are device dependent (usually the oscillators, RC oscillators, DPLL, and DFLL). Only Generator 1 can be used as a common source for all other generators.

#### 17.6.2.7. Changing the Clock Frequency

The selected source for a Generator can be divided by writing a division value in the Division Factor bit field of the Generator Control register (GENCTRLn.DIV). How the actual division factor is calculated is depending on the Divide Selection bit (GENCTRLn.DIVSEL).

If GENCTRLn.DIVSEL=0 and GENCTRLn.DIV is either 0 or 1, the output clock will be undivided.

**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. Setting the Improve Duty Cycle bit of the Generator Control register (GENCTRLn.IDC) will result in a 50/50 duty cycle.

#### 17.6.2.9. External Clock

The output clock (GCLK\_GEN) of each Generator can be sent to I/O pins (GCLK\_IO).

If the Output Enable bit in the Generator Control register is set (GENCTRLn.OE = 1) and the generator is enabled (GENCTRLn.GENEN=1), the Generator requests its clock source and the GCLK\_GEN clock is output to an I/O pin.

If GENCTRLn.OE is 0, the according I/O pin is set to an Output Off Value, which is selected by GENCTRLn.OOV: If GENCTRLn.OOV is '0', the output clock will be low when turned off. If this bit is '1', the output clock will be high when turned off.

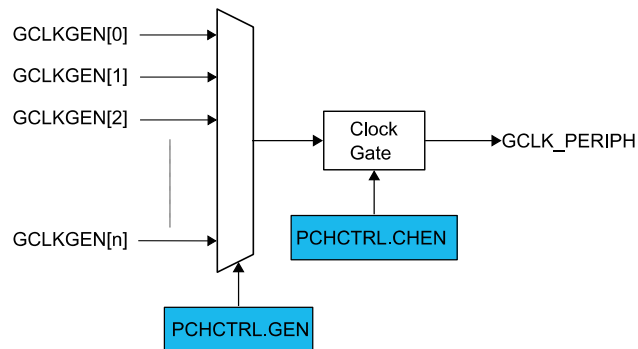
In Standby mode, if the clock is output (GENCTRLn.OE=1), the clock on the I/O pin is frozen to the OOV value if the Run In Standby bit of the Generic Control register (GENCTRLn.RUNSTDBY) is zero.

**Note:** With GENCTRLn.OE=1 and RUNSTDBY=0, entering the Standby mode can take longer due to a clock source dependent delay between turning off Power Domain 1 and 2. The maximum delay can be equal to the clock source period multiplied by the division factor.

If GENCTRLn.RUNSTDBY is '1', the GCLKGEN clock is kept running and output to the I/O pin.

### 17.6.3. Peripheral Clock

Figure 17-4. Peripheral Clock



#### 17.6.3.1. Enabling a Peripheral Clock

Before a Peripheral Clock is enabled, one of the Generators must be enabled (GENCTRLn.GENEN) and selected as source for the Peripheral Channel by setting the Generator Selection bits in the Peripheral Channel Control register (PCHCTRL.GEN). Any available Generator can be selected as clock source for each Peripheral Channel.

When a Generator has been selected, the peripheral clock is enabled by setting the Channel Enable bit in the Peripheral Channel Control register, PCHCTRLm.CHEN = 1. The PCHCTRLm.CHEN bit must be synchronized to the generic clock domain. PCHCTRLm.CHEN will continue to read as its previous state until the synchronization is complete.

#### 17.6.3.2. Disabling a Peripheral Clock

A Peripheral Clock is disabled by writing PCHCTRLm.CHEN=0. The PCHCTRLm.CHEN bit must be synchronized to the Generic Clock domain. PCHCTRLm.CHEN will stay in its previous state until the synchronization is complete. The Peripheral Clock is gated when disabled.

#### Related Links

[PCHCTRLm](#) on page 128

#### 17.6.3.3. Selecting the Clock Source for a Peripheral

When changing a peripheral clock source by writing to PCHCTRLm.GEN, the peripheral clock must be disabled before re-enabling it with the new clock source setting. This prevents glitches during the transition:

1. Disable the Peripheral Channel by writing PCHCTRLm.CHEN=0
2. Assert that PCHCTRLm.CHEN reads '0'
3. Change the source of the Peripheral Channel by writing PCHCTRLm.GEN
4. Re-enable the Peripheral Channel by writing PCHCTRLm.CHEN=1

#### Related Links

[PCHCTRLm](#) on page 128

#### 17.6.3.4. Configuration Lock

The peripheral clock configuration can be locked for further write accesses by setting the Write Lock bit in the Peripheral Channel Control register (PCHCTRLm.WRTLOCK=1). All writing to the PCHCTRLm register will be ignored. It can only be unlocked by a Power Reset.

The Generator source of a locked Peripheral Channel will be locked, too: The corresponding GENCTRLn register is locked, and can be unlocked only by a Power Reset.

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

**Related Links**

[CTRLA](#) on page 123

[PCHCTRLm](#) on page 128

**17.6.4. Additional Features**

**17.6.4.1. Peripheral 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 Peripheral Channels after Reset is device-dependent.

Refer to GENCTRLn.SRC for details on GENCTRLn reset.

Refer to PCHCTRLm.SRC for details on PCHCTRLm reset.

**17.6.5. Sleep Mode Operation**

**17.6.5.1. SleepWalking**

The GCLK module supports the SleepWalking 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 Peripheral Channel stages successively, and delivers the clock to the peripheral.

The RUNSTDBY bit in the Generator Control register controls clock output to pin during standby sleep mode. If the bit is cleared, the Generator output is not available on pin. When set, the GCLK can continuously output the generator output to GCLK\_IO. Refer to [External Clock](#) on page 118 for details.

**Related Links**

[PM – Power Manager](#) on page 162

**17.6.5.2. Minimize Power Consumption in Standby**

The following table identifies when a Clock Generator is off in Standby Mode, minimizing the power consumption:

**Table 17-2. Clock Generator n Activity in Standby Mode**

Request for Clock n present	GENCTRLn.RUNSTDBY	GENCTRLn.OE	Clock Generator n
yes	-	-	active
no	1	1	active
no	1	0	OFF
no	0	1	OFF
no	0	0	OFF

### 17.6.5.3. Entering Standby Mode

There may occur a delay when the device is put into Standby, until the power is turned off. This delay is caused by running Clock Generators: if the Run in Standby bit in the Generator Control register (GENCTRLn.RUNSTDBY) is '0', GCLK must verify that the clock is turned off properly. The duration of this verification is frequency-dependent.

#### Related Links

[PM – Power Manager](#) on page 162

### 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. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

The following registers are synchronized when written:

- Generic Clock Generator Control register (GENCTRLn)
- Control A register (CTRLA)

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

#### Related Links

[CTRLA](#) on page 123

[Register Synchronization](#) on page 110

[PCHCTRLm](#) on page 128

## 17.7. Register Summary

### 17.7.1. Common Registers

Offset	Name	Bit Pos.								
0x0	<a href="#">CTRLA</a>	7:0								SWRST
0x1	Reserved									
0x2	Reserved									
0x3	Reserved									
0x4	<a href="#">SYNCBUSY</a>	7:0	GENCTRL5	GENCTRL4	GENCTRL3	GENCTRL2	GENCTRL1	GENCTRL0		SWRST
0x5		15:8						GENCTRL8	GENCTRL7	GENCTRL6
0x6		23:16								
0x7		31:24								

### 17.7.2. Generic Clock Generator n

Offset <sup>1)</sup>	Name	Bit Pos.									
0x20 + 0x4*n	GENCTRLn on page 125	7:0					SRC[4:0]				
0x21 + 0x4*n		15:8			RUNSTDBY	DIVSEL	OE	OOV	IDC	GENEN	
0x22 + 0x4*n		23:16	DIV[7:0]								
0x23 + 0x4*n		31:24	DIV[15:8]								

### 17.7.3. Peripheral Channel Control m

Offset <sup>2)</sup>	Name	Bit Pos.									
0x80 + 0x4*m	PCHCTRLm on page 128	7:0	WRTLOCK	CHEN			GEN[3:0]				
0x81 + 0x4*m		15:8									
0x82 + 0x4*m		23:16									
0x83 + 0x4*m		31:24									

1. n is Generic Clock Generator number. Refer to [GENCTRLn](#) on page 125 for details.

2. m is Peripheral Channel number.

## 17.8. Register Description

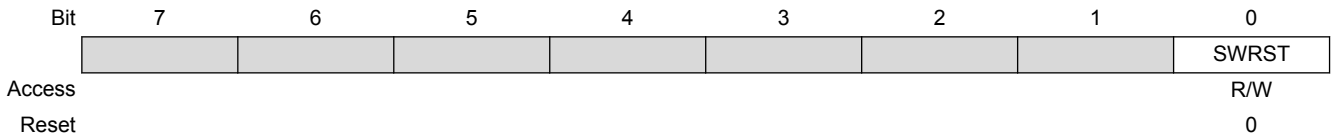
Registers can be 8, 16, or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#) on page 116.

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](#) on page 121.

### 17.8.1. Control A

**Name:** CTRLA  
**Offset:** 0x0  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized



#### Bit 0 – SWRST: Software Reset

Writing a zero to this bit has no effect.

Setting this bit to 1 will reset 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 PCHCTRLm set to 1.

Refer to GENCTRL Reset Value for details on GENCTRL register reset.

Refer to PCHCTRL Reset Value for details on PCHCTRL register reset.

Due to synchronization, there is a waiting period between setting CTRLA.SWRST and a completed Reset. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.

Value	Description
0	There is no Reset operation ongoing.
1	A Reset operation is ongoing.

## 17.8.2. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x04  
**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						R	R	R
Reset						0	0	0
Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R		R
Reset	0	0	0	0	0	0		0

### Bit 0 – SWRST: SWRST Synchronization Busy

This bit is cleared when the synchronization of the CTRLA.SWRST register bit between clock domains is complete.

This bit is set when the synchronization of the CTRLA.SWRST register bit between clock domains is started.

### Bits 2,3,4,5,6,7,8,9,10 – GENCTRLx: Generator Control x Synchronization Busy

This bit is cleared when the synchronization of the Generator Control n register (GENCTRLn) between clock domains is complete, or when clock switching operation is complete.

This bit is set when the synchronization of the Generator Control n register (GENCTRLn) between clock domains is started.



### 17.8.3. Generator Control

GENCTRLn controls the settings of Generic Generator n (n=0..8).

**Name:** GENCTRLn

**Offset:** 0x20+4\*n, (n=0..8)

**Reset:** 0x00000106 for Generator n=0, else 0x00000000

**Property:** PAC Write-Protection, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	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	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
			RUNSTDBY	DIVSEL	OE	OOV	IDC	GENEN
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
				SRC[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

#### Bits 31:16 – DIV[15:0]: Division Factor

These bits represent a division value for the corresponding Generator. The actual division factor is dependent on the state of DIVSEL. The number of relevant DIV bits for each Generator can be seen in this table. Written bits outside of the specified range will be ignored.

**Table 17-3. Division Factor Bits**

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

#### Bit 13 – RUNSTDBY: Run in Standby

This bit is used to keep the Generator running in Standby as long as it is configured to output to a dedicated GCLK\_IO pin. If GENCTRLn.OE is zero, this bit has no effect and the generator will only be running if a peripheral requires the clock.

Value	Description
0	The 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 Generator is kept running and output to its dedicated GCLK_IO pin during Standby mode.

#### Bit 12 – DIVSEL: Divide Selection

This bit determines how the division factor of the clock source of the Generator will be calculated from DIV. If the clock source should not be divided, DIVSEL must be 0 and the GENCTRLn.DIV value must be either 0 or 1.

Value	Description
0	The Generator clock frequency equals the clock source frequency divided by GENCTRLn.DIV.
1	The Generator clock frequency equals the clock source frequency divided by $2^{(GENCTRLn.DIV+1)}$ .

#### Bit 11 – OE: Output Enable

This bit is used to output the Generator clock output to the corresponding pin (GCLK\_IO), as long as GCLK\_IO is not defined as the Generator source in the GENCTRLn.SRC bit field.

Value	Description
0	No Generator clock signal on pin GCLK_IO.
1	The Generator clock signal is output on the corresponding GCLK_IO, unless GCLK_IO is selected as a generator source in the GENCTRLn.SRC bit field.

#### Bit 10 – OOV: Output Off Value

This bit is used to control the clock output value on pin (GCLK\_IO) when the Generator is turned off or the OE bit is zero, as long as GCLK\_IO is not defined as the Generator source in the GENCTRLn.SRC bit field.

Value	Description
0	The GCLK_IO will be LOW when generator is turned off or when the OE bit is zero.
1	The GCLK_IO will be HIGH when generator is turned off or when the OE bit is zero.

#### Bit 9 – IDC: Improve Duty Cycle

This bit is used to improve the duty cycle of the Generator output to 50/50 for odd division factors.

Value	Description
0	Generator output clock duty cycle is not balanced to 50/50 for odd division factors.
1	Generator output clock duty cycle is 50/50.

#### Bit 8 – GENEN: Generator Enable

This bit is used to enable and disable the Generator.

Value	Description
0	Generator is disabled.
1	Generator is enabled.

#### Bits 4:0 – SRC[4:0]: Generator Clock Source Selection

These bits select the Generator clock source, as shown in this table.

**Table 17-4. Generator Clock Source Selection**

Value	Name	Description
0x00	XOSC	XOSC oscillator output
0x01	GCLK_IN	Generator input pad (GCLK_IO)
0x02	GCLK_GEN1	Generic clock generator 1 output
0x03	OSCULP32K	OSCULP32K oscillator output
0x04	OSC32K	OSC32K oscillator output
0x05	XOSC32K	XOSC32K oscillator output
0x06	OSC48M	OSC16M oscillator output
0x07	DPLL96M	DPLL96M output
0x08-0x1F	Reserved	Reserved for future use

A Power Reset will reset all GENCTRLn registers. the Reset values of the GENCTRLn registers are shown in table below.

**Table 17-5. GENCTRLn Reset Value after a Power Reset**

GCLK Generator	Reset Value after a Power Reset
0	0x00000106
others	0x00000000

A User Reset will reset the associated GENCTRL register unless the Generator is the source of a locked Peripheral Channel (PCHCTRLm.WRTLOCK=1). The reset values of the GENCTRL register are as shown in the table below.

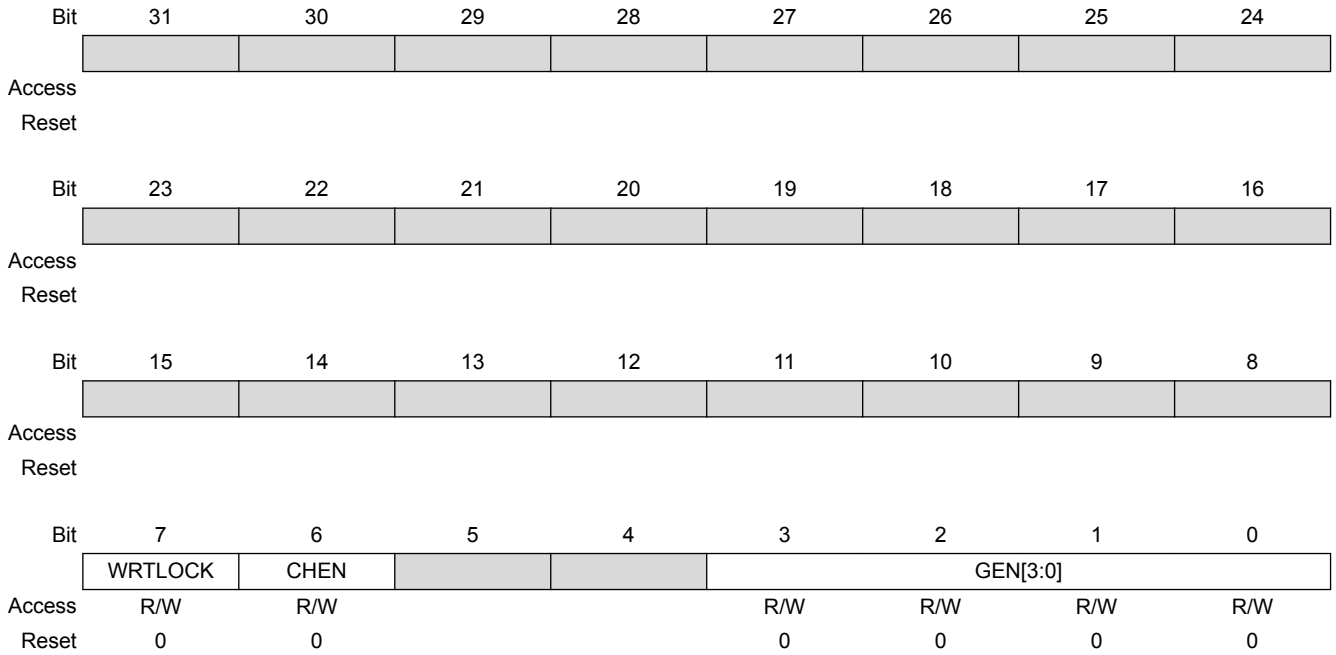
**Table 17-6. GENCTRLn Reset Value after a User Reset**

GCLK Generator	Reset Value after a User Reset
0	0x00000106
others	No change if the generator is used by a Peripheral Channel m with PCHCTRLm.WRTLOCK=1 else 0x00000000

### 17.8.4. Peripheral Channel Control

PCHCTRLm controls the settings of Peripheral Channel number m (m=0..34).

**Name:** PCHCTRLm  
**Offset:** 0x80 + 4\*m, (m= 0..40)  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



#### Bit 7 – WRTLOCK: Write Lock

After this bit is set to '1', further writes to the PCHCTRLm register will be discarded. The control register of the corresponding Generator n (GENCTRLn), as assigned in PCHCTRLm.GEN, will also be locked. It can only be unlocked by a Power Reset.

Note that Generator 0 cannot be locked.

Value	Description
0	The Peripheral Channel register and the associated Generator register are not locked
1	The Peripheral Channel register and the associated Generator register are locked

#### Bit 6 – CHEN: Channel Enable

This bit is used to enable and disable a Peripheral Channel.

Value	Description
0	The Peripheral Channel is disabled
1	The Peripheral Channel is enabled

#### Bits 3:0 – GEN[3:0]: Generator Selection

This bit field selects the Generator to be used as the source of a peripheral clock, as shown in the table below:

**Table 17-7. Generator Selection**

Value	Description
0x0	Generic Clock Generator 0
0x1	Generic Clock Generator 1
0x2	Generic Clock Generator 2
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

**Table 17-8. Reset Value after a User Reset or a Power Reset**

Reset	PCHCTRLm.GEN	PCHCTRLm.CHEN	PCHCTRLm.WRTLOCK
Power Reset	0x0	0x0	0x0
User Reset	If WRTLOCK = 0 : 0x0  If WRTLOCK = 1: no change	If WRTLOCK = 0 : 0x0  If WRTLOCK = 1: no change	No change

A Power Reset will reset all the PCHCTRLm registers.

A User Reset will reset a PCHCTRL if WRTLOCK=0, or else, the content of that PCHCTRL remains unchanged.

PCHCTRL register Reset values are shown in the table PCHCTRLm Mapping.

**Table 17-9. PCHCTRLm Mapping**

index(m)	Name	Description
0	GCLK_DPLL	FDPLL96M input clock source for reference
1	GCLK_DPLL_32K	FDPLL96M 32kHz clock for FDPLL96M internal clock timer
2	GCLK_EIC	EIC
3	GCLK_FREQM_MSR	FREQM Measure
4	GCLK_FREQM_REF	FREQM Reference
5	GCLK_TSENS	TSENS
6	GCLK_EVSYS_CHANNEL_0	EVSYS_CHANNEL_0
7	GCLK_EVSYS_CHANNEL_1	EVSYS_CHANNEL_1
8	GCLK_EVSYS_CHANNEL_2	EVSYS_CHANNEL_2

index(m)	Name	Description
9	GCLK_EVSYS_CHANNEL_3	EVSYS_CHANNEL_3
10	GCLK_EVSYS_CHANNEL_4	EVSYS_CHANNEL_4
11	GCLK_EVSYS_CHANNEL_5	EVSYS_CHANNEL_5
12	GCLK_EVSYS_CHANNEL_6	EVSYS_CHANNEL_6
13	GCLK_EVSYS_CHANNEL_7	EVSYS_CHANNEL_7
14	GCLK_EVSYS_CHANNEL_8	EVSYS_CHANNEL_8
15	GCLK_EVSYS_CHANNEL_9	EVSYS_CHANNEL_9
16	GCLK_EVSYS_CHANNEL_10	EVSYS_CHANNEL_10
17	GCLK_EVSYS_CHANNEL_11	EVSYS_CHANNEL_11
18	GCLK_SERCOM[0,1,2,3,4]_SLOW	SERCOM[0,1,2,3,4]_SLOW
19	GCLK_SERCOM0_CORE	SERCOM0_CORE
20	GCLK_SERCOM1_CORE	SERCOM1_CORE
21	GCLK_SERCOM2_CORE	SERCOM2_CORE
22	GCLK_SERCOM3_CORE	SERCOM3_CORE
23	GCLK_SERCOM4_CORE	SERCOM4_CORE
24	GCLK_SERCOM5_SLOW	SERCOM5_SLOW
25	GCLK_SERCOM5_CORE	SERCOM5_CORE
26	GCLK_CAN0	CAN0
27	GCLK_CAN1	CAN1
28	GCLK_TCC0, GCLK_TCC1	TCC0,TCC1
29	GCLK_TCC2	TCC2
30	GCLK_TC0, GCLK_TC1	TC0,TC1
31	GCLK_TC2, GCLK_TC3	TC2,TC3
32	GCLK_TC4	TC4
33	GCLK_ADC0	ADC0
34	GCLK_ADC1	ADC1
35	GCLK_SDADC	SDADC
36	GCLK_DAC	DAC
37	GCLK_PTC	PTC
38	GCLK_CCL	CCL
39	-	Reserved
40	GCLK_AC	AC

## 18. MCLK – Main Clock

### 18.1. Overview

The Main Clock (MCLK) controls the synchronous clock generation of the device.

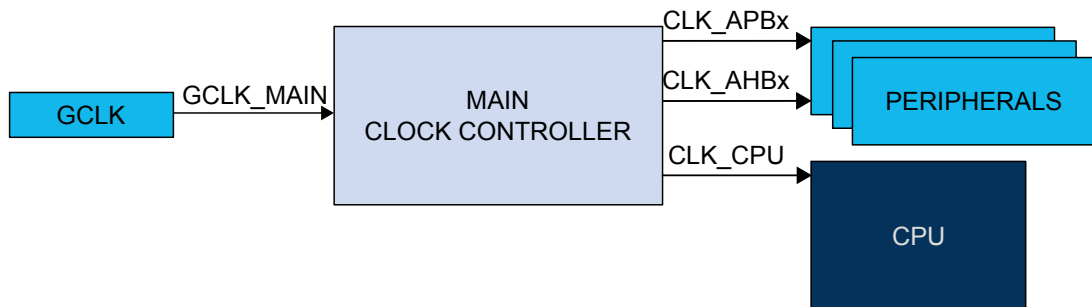
Using a clock provided by the Generic Clock Module (GCLK\_MAIN), the Main Clock Controller provides synchronous system clocks to the CPU and the modules connected to the AHBx and the APBx bus. The synchronous system clocks are divided into a number of clock domains. Each clock domain can run at different frequencies, enabling the user to save power by running peripherals at a relatively low clock frequency, while maintaining high CPU performance or vice versa. In addition, the clock can be masked for individual modules, enabling the user to minimize power consumption.

### 18.2. Features

- Generates CPU, AHB, and APB system clocks
  - Clock source 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

### 18.3. Block Diagram

Figure 18-1. MCLK Block Diagram



### 18.4. Signal Description

Not applicable.

### 18.5. Product Dependencies

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

#### 18.5.1. I/O Lines

Not applicable.

## 18.5.2. Power Management

The MCLK will operate in all sleep modes if a synchronous clock is required in these modes.

### Related Links

[PM – Power Manager](#) on page 162

## 18.5.3. Clocks

The MCLK bus clock (CLK\_MCLK\_APB) can be enabled and disabled in the Main Clock module, and the default state of CLK\_MCLK\_APB can be found in the Peripheral Clock Masking section. If this clock is disabled, it can only be re-enabled by a reset.

The Generic Clock GCLK\_MAIN is required to generate the Main Clocks. GCLK\_MAIN is configured in the Generic Clock Controller, and can be re-configured by the user if needed.

### Related Links

[GCLK - Generic Clock Controller](#) on page 114

[Peripheral Clock Masking](#) on page 135

### 18.5.3.1. Main Clock

The main clock GCLK\_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, AHBx, 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. APBx and AHBx Clock

The APBx clocks (CLK\_APBx) and the AHBx clocks (CLK\_AHBx) are the root clock source used by modules requiring a clock on the APBx and the AHBx bus. These clocks are always synchronous to the CPU clock, but can be divided by a prescaler, and can run even when the CPU clock is turned off in sleep mode. A clock gater is inserted after the common APB clock to gate any APBx clock of a module on APBx bus, as well as the AHBx clock.

### 18.5.3.4. Clock Domains

The device has these synchronous clock domains:

- CPU synchronous clock domain (CPU Clock Domain). Frequency is  $f_{CPU}$ .

See also the related links for the clock domain partitioning.

### Related Links

[Peripheral Clock Masking](#) on page 135

## 18.5.4. DMA

Not applicable.

## 18.5.5. Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the MCLK interrupt requires the Interrupt Controller to be configured first.

## 18.5.6. Events

Not applicable.



### 18.5.7. Debug Operation

When the CPU is halted in debug mode, the MCLK continues normal operation. In sleep mode, the clocks generated from the MCLK are kept running to allow the debugger accessing any module. As a consequence, power measurements are incorrect in debug mode.

### 18.5.8. 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 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](#) on page 46

### 18.5.9. Analog Connections

Not applicable.

## 18.6. Functional Description

### 18.6.1. Principle of Operation

The GCLK\_MAIN clock signal from the GCLK module is the source for the main clock, which in turn is the common root for the synchronous clocks for the CPU, APBx, and AHBx modules. The GCLK\_MAIN is divided by an 8-bit prescaler. Each of the derived clocks can run from any divided or undivided main clock, ensuring synchronous clock sources for each clock domain. The clock domain (CPU) can be changed on the fly to respond to variable load in the application. The clocks for each module in a clock domain can be masked individually to avoid power consumption in inactive modules. Depending on the sleep mode, some clock domains can be turned off.

### 18.6.2. Basic Operation

#### 18.6.2.1. Initialization

After a Reset, 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 prescaler division.

By default, only the necessary clocks are enabled.

#### Related Links

[Peripheral Clock Masking](#) on page 135

#### 18.6.2.2. Enabling, Disabling, and Resetting

The MCLK module is always enabled and cannot be reset.

#### 18.6.2.3. Selecting the Main Clock Source

Refer to the Generic Clock Controller description for details on how to configure the clock source of the GCLK\_MAIN clock.

#### Related Links

[GCLK - Generic Clock Controller](#) on page 114

#### 18.6.2.4. Selecting the Synchronous Clock Division Ratio

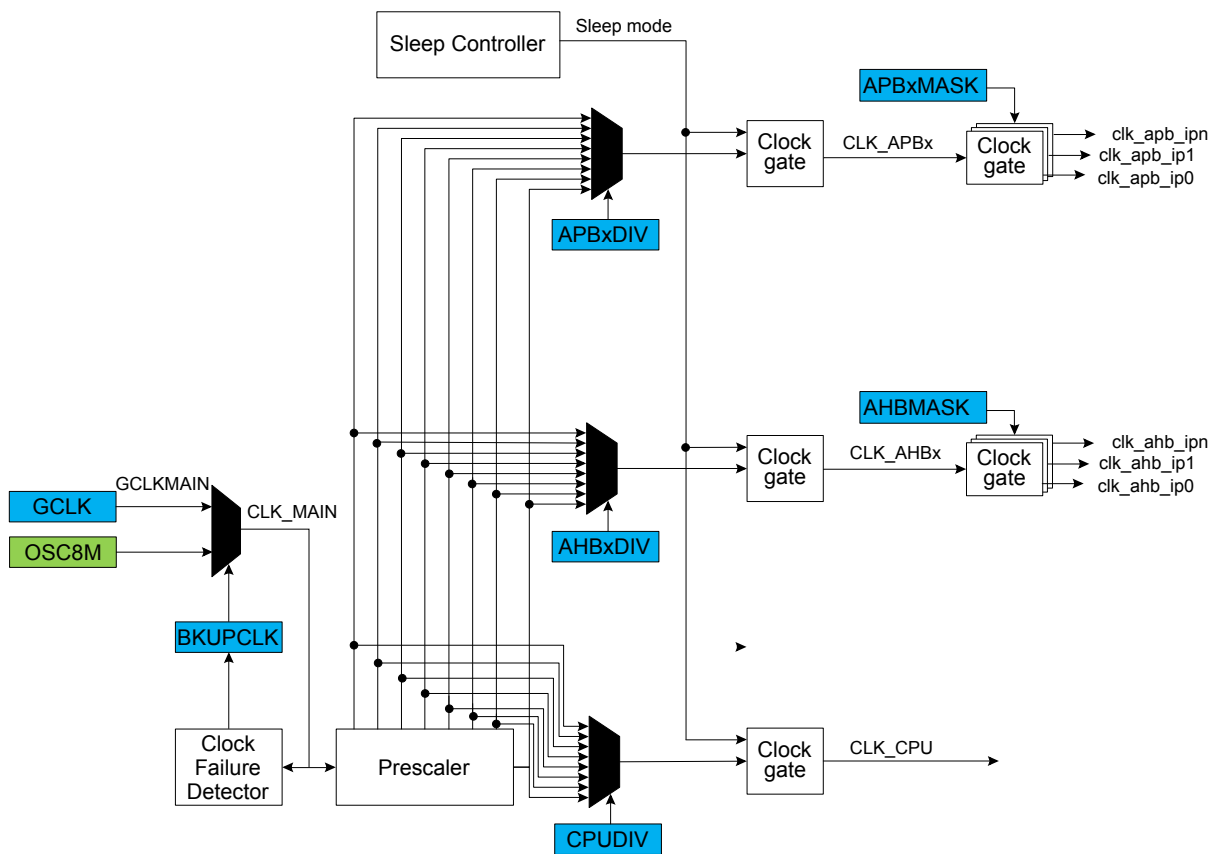
The main clock GCLK\_MAIN feeds an 8-bit prescaler, which can be used to generate the synchronous clocks. By default, the synchronous clocks run on the undivided main clock. The user can select a prescaler division for the CPU clock domain by writing the Division (DIV) bits in the CPU Clock Division register CPUDIV, resulting in a CPU clock domain frequency determined by this equation:

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

If the application attempts to write forbidden values in CPUDIV register, registers are written but these bad values are not used and a violation is reported to the PAC module.

Division bits (DIV) can be written without halting or disabling peripheral modules. Writing DIV bits allows a new clock setting to be written to all synchronous clocks belonging to the corresponding clock domain at the same time.

Figure 18-2. Synchronous Clock Selection and Prescaler



#### Related Links

[PAC - Peripheral Access Controller](#) on page 46

#### 18.6.2.5. Clock Ready Flag

There is a slight delay between writing to CPUDIV until the new clock settings become effective.

During this interval, the Clock Ready flag in the Interrupt Flag Status and Clear register (**INTFLAG.CKRDY**) will return zero when read. If **CKRDY** in the **INTENSET** register is set to '1', the Clock Ready interrupt will be triggered when the new clock setting is effective. The clock settings (**CLKCFG**) must not be re-written while **INTFLAG.CKRDY** reads '0'. The system may become unstable or hang, and a violation is reported to the PAC module.

## Related Links

[PAC - Peripheral Access Controller](#) on page 46

### 18.6.2.6. Peripheral Clock Masking

It is possible to disable/enable the AHB or APB clock for a peripheral by writing the corresponding bit in the Clock Mask registers (APBxMASK) to '0'/'1'. The default state of the peripheral clocks is shown here.

**Table 18-1. Peripheral Clock Default State**

CPU Clock Domain	
Peripheral Clock	Default State
CLK_AC_APB	Disabled
CLK_ADC0_APB	Disabled
CLK_ADC1_APB	Disabled
CLK_BRIDGE_A_AHB	Enabled
CLK_BRIDGE_B_AHB	Enabled
CLK_BRIDGE_C_AHB	Enabled
CLK_CAN0_AHB	Disabled
CLK_CAN1_AHB	Disabled
CLK_CCL_APB	Disabled
CLK_DAC_APB	Disabled
CLK_DIVAS_AHB	Enabled
CLK_DMAC_AHB	Enabled
CLK_DMAC_APB	Enabled
CLK_DSU_AHB	Enabled
CLK_DSU_APB	Enabled
CLK_EIC_APB	Enabled
CLK_EVSYN_APB	Disabled
CLK_FREQM_APB	Enabled
CLK_GCLK_AHB	Enabled
CLK_HAMATRIX_APB	Disabled
CLK_MCLK_APB	Enabled
CLK_MTB_APB	Enabled
CLK_NVMCTRL_AHB	Enabled
CLK_NVMCTRL_APB	Enabled
CLK_OSCCTRL_APB	Enabled
CLK_OSC32CTRL_APB	Enabled

CPU Clock Domain	
Peripheral Clock	Default State
CLK_PAC_AHB	Enabled
CLK_PAC_APB	Enabled
CLK_PM_APB	Enabled
CLK_PORT_APB	Enabled
CLK_PTC_APB	Disabled
CLK_RSTC_APB	Enabled
CLK_RTC_APB	Enabled
CLK_SDADC_APB	Disabled
CLK_SERCOM0_APB	Disabled
CLK_SERCOM1_AHB	Disabled
CLK_SERCOM2_APB	Disabled
CLK_SERCOM3_APB	Disabled
CLK_SERCOM4_APB	Disabled
CLK_SERCOM5_APB	Disabled
CLK_SUPC_APB	Enabled
CLK_TCC0_APB	Disabled
CLK_TCC1_APB	Disabled
CLK_TCC2_APB	Disabled
CLK_TC0_APB	Disabled
CLK_TC1_APB	Disabled
CLK_TC2_APB	Disabled
CLK_TC3_APB	Disabled
CLK_TC4_APB	Disabled
CLK_TSENS_APB	Disabled
CLK_WDT_APB	Enabled

When the APB clock is not provided to a module, its registers cannot be read or written. The module can be re-enabled later by writing the corresponding mask bit to '1'.

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

Note that clocks should only be switched off if it is certain that the module will not be used: Switching off the clock for the NVM Controller (NVMCTRL) will cause a problem if the CPU needs to read from the Flash Memory. Switching off the clock to the MCLK module (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.

### 18.6.3. DMA Operation

Not applicable.

### 18.6.4. Interrupts

The peripheral has the following interrupt sources:

- Clock Ready (CKRDY): indicates that CPU clocks are ready. This interrupt is a synchronous wake-up source.

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 enabled individually by writing a '1' to the corresponding enabling bit in the Interrupt Enable Set (**INTENSET**) register, and disabled by writing a '1' to the corresponding clearing 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 '1' 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. 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

[Sleep Mode Controller](#) on page 164

[PM – Power Manager](#) on page 162

### 18.6.5. Events

Not applicable.

### 18.6.6. Sleep Mode Operation

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

In STANDBY sleep mode, the MCLK is frozen if no synchronous clock is required.

## 18.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0								
0x01	INTENCLR	7:0								CKRDY
0x02	INTENSET	7:0								CKRDY
0x03	INTFLAG	7:0								CKRDY
0x04	Reserved									
0x05	CPUDIV	7:0	CPUDIV[7:0]							
0x06 ... 0x0F	Reserved									
0x10	AHBMASK	7:0	DMAC	HMCRAMCH S	NVMCTRL	HMATRIXHS	DSU	APBC	APBB	APBA
0x11		15:8				DIVAS	Reserved	PAC	CAN1	CAN0
0x12		23:16								
0x13		31:24								
0x14	APBAMASK	7:0	GCLK	SUPC	OSC32KCTR L	OSCCTRL	RSTC	MCLK	PM	PAC
0x15		15:8				TSENS	FREQM	EIC	RTC	WDT
0x16		23:16								
0x17		31:24								
0x18	APBBMASK	7:0			HMATRIXHS			NVMCTRL	DSU	PORT
0x19		15:8								
0x1A		23:16								
0x1B		31:24								
0x1C	APBCMASK	7:0		SERCOM5	SERCOM4	SERCOM3	SERCOM2	SERCOM1	SERCOM0	EVSYS
0x1D		15:8	TC3	TC2	TC1	TC0	TCC2	TCC1	TCC0	
0x1E		23:16	CCL	PTC	DAC	AC	SDADC	ADC1	ADC0	TC4
0x1F		31:24								

## 18.8. Register Description

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

Some registers can be write-protected optionally by the Peripheral Access Controller (PAC). This is denoted by the property "PAC Write-Protection" in each individual register description. Refer to the [Register Access Protection](#) on page 133 for details.

### 18.8.1. Control A

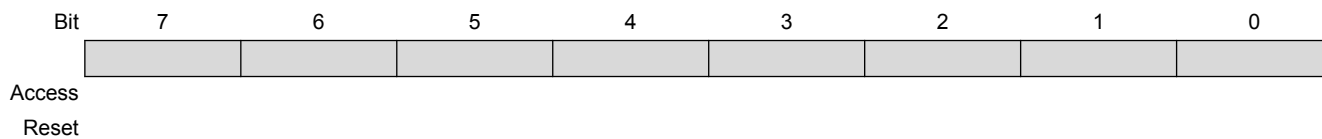
All bits in this register are reserved.

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x00

**Property:** PAC Write-Protection



### 18.8.2. Interrupt Enable Clear

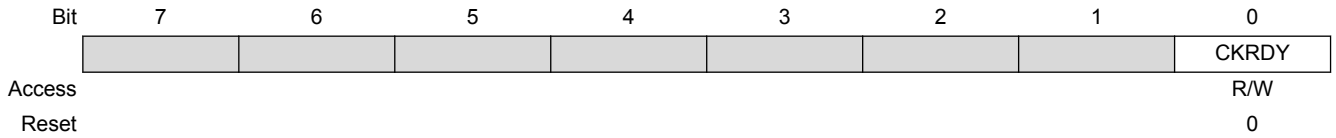
This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (INTENSET) register.

**Name:** INTENCLR

**Offset:** 0x01

**Reset:** 0x00

**Property:** PAC Write-Protection



#### Bit 0 – CKRDY: Clock Ready Interrupt Enable

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

Writing a '1' 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.



### 18.8.3. Interrupt Enable Set

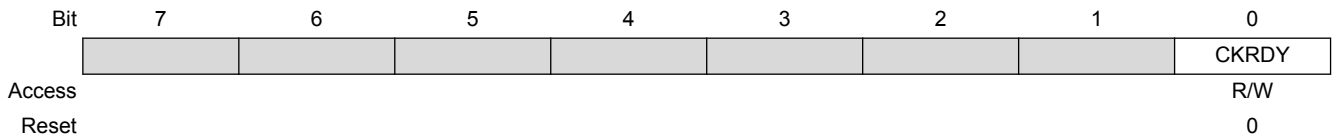
This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

**Name:** INTENSET

**Offset:** 0x02

**Reset:** 0x00

**Property:** PAC Write-Protection



#### Bit 0 – CKRDY: Clock Ready Interrupt Enable

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

Writing a '1' 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.

#### 18.8.4. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x03  
**Reset:** 0x01  
**Property:** –

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

##### **Bit 0 – CKRDY: Clock Ready**

This flag is cleared by writing a '1' to the flag.

This flag is set when the synchronous CPU, APBx, and AHBx clocks have frequencies as indicated in the CLKCFG registers and will generate an interrupt if [INTENCLR/SET.CKRDY](#) is '1'.

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

Writing a '1' to this bit clears the Clock Ready interrupt flag.

### 18.8.5. CPU Clock Division

**Name:** CPUDIV  
**Offset:** 0x05  
**Reset:** 0x01  
**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
	CPUDIV[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 – CPUDIV[7:0]: CPU Clock Division Factor

These bits define the division ratio of the main clock prescaler related to the CPU clock domain.

Frequencies must never exceed the specified maximum frequency for each clock domain.

Value	Name	Description
0x01	DIV1	Divide by 1
0x02	DIV2	Divide by 2
0x04	DIV4	Divide by 4
0x08	DIV8	Divide by 8
0x10	DIV16	Divide by 16
0x20	DIV32	Divide by 32
0x40	DIV64	Divide by 64
0x80	DIV128	Divide by 128
others	-	Reserved

## 18.8.6. AHB Mask

**Name:** AHBMASK  
**Offset:** 0x10  
**Reset:** 0x000001CFF  
**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								
Reset								
Bit	15	14	13	12	11	10	9	8
Access				DIVAS	Reserved	PAC	CAN1	CAN0
Reset				R/W	R	R/W	R/W	R/W
Reset				1	1	1	0	0
Bit	7	6	5	4	3	2	1	0
Access	DMAC	HMCGRAMCHS	NVMCTRL	HMATRIXHS	DSU	APBC	APBB	APBA
Reset	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 12 – DIVAS: DIVAS AHB Clock Enable

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

### Bit 11 – Reserved

### Bit 10 – PAC: PAC AHB Clock Enable

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

### Bit 9 – CAN1: CAN1 AHB Clock Enable

Value	Description
0	The AHB clock for the CAN1 is stopped.
1	The AHB clock for the CAN1 is enabled.

### Bit 8 – CAN0: CAN0 AHB Clock Enable

Value	Description
0	The AHB clock for the CAN0 is stopped.
1	The AHB clock for the CAN0 is enabled.

#### Bit 7 – DMAC: DMAC AHB Clock Enable

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

#### Bit 6 – HMCGRAMCHS: HMCGRAMCHS AHB Clock Enable

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

#### Bit 5 – NVMCTRL: NVMCTRL AHB Clock Enable

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

#### Bit 4 – HMATRIXHS: HMATRIXHS AHB Clock Enable

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

#### Bit 3 – DSU: DSU AHB Clock Enable

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

#### Bit 2 – APBC: APBC AHB Clock Enable

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

#### Bit 1 – APBB: APBB AHB Clock Enable

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

#### Bit 0 – APBA: APBA AHB Clock Enable

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

### 18.8.7. APBA Mask

**Name:** APBAMASK  
**Offset:** 0x14  
**Reset:** 0x00000FFF  
**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								
Reset								
Bit	15	14	13	12	11	10	9	8
Access				TSENS	FREQM	EIC	RTC	WDT
Reset				R/W	R/W	R/W	R/W	R/W
Reset				0	1	1	1	1
Bit	7	6	5	4	3	2	1	0
Access	GCLK	SUPC	OSC32KCTRL	OSCCTRL	RSTC	MCLK	PM	PAC
Reset	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 12 – TSENS: TSENS APBA Clock Enable

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

#### Bit 11 – FREQM: FREQM APBA Clock Enable

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

#### Bit 10 – EIC: EIC APBA Clock Enable

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

#### Bit 9 – RTC: RTC APBA Clock Enable

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

#### Bit 8 – WDT: WDT APBA Clock Enable

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

#### Bit 7 – GCLK: GCLK APBA Clock Enable

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

#### Bit 6 – SUPC: SUPC APBA Clock Enable

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

#### Bit 5 – OSC32KCTRL: OSC32KCTRL APBA Clock Enable

Value	Description
0	The APBA clock for the OSC32KCTRL is stopped.
1	The APBA clock for the OSC32KCTRL is enabled.

#### Bit 4 – OSCCTRL: OSCCTRL APBA Clock Enable

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

#### Bit 3 – RSTC: RSTC APBA Clock Enable

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

#### Bit 2 – MCLK: MCLK APBA Clock Enable

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



**Bit 1 – PM: PM APBA Clock Enable**

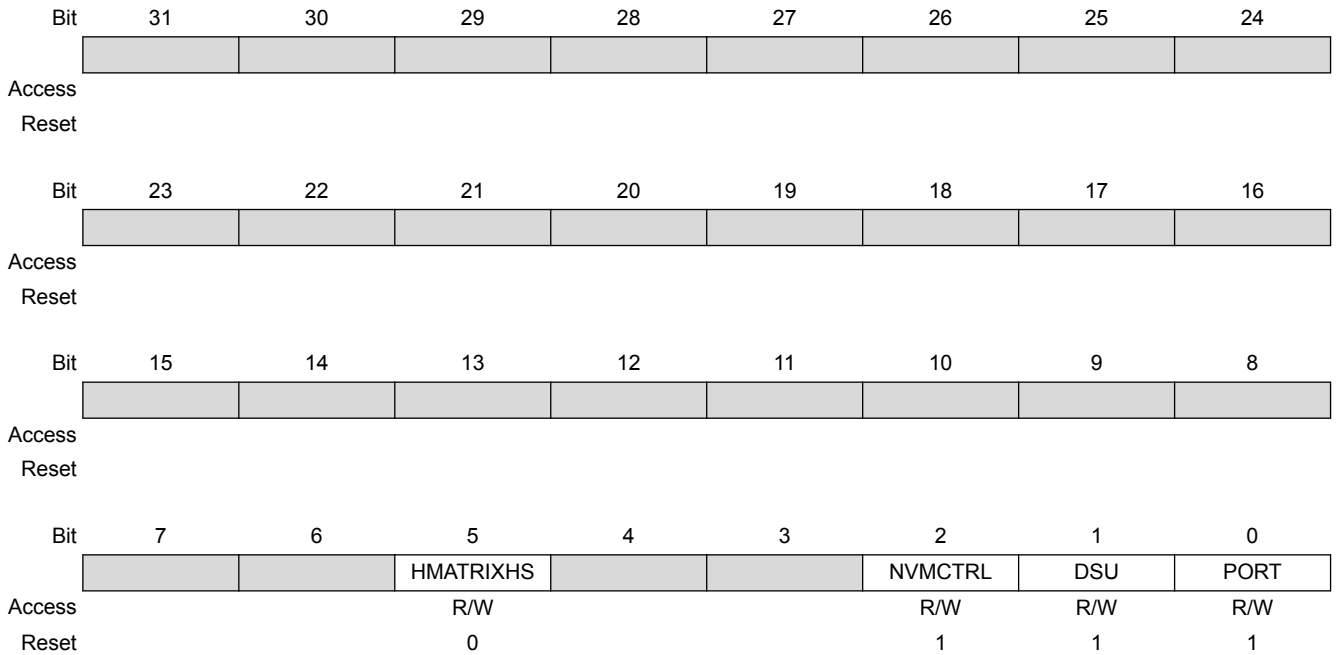
Value	Description
0	The APBA clock for the PM is stopped.
1	The APBA clock for the PM is enabled.

**Bit 0 – PAC: PAC APBA Clock Enable**

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

### 18.8.8. APBB Mask

**Name:** APBBMASK  
**Offset:** 0x18  
**Reset:** 0x00000007  
**Property:** PAC Write-Protection



#### Bit 5 – HMATRIXHS: HMATRIXHS APBB Clock Enable

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

#### Bit 2 – NVMCTRL: NVMCTRL APBB 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 APBB Clock Enable

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

#### Bit 0 – PORT: PORT APBB Clock Enable

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

## 18.8.9. APBC Mask

**Name:** APBCMASK  
**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
Access	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	R/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
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 – CCL: CCL APBC Clock Enable

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

### Bit 22 – PTC: PTC APBC Mask Clock Enable

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

### Bit 21 – DAC: DAC APBC Mask Clock Enable

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

### Bit 20 – AC: AC APBC Clock Enable

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

**Bit 19 – SDADC: SDADC APBC Clock Enable**

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

**Bit 18 – ADC1: ADC1 APBC Clock Enable**

Value	Description
0	The APBC clock for the ADC1 is stopped.
1	The APBC clock for the ADC1 is enabled.

**Bit 17 – ADC0: ADC0 APBC Clock Enable**

Value	Description
0	The APBC clock for the ADC0 is stopped.
1	The APBC clock for the ADC0 is enabled.

**Bit 16 – TC4: TC4 APBC Mask Clock Enable**

**Bit 15 – TC3: TC3 APBC Mask Clock Enable**

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

**Bit 14 – TC2: TC2 APBC Mask Clock Enable**

Value	Description
0	The APBC clock for the TC2 is stopped.
1	The APBC clock for the TC2 is enabled.

**Bit 13 – TC1: TC1 APBC Mask Clock Enable**

Value	Description
0	The APBC clock for the TC1 is stopped.
1	The APBC clock for the TC1 is enabled.

**Bit 12 – TC0: TC0 APBC Mask Clock Enable**

Value	Description
0	The APBC clock for the TC0 is stopped.
1	The APBC clock for the TC0 is enabled.

**Bit 11 – TCC2: TCC2 APBC Mask Clock Enable**

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

**Bit 10 – TCC1: TCC1 APBC Mask Clock Enable**

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

**Bit 9 – TCC0: TCC0 APBC Mask Clock Enable**

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

**Bit 6 – SERCOM5: SERCOM5 APBC Mask Clock Enable**

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

**Bit 5 – SERCOM4: SERCOM4 APBC Mask Clock Enable**

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

**Bit 4 – SERCOM3: SERCOM3 APBC Mask Clock Enable**

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

**Bit 3 – SERCOM2: SERCOM2 APBC Mask Clock Enable**

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

**Bit 2 – SERCOM1: SERCOM1 APBC Mask Clock Enable**

Value	Description
0	The APBC clock for the SERCOM1 is stopped.
1	The APBC clock for the SERCOM1 is enabled.

**Bit 1 – SERCOM0: SERCOM0 APBC Mask Clock Enable**

Value	Description
0	The APBC clock for the SERCOM0 is stopped.
1	The APBC clock for the SERCOM0 is enabled.

**Bit 0 – EVSYS: EVSYS APBC Clock Enable**

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

## 19. RSTC – Reset Controller

### 19.1. Overview

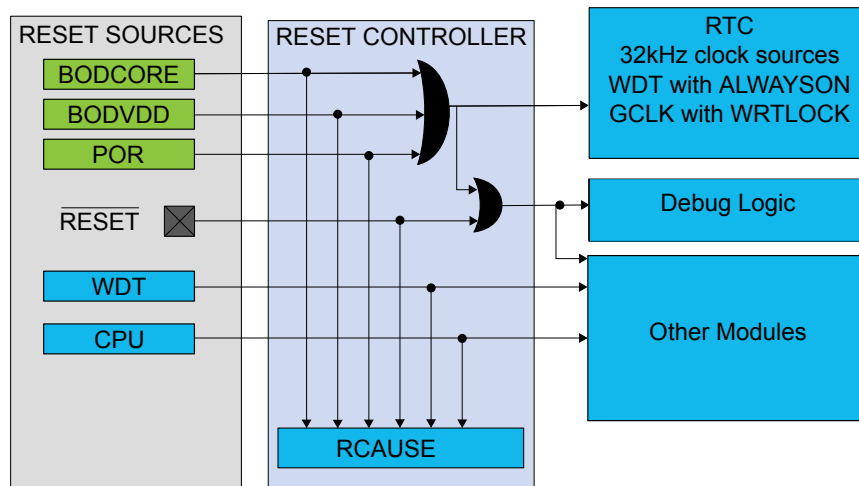
The Reset Controller (RSTC) manages the reset of the microcontroller. It issues a microcontroller reset, sets the device to its initial state and allows the reset source to be identified by software.

### 19.2. Features

- Reset the microcontroller and set it to an initial state according to the reset source
- Reset cause register for reading the reset source from the application code
- Multiple reset sources
  - Power supply reset sources: POR, BODCORE, BODVDD
  - User reset sources: External reset ( $\overline{\text{RESET}}$ ), Watchdog reset, and System Reset Request

### 19.3. Block Diagram

Figure 19-1. Reset System



### 19.4. Signal Description

Signal Name	Type	Description
$\overline{\text{RESET}}$	Digital input	External reset

One signal can be mapped on several pins.

#### Related Links

[I/O Multiplexing and Considerations](#) on page 27

### 19.5. Product Dependencies

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



### 19.5.1. I/O Lines

Not applicable.

### 19.5.2. Power Management

The Reset Controller module is always on.

### 19.5.3. Clocks

The RSTC bus clock (CLK\_RSTC\_APB) can be enabled and disabled in the Main Clock Controller.

#### Related Links

[MCLK – Main Clock](#) on page 131

[Peripheral Clock Masking](#) on page 135

### 19.5.4. DMA

Not applicable.

### 19.5.5. Interrupts

Not applicable.

### 19.5.6. Events

Not applicable.

### 19.5.7. Debug Operation

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

### 19.5.8. 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.

When the CPU is halted in debug mode, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger.

### 19.5.9. Analog Connections

Not applicable.

## 19.6. Functional Description

### 19.6.1. Principle of Operation

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

### 19.6.2. Basic Operation

#### 19.6.2.1. Initialization

After a power-on Reset, the RSTC is enabled and the Reset Cause (RCAUSE) register indicates the POR source.

#### 19.6.2.2. Enabling, Disabling, and Resetting

The RSTC module is always enabled.

### 19.6.2.3. Reset Causes and Effects

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

These are the groups of Reset sources:

- Power supply Reset: Resets caused by an electrical issue. It covers POR and BODs Resets
- User Reset: Resets caused by the application. It covers external Resets, system Reset requests and watchdog Resets

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

**Table 19-1. Effects of the Different Reset Causes**

	Power Supply Reset		User Reset		Backup Reset
	POR, BODVDD	BODCORE	External Reset	WDT Reset, System Reset Request	RTC, BBPS
RTC, OSC32KCTRL, RSTC, CTRLA.IORET bit of PM	Y	N	N	N	N
GCLK with WRTLOCK	Y	Y	N	N	Y
Debug logic	Y	Y	Y	N	Y
Others	Y	Y	Y	Y	N

The external Reset is generated when pulling the  $\overline{\text{RESET}}$  pin low.

The POR, BODCORE, and BODVDD Reset sources are generated by their corresponding module in the Supply Controller Interface (SUPC).

The WDT Reset is generated by the Watchdog Timer.

The System Reset Request is a Reset generated by the CPU when asserting the SYSRESETREQ bit located in the Reset Control register of the CPU (for details refer to the ARM<sup>®</sup> Cortex<sup>™</sup> Technical Reference Manual on <http://www.arm.com>).

#### Related Links

[SUPC – Supply Controller](#) on page 237

### 19.6.3. Additional Features

Not applicable.

### 19.6.4. DMA Operation

Not applicable.

### 19.6.5. Interrupts

Not applicable.

### 19.6.6. Events

Not applicable.

### 19.6.7. Sleep Mode Operation

The RSTC module is active in all sleep modes.

## 19.7. Register Summary

Offset	Name	Bit Pos.								
0x00	<a href="#">RCAUSE</a>	7:0		SYST	WDT	EXT		BODVDD	BODCORE	POR
0x01	Reserved									
0x02	<a href="#">BKUPEXIT</a>	7:0						BBPS	RTC	

## 19.8. Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#) on page 157.

### 19.8.1. Reset Cause

When a Reset occurs, the bit corresponding to the Reset source is set to '1' and all other bits are written to '0'.

**Name:** RCAUSE  
**Offset:** 0x00  
**Reset:** Latest Reset Source  
**Property:** –

Bit	7	6	5	4	3	2	1	0
		SYST	WDT	EXT		BODVDD	BODCORE	POR
Access		R	R	R		R	R	R
Reset		x	x	x		x	x	x

#### Bit 6 – SYST: System Reset Request

This bit is set if a System Reset Request has occurred. Refer to the Cortex processor documentation for more details.

#### Bit 5 – WDT: Watchdog Reset

This bit is set if a Watchdog Timer Reset has occurred.

#### Bit 4 – EXT: External Reset

This bit is set if an external Reset has occurred.

#### Bit 2 – BODVDD: Brown Out VDD Detector Reset

This bit is set if a BODVDD Reset has occurred.

#### Bit 1 – BODCORE: Brown Out CORE Detector Reset

This bit is set if a BODCORE Reset has occurred.

#### Bit 0 – POR: Power On Reset

This bit is set if a POR has occurred.

## 20. PM – Power Manager

### 20.1. Overview

The Power Manager (PM) controls the sleep modes of the device.

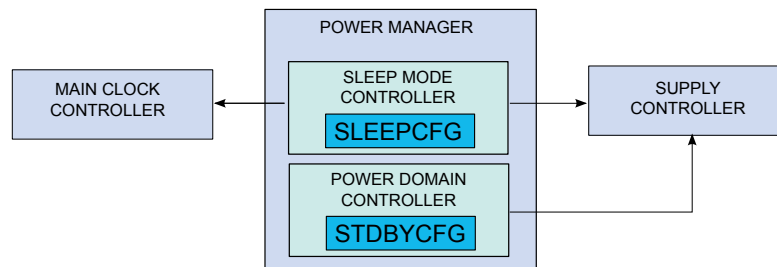
Various sleep modes are provided in order to fit power consumption requirements. This enables the PM to stop unused modules in order to save power. In active mode, the CPU is executing application code. When the device enters a sleep mode, program execution is stopped and some modules and clock domains 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.

### 20.2. Features

- Power management control
  - Sleep modes: Idle, Standby

### 20.3. Block Diagram

Figure 20-1. PM 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. Clocks

The PM bus clock (CLK\_PM\_APB) can be enabled and disabled in the Main Clock module. If this clock is disabled, it can only be re-enabled by a system reset.

#### 20.5.3. DMA

Not applicable.

#### 20.5.4. Interrupts

The interrupt request line is connected to the interrupt controller. Using the PM interrupt requires the interrupt controller to be configured first.

#### 20.5.5. Events

Not applicable.

#### 20.5.6. Debug Operation

When the CPU is halted in debug mode, the PM continues normal operation. If standby sleep mode is requested by the system while in debug mode, the power domains are not turned off. As a consequence, power measurements while in debug mode are not relevant.

Hot plugging in standby mode is supported.

#### 20.5.7. Register Access Protection

Registers with write-access can be write-protected optionally by the peripheral access controller (PAC).

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.

#### 20.5.8. Analog Connections

Not applicable.

### 20.6. Functional Description

#### 20.6.1. Terminology

The following is a list of terms used to describe the Power Management features of this microcontroller.

##### 20.6.1.1. Sleep Modes

The device can be set in a sleep mode. In sleep mode, the CPU is stopped and the peripherals are either active or idle, according to the sleep mode depth:

- Idle sleep mode: The CPU is stopped. Synchronous clocks are stopped except when requested. The logic is retained.
- Standby sleep mode: The CPU is stopped as well as the peripherals.

#### 20.6.2. Principle of Operation

In active mode, all clock domains and power domains are active, allowing software execution and peripheral operation. The PM Sleep Mode Controller allows to save power by choosing between different sleep modes depending on application requirements, see [Sleep Mode Controller](#) on page 164.

The PM Power Domain Controller allows to reduce the power consumption in standby mode even further.

#### 20.6.3. Basic Operation

##### 20.6.3.1. Initialization

After a power-on reset, the PM is enabled, the device is in ACTIVE mode.

##### 20.6.3.2. Enabling, Disabling and Resetting

The PM is always enabled and can not be reset.

### 20.6.3.3. Sleep Mode Controller

A Sleep mode is entered by executing the Wait For Interrupt instruction (WFI). The Sleep Mode bits in the Sleep Configuration register (SLEEPCFG.SLEEPMODE) select the level of the sleep mode.

**Note:** A small latency happens between the store instruction and actual writing of the SLEEPCFG register due to bridges. Software must ensure that the SLEEPCFG register reads the desired value before issuing a WFI instruction.

**Table 20-1. Sleep Mode Entry and Exit Table**

Mode	Mode Entry	Wake-Up Sources
IDLE	SLEEPCFG.SLEEPMODE = IDLE _n	Synchronous <sup>(2)</sup> (APB, AHB), asynchronous <sup>(1)</sup>
STANDBY	SLEEPCFG.SLEEPMODE = STANDBY	Synchronous <sup>(3)</sup> , Asynchronous

**Note:**

1. Asynchronous: interrupt generated on generic clock, external clock, or external event.
2. Synchronous: interrupt generated on the APB clock.
3. Synchronous interrupt only for peripherals configured to run in standby.

**Note:** The type of wake-up sources (synchronous or asynchronous) is given in each module interrupt section.

The sleep modes (idle, standby) and their effect on the clocks activity, the regulator and the NVM state are described in the table and the sections below.

**Table 20-2. Sleep Mode Overview**

Mode	CPU clock	AHB clock	APB clock	Main clock	GCLK clocks	Oscillators		Regulator	RAM
						ONDEMAND = 0	ONDEMAND = 1		
IDLE 0	Stop	Run	Run	Run	Run <sup>(1)</sup>	Run	Run if requested	MAIN	Normal
IDLE 1	Stop	Stop <sup>(2)</sup>	Run	Run	Run <sup>(1)</sup>	Run	Run if requested	MAIN	Normal
IDLE 2	Stop	Stop <sup>(2)</sup>	Stop <sup>(2)</sup>	Run	Run <sup>(1)</sup>	Run	Run if requested	Main	Normal
STANDBY	Stop	Stop <sup>(2)</sup>	Stop <sup>(2)</sup>	Stop	Stop <sup>(2)</sup>	Run if requested or RUNSTDBY=1	Run if requested	LPVREG <sup>(3)</sup>	Low power <sup>(4)</sup>

**Note:**

1. Running if requested by peripheral.
2. Running during SleepWalking.
3. Regulator state is programmable by using STDBYCFG.VREGSMOD bits.
4. RAM state is programmable by using STDBYCFG.BBIASHS bit.

**IDLE Mode**

The IDLE mode allows 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 SLEEPCFG bit group to IDLE level 1 or 2. The peripheral will be halted regardless of the bit settings of the mask registers in the MCLK (MCLK.AHBMASK, MCLK.APBxMASK).



- 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 Sleep Configuration register.
- Exiting IDLE mode: The processor wakes the system up when it detects any non-masked interrupt with sufficient priority to cause exception entry. The system goes back to the ACTIVE mode. The CPU and affected modules are restarted.

Regulator operates in normal mode.

### STANDBY Mode

The STANDBY mode is the lowest power configuration while keeping the state of the logic and the content of the RAM.

In this mode, all clocks are stopped except those which are kept running if requested by a running peripheral or have the ONDEMAND bit written to "0". For example, the RTC can operate in STANDBY mode. In this case, its GCLK clock source will also be enabled.

All features that don't require CPU intervention are supported in STANDBY mode. Here are examples:

- Autonomous peripherals features.
- Features relying on Event System allowing autonomous communication between peripherals.
- Features relying on on-demand clock.
- DMA transfers.
- Entering STANDBY mode: This mode is entered by executing the WFI instruction with the SLEEPCFG register written to STANDBY. The SLEEPONEXIT feature is also available as in IDLE mode.
- Exiting STANDBY mode: Any peripheral able to generate an asynchronous interrupt can wake up the system. For example, a peripheral running on a GCLK 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.

Depending on the configuration of these modules, the current consumption of the device in STANDBY mode can be slightly different.

The regulator operates in low-power mode (LP VREG) by default and can switch automatically to the main regulator if a task required by a peripheral requires more power. It returns automatically in the low power mode as soon as the task is completed.

## 20.6.4. Advanced Features

### 20.6.4.1. RAM Automatic Low Power Mode

The RAM is by default put in low power mode (back-biased) if the device is in standby sleep mode.

This behavior can be changed by configuring the Back Bias bit in the Standby Configuration register (STDBYCFG.BBIASHS), refer to the table below for details.

**Note:** In standby sleep mode, the RAM is put in low-power mode by default. This means that the RAM is back-biased, and the DMAC cannot access it. The DMAC can only access the RAM when it is not back biased (PM.STDBYCFG.BBIASxx=0x0).

**Table 20-3. RAM Back-Biasing Mode**

STBYCFG.BBIASHS		RAM
0x0	No Back Biasing	RAM is not back-biased if the device is in standby sleep mode.
0x1	Standby Back Biasing mode	RAM is back-biased if the device is in standby sleep mode.

**20.6.4.2. Regulator Automatic Low Power Mode**

In standby mode, the PM selects either the main or the low power voltage regulator to supply the VDDCORE. By default the low power voltage regulator is used.

If a sleepwalking task is working on either asynchronous clocks (generic clocks) or synchronous clock (APB/AHB clocks), the main voltage regulator is used. This behavior can be changed by writing the Voltage Regulator Standby Mode bits in the Standby Configuration register (STBYCFG.VREGSMOD). Refer to the following table for details.

**Table 20-4. Regulator State in Sleep Mode**

Sleep Mode	STBYCFG.VREGSMOD	SleepWalking	Regulator state for VDDCORE
Active	-	-	main voltage regulator
Idle	-	-	main voltage regulator
Standby	0x0: AUTO	NO	low power regulator
		YES	main voltage regulator
	0x1: PERFORMANCE	-	main voltage regulator
	0x2: LP	-	low power regulator

**20.6.5. DMA Operation**

Not applicable.

**20.6.6. Interrupts**

Not applicable.

**20.6.7. Events**

Not applicable.

**20.6.8. Sleep Mode Operation**

The Power Manager is always active.

## 20.7. Register Summary

Offset	Name	Bit Pos.							
0x01	SLEEPCFG	7:0						SLEEPMODE[2:0]	
0x02	Reserved								
...									
0x07									
0x08	STDBYCFG	7:0	VREGSMOD[1:0]						
0x09		15:8					BBIASHS		

## 20.8. Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#) on page 163.

## 20.8.1. Sleep Configuration

**Name:** SLEEPCFG  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

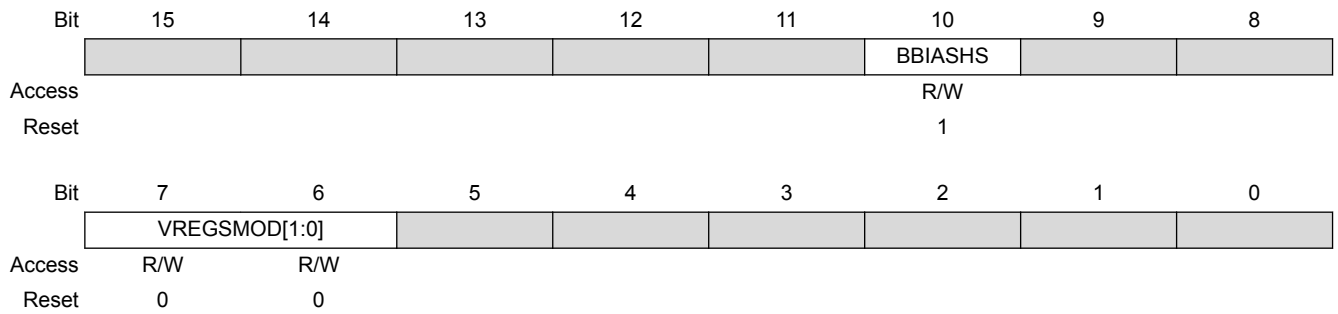
### Bits 2:0 – SLEEPMODE[2:0]: Sleep Mode

**Note:** A small latency happens between the store instruction and actual writing of the SLEEPCFG register due to bridges. Software has to make sure the SLEEPCFG register reads the wanted value before issuing Wait For Interrupt (WFI) instruction.

Value	Name	Definition
0x0	IDLE0	CPU clock is OFF
0x1	IDLE1	AHB clock is OFF
0x2	IDLE2	APBx clocks are OFF
0x3	Reserved	Reserved
0x4	STANDBY	ALL clocks are OFF
0x5 - 0x7	Reserved	Reserved

## 20.8.2. Standby Configuration

**Name:** STDBYCFG  
**Offset:** 0x08  
**Reset:** 0x0400  
**Property:** PAC Write-Protection



### Bit 10 – BBIASHS: Back Bias for HMC RAMCHS

Refer to [RAM Automatic Low Power Mode](#) on page 165 for details.

Value	Description
0	No Back Biasing Mode
1	Standby Back Biasing Mode

### Bits 7:6 – VREGSMOD[1:0]: VREG Switching Mode

Refer to for [Regulator Automatic Low Power Mode](#) on page 166 details.

Value	Name	Description
0x0	AUTO	Automatic Mode
0x1	PERFORMANCE	Performance oriented
0x2	LP	Low Power consumption oriented
0x9	Reserved	Reserved

## 21. OSCCTRL – Oscillators Controller

### 21.1. Overview

The Oscillators Controller (OSCCTRL) provides a user interface to the XOSC, OSC48M, and FDPLL96M.

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

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

#### Related Links

[INTENCLR](#) on page 184

[INTENSET](#) on page 187

[INTFLAG](#) on page 190

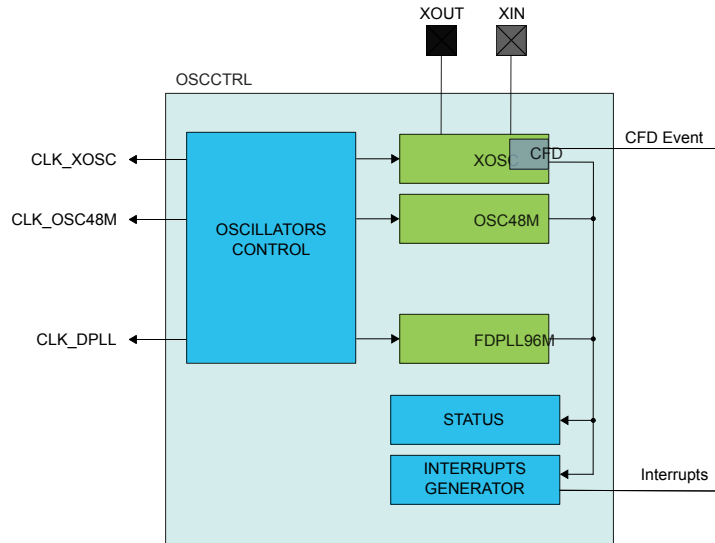
[STATUS](#) on page 192

### 21.2. Features

- 0.4-32MHz Crystal Oscillator (XOSC)
  - Tunable gain control
  - Programmable start-up time
  - Crystal or external input clock on XIN I/O
  - Clock failure detection with safe clock switch
  - Clock failure event output
- 48MHz Internal Oscillator (OSC48M)
  - Fast start-up
  - Programmable start-up time
  - 4-bit linear divider available
- Fractional Digital Phase Locked Loop (FDPLL96M)
  - 48MHz to 96MHz output frequency
  - 32kHz to 2MHz reference clock
  - A selection of sources for the reference clock
  - Adjustable proportional integral controller
  - Fractional part used to achieve 1/16th of reference clock step

## 21.3. Block Diagram

Figure 21-1. OSCCTRL Block Diagram



## 21.4. Signal Description

Signal	Description	Type
XIN	Multipurpose Crystal Oscillator or external clock generator input	Analog input
XOUT	Multipurpose Crystal Oscillator output	Analog output

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

## 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

I/O lines are configured by OSCCTRL when XOSC is enabled, and need no user configuration.

### 21.5.2. Power Management

The OSCCTRL can continue to operate in any sleep mode where the selected source clock is running. The OSCCTRL 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](#) on page 162

### 21.5.3. Clocks

The OSCCTRL gathers controls for all device oscillators and provides clock sources to the Generic Clock Controller (GCLK). The available clock sources are: XOSC, OSC48M, and FDPLL96M.

The OSCCTRL bus clock (CLK\_OSCCTRL\_APB) can be enabled and disabled in the Main Clock module (MCLK).

The OSC48M control logic uses the oscillator output, which is also asynchronous to the user interface clock (CLK\_OSCCTRL\_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) on page 180 for further details.

#### Related Links

[MCLK – Main Clock](#) on page 131

[Peripheral Clock Masking](#) on page 135

#### 21.5.4. DMA

Not applicable.

#### 21.5.5. Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the OSCCTRL interrupts requires the interrupt controller to be configured first.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

[INTFLAG](#) on page 190

[Sleep Mode Controller](#) on page 164

#### 21.5.6. Events

The events of this peripheral are connected to the Event System.

#### Related Links

[EVSYS – Event System](#) on page 474

#### 21.5.7. Debug Operation

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

#### 21.5.8. 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 register (INTFLAG)

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

When the CPU is halted in debug mode, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger.

#### 21.5.9. Analog Connections

The 0.4-32MHz crystal must be connected between the XIN and XOUT pins, along with any required load capacitors.

## 21.6. Functional Description

### 21.6.1. Principle of Operation

XOSC, OSC48M, and FDPLL96M are configured via OSCCTRL control registers. Through this interface, the sub-peripherals are enabled, disabled, or have their calibration values updated.



The Status register gathers different status signals coming from the sub-peripherals controlled by the OSCCTRL. 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.

## 21.6.2. 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. This is configured by the 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 OSCCTRL, and GPIO functions are overridden on both pins. When in external clock mode, only the XIN pin will be overridden and controlled by the OSCCTRL, while the XOUT pin can still be used as a GPIO pin.

The XOSC is enabled by writing a '1' to the Enable bit in the External Multipurpose Crystal Oscillator Control register (XOSCCTRL.ENABLE).

To enable XOSC as an external crystal oscillator, the XTAL Enable bit (XOSCCTRL.XTALEN) must be written to '1'. If XOSCCTRL.XTALEN is zero, the external clock input on XIN will be enabled.

When in crystal oscillator mode (XOSCCTRL.XTALEN=1), the External Multipurpose Crystal Oscillator Gain (XOSCCTRL.GAIN) must be set to match the external crystal oscillator frequency. If the External Multipurpose Crystal Oscillator Automatic Amplitude Gain Control (XOSCCTRL.AMPGC) is '1', 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 XOSCCTRL.RUNSTDBY, XOSCCTRL.ONDEMAND, and XOSCCTRL.ENABLE. If XOSCCTRL.ENABLE=0, the XOSC will be always stopped. For XOSCCTRL.ENABLE=1, this table is valid:

**Table 21-1. XOSC Sleep Behavior**

CPU Mode	XOSCCTRL.RUNSTDBY	XOSCCTRL.ONDEMAND	Sleep Behavior
Active or Idle	-	0	Always run
Active or Idle	-	1	Run if requested by peripheral
Standby	1	0	Always run
Standby	1	1	Run if requested by peripheral
Standby	0	-	Run if requested by 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 (XOSCCTRL.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 Status register (STATUS.XOSCRDY) is set once the external clock or crystal oscillator is stable and ready to be used as a clock source. An interrupt is generated on a zero-to-one transition on STATUS.XOSCRDY if the External Multipurpose Crystal Oscillator Ready bit in the Interrupt Enable Set register (INTENSET.XOSCRDY) is set.

#### Related Links

[GCLK - Generic Clock Controller](#) on page 114

### 21.6.3. Clock Failure Detection Operation

The Clock Failure Detector (CFD) allows the user to monitor the external clock or crystal oscillator signal provided by the external oscillator (XOSC). The CFD detects failing operation of the XOSC clock with reduced latency, and allows to switch to a safe clock source in case of clock failure. The user can also switch from the safe clock back to XOSC in case of recovery. The safe clock is derived from the OSC48M oscillator with a configurable prescaler. This allows to configure the safe clock in order to fulfill the operative conditions of the microcontroller.

In sleep modes, CFD operation is automatically disabled when the external oscillator is not requested to run by a peripheral. See the Sleep Behavior table above when this is the case.

The user interface registers allow to enable, disable, and configure the CFD. The Status register provides status flags on failure and clock switch conditions. The CFD can optionally trigger an interrupt or an event when a failure is detected.

#### Clock Failure Detection

The CFD is reset only at power-on (POR). The CFD does not monitor the XOSC clock when the oscillator is disabled (XOSCCTRL.ENABLE=0).

Before starting CFD operation, the user must start and enable the safe clock source (OSC48M oscillator).

CFD operation is started by writing a '1' to the CFD Enable bit in the External Oscillator Control register (XOCTRL.CFDEN). After starting or restarting the XOSC, the CFD does not detect failure until the start-up time has elapsed. The start-up time is configured by the Oscillator Start-Up Time in the External Multipurpose Crystal Oscillator Control register (XOSCCTRL.STARTUP). Once the XOSC Start-Up Time is elapsed, the XOSC clock is constantly monitored.

During a period of 4 safe clocks (monitor period), the CFD watches for a clock activity from the XOSC. There must be at least one rising and one falling XOSC clock edge during 4 safe clock periods to meet non-failure conditions. If no or insufficient activity is detected, the failure status is asserted: The Clock Failure Detector status bit in the Status register (STATUS.CLKFAIL) and the Clock Failure Detector interrupt flag bit in the Interrupt Flag register (INTFLAG.CLKFAIL) are set. If the CLKFAIL bit in the Interrupt Enable Set register (INTENSET.CLKFAIL) is set, an interrupt is generated as well. If the Event Output enable bit in the Event Control register (EVCTRL.CFDEO) is set, an output event is generated, too.

After a clock failure was issued the monitoring of the XOSC clock is continued, and the Clock Failure Detector status bit in the Status register (STATUS.CLKFAIL) reflects the current XOSC activity.

#### Clock Switch

When a clock failure is detected, the XOSC clock is replaced by the safe clock in order to maintain an active clock during the XOSC clock failure. The safe clock source is the OSC48M oscillator clock. The safe clock source can be scaled down by a configurable prescaler to ensure that the safe clock frequency does not exceed the operating conditions selected by the application. When the XOSC clock is switched to the safe clock, the Clock Switch bit in the Status register (STATUS.CLKSW) is set.

When the CFD has switched to the safe clock, the XOSC is not disabled. If desired, the application must take the necessary actions to disable the oscillator. The application must also take the necessary actions to configure the system clocks to continue normal operations.

In the case the application can recover the XOSC, the application can switch back to the XOSC clock by writing a '1' to Switch Back Enable bit in the Clock Failure Control register (XOSCCTRL.SWBACK). Once the XOSC clock is switched back, the Switch Back bit (XOSCCTRL.SWBACK) is cleared by hardware.

### Prescaler

The CFD has an internal configurable prescaler to generate the safe clock from the OSC48M oscillator. The prescaler size allows to scale down the OSC48M oscillator so the safe clock frequency is not higher than the XOSC clock frequency monitored by the CFD. The division factor is  $2^P$ , with P being the value of the CFD Prescaler bits in the CFD Prescaler Register (CFDPRESC.CFDPRESC).

#### Example

For an external crystal oscillator at 0.4MHz and the OSC48M frequency at 16MHz, the CFDPRESC.CFDPRESC value should be set scale down by more than factor  $16/0.4=80$ , e.g. to 128, for a safe clock of adequate frequency.

### Event

If the Event Output Enable bit in the Event Control register (EVCTRL.CFDEO) is set, the CFD clock failure will be output on the Event Output. When the CFD is switched to the safe clock, the CFD clock failure will not be output on the Event Output.

### Sleep Mode

The CFD is halted depending on configuration of the XOSC and the peripheral clock request. For further details, refer to the Sleep Behavior table above. The CFD interrupt can be used to wake up the device from sleep modes.

#### 21.6.4. 48MHz Internal Oscillator (OSC48M) Operation

The OSC48M is an internal oscillator operating in open-loop mode and generating 48MHz frequency. The OSC48M frequency is selected by writing to the Division Factor field in the OSC48MDIV register (OSC48MDIV.DIV). OSC48M is enabled by writing '1' to the Oscillator Enable bit in the OSC48M Control register (OSC48MCTRL.ENABLE), and disabled by writing a '0' to this bit. Frequency selection must be done when OSC48M is disabled.

After enabling OSC48M, the OSC48M clock is output as soon as the oscillator is ready (STATUS.OSC48MRDY=1). User must ensure that the OSC48M is fully disabled before enabling it by reading STATUS.OSC48MRDY=0.

After reset, OSC16M is enabled and serves as the default clock source at 4MHz.

OSC48M will behave differently in different sleep modes based on the settings of OSC48MCTRL.RUNSTDBY, OSC48MCTRL.ONDEMAND, and OSC48MCTRL.ENABLE. If OSC48MCTRL.ENABLE=0, the OSC48M will be always stopped. For OSC48MCTRL.ENABLE=1, this table is valid:

**Table 21-2. OSC48M Sleep Behavior**

CPU Mode	OSC48MCTRL.RUN STDBY	OSC48MCTRL.OND EMAND	Sleep Behavior
Active or Idle	-	0	Always run
Active or Idle	-	1	Run if requested by peripheral
Standby	1	0	Always run
Standby	1	1	Run if requested by peripheral
Standby	0	-	Run if requested by peripheral

After a hard reset, or when waking up from a sleep mode where the OSC48M was disabled, the OSC48M 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 Delay bit group (OSC48MSTUP.STARTUP) in the OSC48M Startup register. During the start-up time, the oscillator output is masked to ensure that no unstable clock propagates to the digital logic. The OSC48M Ready bit in the Status register (STATUS.OSC48MRDY) is set when the oscillator is stable and ready to be used as a clock source. An interrupt is generated on a zero-to-one transition on STATUS.OSC48MRDY if the OSC48M Ready bit in the Interrupt Enable Set register (INTENSET.OSC48MRDY) is set.

Faster start-up times are achievable by selecting shorter delays. However, the oscillator frequency may not stabilize within tolerances when short delays are used. If a fast start-up time is desired at the expense of initial accuracy, the division factor should be set to two or higher (OSC48MDIV.DIV > 0).

The OSC48M is used as a clock source for the generic clock generators.

**Related Links**

[GCLK - Generic Clock Controller](#) on page 114

**21.6.5. Digital Phase Locked Loop (DPLL) Operation**

The task of the DPLL is to maintain coherence between the input (reference) signal and the respective output frequency, CLK\_DPLL, via phase comparison. The DPLL controller supports three independent sources of reference clocks:

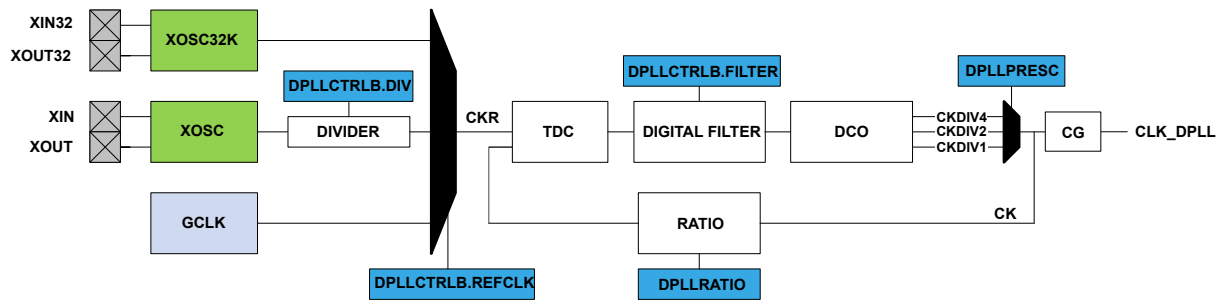
- XOSC32K: this clock is provided by the 32K External Crystal Oscillator (XOSC32K).
- XOSC: this clock is provided by the External Multipurpose Crystal Oscillator (XOSC).
- GCLK: this clock is provided by the Generic Clock Controller.

When the controller is enabled, the relationship between the reference clock frequency and the output clock frequency is:

$$f_{CK} = f_{CKR} \times \left( LDR + 1 + \frac{LDRFRAC}{16} \right) \times \frac{1}{2^{PRESC}}$$

Where  $f_{CK}$  is the frequency of the DPLL output clock, LDR is the loop divider ratio integer part, LDRFRAC is the loop divider ratio fractional part,  $f_{CKR}$  is the frequency of the selected reference clock, and PRESC is the output prescaler value.

**Figure 21-2. DPLL Block Diagram**



When the controller is disabled, the output clock is low. If the Loop Divider Ratio Fractional part bit field in the DPLL Ratio register (DPLLCTRLB.LDRFRAC) is zero, the DPLL works in integer mode. Otherwise, the fractional mode is activated. Note that the fractional part has a negative impact on the jitter of the DPLL.

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.

#### Related Links

[GCLK - Generic Clock Controller](#) on page 114

[OSC32KCTRL – 32KHz Oscillators Controller](#) on page 211

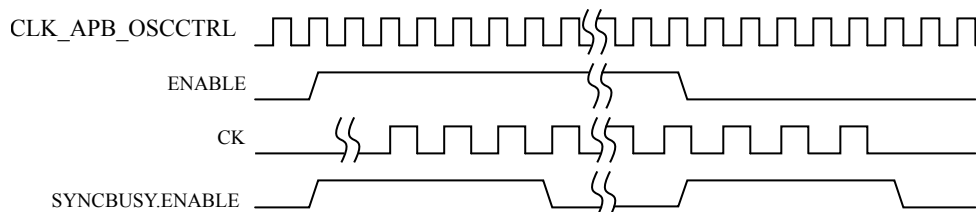
### 21.6.5.1. Basic Operation

#### Initialization, Enabling, Disabling, and Resetting

The DPLL is enabled by writing a '1' to the Enable bit in the DPLL Control A register (DPLLCTRLA.ENABLE). The DPLL is disabled by writing a zero to this bit.

The DPLLSYNCBUSY.ENABLE is set when the DPLLCTRLA.ENABLE bit is modified. It is cleared when the DPLL output clock CK has sampled the bit at the high level after enabling the DPLL. When disabling the DPLL, DPLLSYNCBUSY.ENABLE is cleared when the output clock is no longer running.

#### Figure 21-3. Enable Synchronization Busy Operation



The frequency of the DPLL output clock CK is stable when the module is enabled and when the Lock bit in the DPLL Status register is set (DPLLSTATUS.LOCK).

When the Lock Time bit field in the DPLL Control B register (DPLLCTRLB.LTIME) is non-zero, a user defined lock time is used to validate the lock operation. In this case the lock time is constant. If DPLLCTRLB.LTIME=0, the lock signal is linked with the status bit of the DPLL, and the lock time varies depending on the filter selection and the final target frequency.

When the Wake Up Fast bit (DPLLCTRLB.WUF) is set, the wake up fast mode is activated. In this mode the clock gating cell is enabled at the end of the startup time. At this time the final frequency is not stable, as it is still during the acquisition period, but it allows to save several milliseconds. After first acquisition, the Lock Bypass bit (DPLLCTRLB.LBYPASS) indicates if the lock signal is discarded from the control of the clock gater (CG) generating the output clock CLK\_DPLL.

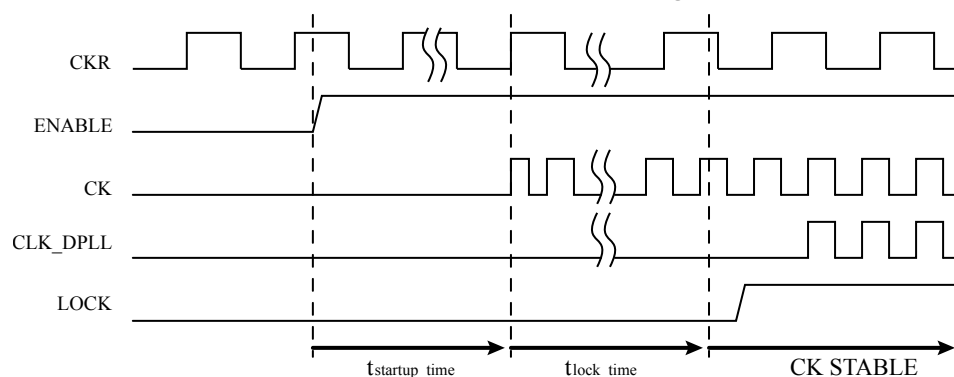
**Table 21-3. CLK\_DPLL Behavior from Startup to First Edge Detection**

WUF	LTIME	CLK_DPLL 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 down-counts to 0.
1	X	Wake Up Fast Mode: First Edge when CK is active (startup time)

**Table 21-4. CLK\_DPLL Behavior after First Edge Detection**

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

**Figure 21-4. CK and CLK\_DPLL Output from DPLL Off Mode to Running Mode**



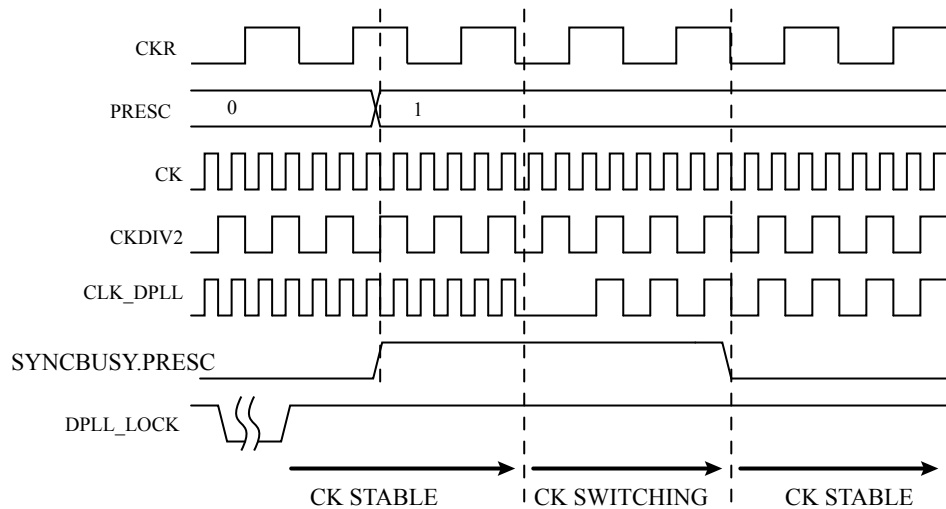
#### Reference Clock Switching

When a software operation requires reference clock switching, the recommended procedure is to turn the DPLL into the standby mode, modify the DPLLCTRLB.REFCLK to select the desired reference source, and activate the DPLL again.

#### Output Clock Prescaler

The DPLL controller includes an output prescaler. This prescaler provides three selectable output clocks CK, CKDIV2 and CKDIV4. The Prescaler bit field in the DPLL Prescaler register (DPLLPRESC.PRESC) is used to select a new output clock prescaler. When the prescaler field is modified, the DPLLSYNCBUSY.DPLLPRESC bit is set. It will be cleared by hardware when the synchronization is over.

**Figure 21-5. Output Clock Switching Operation**

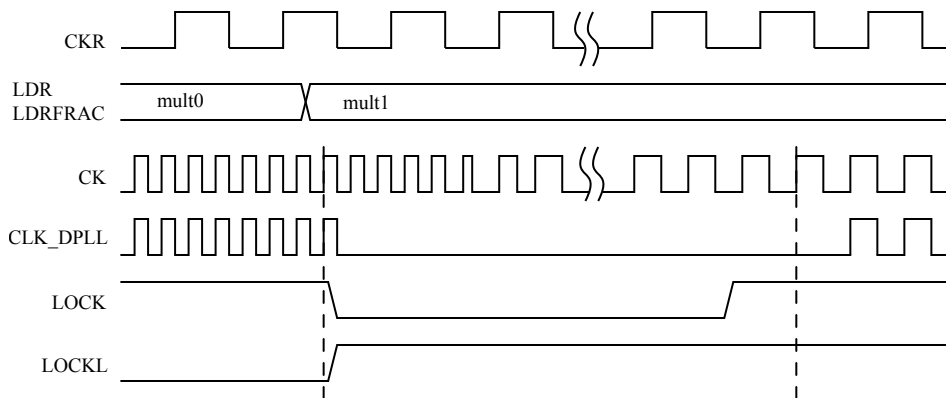


### Loop Divider Ratio Updates

The DPLL Controller supports on-the-fly update of the DPLL Ratio Control (DPLL\_RATIO) register, allowing to modify the loop divider ratio and the loop divider ratio fractional part when the DPLL is enabled.

STATUS.DP\_LLDRT0 is set when the DPLL\_RATIO register has been modified and the DPLL analog cell has successfully sampled the updated value. At that time the DPLLSTATUS.LOCK bit is cleared and set again by hardware when the output frequency reached a stable state.

**Figure 21-6. RATIOCTRL register update operation**



### 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 Filter bit field in the DPLL Control B register (DPLL\_CTRLB.FILTER). The Low Power Enable bit (DPLL\_CTRLB.LPEN) can be used to bypass the Time to Digital Converter (TDC) module.

#### 21.6.6. DMA Operation

Not applicable.

#### 21.6.7. Interrupts

The OSC\_CTRL has the following interrupt sources:

- XOSCRDY - Multipurpose Crystal Oscillator Ready: A 0-to-1 transition on the STATUS.XOSCRDY bit is detected

- CLKFAIL - Clock Failure. A 0-to-1 transition on the STATUS.CLKFAIL bit is detected
- OSC48MRDY - 48MHz Internal Oscillator Ready: A 0-to-1 transition on the STATUS.OSC48MRDY bit is detected
- DPLL-related:
  - DPLLLOCKR - DPLL Lock Rise: A 0-to-1 transition of the STATUS.DPLLLOCKR bit is detected
  - DPLLLOCKF - DPLL Lock Fall: A 0-to-1 transition of the STATUS.DPLLLOCKF bit is detected
  - DPLLLTTO - DPLL Lock Timer Time-out: A 0-to-1 transition of the STATUS.DPLLLTTO bit is detected
  - DPLLLDRTO - DPLL Loop Divider Ratio Update Complete. A 0-to-1 transition of the STATUS.DPLLLDRTO bit is detected

All these interrupts are synchronous wake-up source.

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 the interrupt flag is cleared, the interrupt is disabled, or the OSCCTRL is reset. See the INTFLAG register for details on how to clear interrupt flags.

The OSCCTRL 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. Refer to the INTFLAG register for details.

**Note:** The interrupts must be globally enabled for interrupt requests to be generated.

### 21.6.8. Events

The CFD can generate the following output event:

- Clock Failure (CLKFAIL): Generated when the Clock Failure status bit is set in the Status register (STATUS.CLKFAIL). The CFD event is not generated when the Clock Switch bit (STATUS.CLKSW) in the Status register is set.

Writing a '1' to an Event Output bit in the Event Control register (EVCTRL.CFDEO) enables the CFD output event. Writing a '0' to this bit disables the CFD output event. Refer to the Event System chapter for details on configuring the event system.

### 21.6.9. Synchronization

#### OSC48M

Due to the multiple clock domains, values in the OSC48M control registers need to be synchronized to other clock domains.

When executing an operation that requires synchronization, the relevant synchronization bit in the Synchronization Busy register (OSC48MSYNCBUSY) will be set immediately, and cleared when synchronization is complete.

The following registers need synchronization when written:

- OSC48M Divider register (OSC48MDIV)



## DPLL96M

Due to the multiple clock domains, some registers in the DPLL96M must be synchronized when accessed.

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

The following bits need synchronization when written:

- Enable bit in control register A (DPLLCTRLA.ENABLE)
- DPLL Ratio register (DPLLRATIO)
- DPLL Prescaler register (DPLLPRESC)

### Related Links

[Register Synchronization](#) on page 110

## 21.7. Register Summary

Offset	Name	Bit Pos.								
0x00	INTENCLR	7:0				OSC48MRDY			CLKFAIL	XOSCRDY
0x01		15:8					DPLLLDRTO	DPLLLTO	DPLLLCKF	DPLLLCKR
0x02		23:16								
0x03		31:24								
0x04	INTENSET	7:0				OSC48MRDY			CLKFAIL	XOSCRDY
0x05		15:8					DPLLLDRTO	DPLLLTO	DPLLLCKF	DPLLLCKR
0x06		23:16								
0x07		31:24								
0x08	INTFLAG	7:0				OSC48MRDY			CLKFAIL	XOSCRDY
0x09		15:8					DPLLLDRTO	DPLLLTO	DPLLLCKF	DPLLLCKR
0x0A		23:16								
0x0B		31:24								
0x0C	STATUS	7:0				OSC48MRDY		CLKSW	CLKFAIL	XOSCRDY
0x0D		15:8					DPLLLDRTO	DPLLLTO	DPLLLCKF	DPLLLCKR
0x0E		23:16								
0x0F		31:24								
0x10	XOSCCTRL	7:0	ONDEMAND	RUNSTDBY		SWBACK	CFDEN	XTALEN	ENABLE	
0x11		15:8	STARTUP[3:0]				AMPGC	GAIN[2:0]		
0x12	CFDPRESC	7:0					CFDPRESC[2:0]			
0x13		15:8								
0x13	EVCTRL	7:0								CFDEO
0x14		15:8								
0x14	OSC48MCTRL	7:0	ONDEMAND	RUNSTDBY					ENABLE	
0x15		15:8								
0x16		23:16								
0x17		31:24								
0x18	OSC48MSYNCBUS Y	7:0					OSC48MDIV			
0x19		15:8								
0x1A		23:16								
0x1B		31:24								
0x1E ... 0x1F	Reserved									
0x20	DPLLATIO	7:0	LDR[7:0]							
0x21		15:8	LDR[11:8]							
0x22		23:16	LDRFRAC[3:0]							
0x23		31:24								
0x24	DPLLCTRLB	7:0			REFCLK[1:0]	WUF	LPEN	FILTER[1:0]		
0x25		15:8			LBYPASS		LTIME[2:0]			
0x26		23:16	DIV[7:0]							
0x27		31:24	DIV[10:8]							
0x28	DPLLPRESC	7:0	PRESC[1:0]							

Offset	Name	Bit Pos.								
0x29 ...	Reserved									
0x2B										
0x2C	<a href="#">DPLLSYNCBUSY</a>	7:0					DPLLPRESC	DPLLRATIO	ENABLE	
0x2D ...	Reserved									
0x2F										
0x30	<a href="#">DPLLSTATUS</a>	7:0							CLKRDY	LOCK

## 21.8. Register Description

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

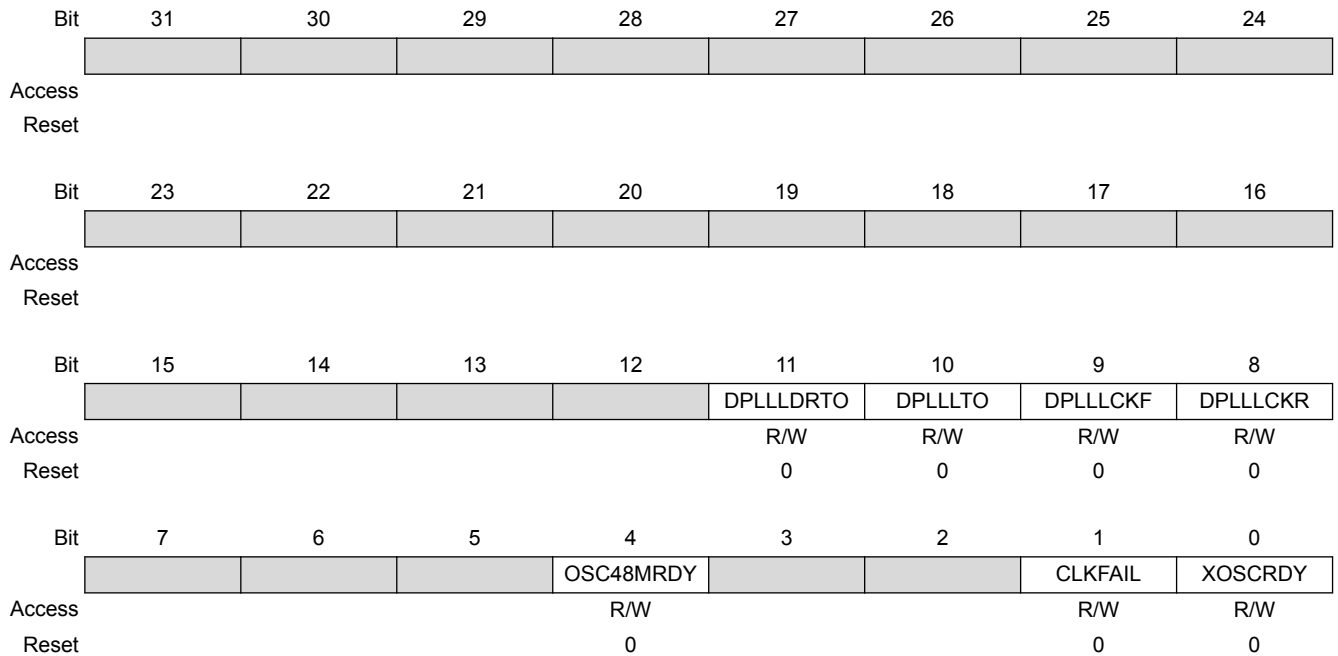
Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Write-protection is denoted by the "PAC Write-Protection" property in each individual register description. Refer to the [Register Access Protection](#) on page 172 section and the [PAC - Peripheral Access Controller](#) on page 46 chapter for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the "Read-Synchronized" or "Write.Synchronized" property in each individual register description. Refer to the [Synchronization](#) on page 180 section for details.

### 21.8.1. Interrupt Enable Clear

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

**Name:** INTENCLR  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



#### Bit 11 – DPLLLDRTO: DPLL Loop Divider Ratio Update Complete Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the DPLL Loop Divider Ratio Update Complete Interrupt Enable bit, which disables the DPLL Loop Divider Ratio Update Complete interrupt.

Value	Description
0	The DPLL Loop Divider Ratio Update Complete interrupt is disabled.
1	The DPLL Loop Divider Ratio Update Complete interrupt is enabled, and an interrupt request will be generated when the DPLL Loop Divider Ratio Update Complete Interrupt flag is set.

#### Bit 10 – DPLLLTO: DPLL Lock Timeout Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' 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 9 – DPLLLCKF: DPLL Lock Fall Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' 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 8 – DPLLLCKR: DPLL Lock Rise Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' 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 4 – OSC48MRDY: OSC48M Ready Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the OSC48M Ready Interrupt Enable bit, which disables the OSC48M Ready interrupt.

Value	Description
0	The OSC48M Ready interrupt is disabled.
1	The OSC48M Ready interrupt is enabled, and an interrupt request will be generated when the OSC48M Ready Interrupt flag is set.

#### Bit 1 – CLKFAIL: Clock Failure Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the XOSC Clock Failure Interrupt Enable bit, which disables the XOSC Clock Failure interrupt.

Value	Description
0	The XOSC Clock Failure interrupt is disabled.
1	The XOSC Clock Failure interrupt is enabled, and an interrupt request will be generated when the XOSC Clock Failure Interrupt flag is set.

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

Writing '0' to this bit has no effect.

Writing '1' 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.

## 21.8.2. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x04

**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
Access								
Reset								
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
Reset				0			0	0

### Bit 11 – DPLLLDRTO: DPLL Loop Divider Ratio Update Complete Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the DPLL Loop Ratio Update Complete Interrupt Enable bit, which enables the DPLL Loop Ratio Update Complete interrupt.

Value	Description
0	The DPLL Loop Divider Ratio Update Complete interrupt is disabled.
1	The DPLL Loop Ratio Update Complete interrupt is enabled, and an interrupt request will be generated when the DPLL Loop Ratio Update Complete Interrupt flag is set.

### Bit 10 – DPLLLTO: DPLL Lock Timeout Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' 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 9 – DPLLLCKF: DPLL Lock Fall Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' 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 8 – DPLLLCKR: DPLL Lock Rise Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' 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 4 – OSC48MRDY: OSC48M Ready Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the OSC48M Ready Interrupt Enable bit, which enables the OSC48M Ready interrupt.

Value	Description
0	The OSC48M Ready interrupt is disabled.
1	The OSC48M Ready interrupt is enabled, and an interrupt request will be generated when the OSC48M Ready Interrupt flag is set.

#### Bit 1 – CLKFAIL: XOSC Clock Failure Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the XOSC Clock Failure Interrupt Enable bit, which enables the XOSC Clock Failure Interrupt.

Value	Description
0	The XOSC Clock Failure Interrupt is disabled.
1	The XOSC Clock Failure Interrupt is enabled, and an interrupt request will be generated when the XOSC Clock Failure Interrupt flag is set.



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

Writing '0' to this bit has no effect.

Writing '1' 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.

### 21.8.3. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**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
Access								
Reset								
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
Reset				0			0	0

#### Bit 11 – DPLLLDRTO: DPLL Loop Divider Ratio Update Complete

This flag is cleared by writing '1' to it.

This flag is set on 0-to-1 transition of the DPLL Loop Divider Ratio Update Complete bit in the Status register (STATUS.DPLLLDRTO) and will generate an interrupt request if INTENSET.DPLLLDRTO is '1'.

Writing '0' to this bit has no effect.

Writing '1' to this bit clears the DPLL Loop Divider Ratio Update Complete interrupt flag.

#### Bit 10 – DPLLLTO: DPLL Lock Timeout

This flag is cleared by writing '1' to it.

This flag is set on 0-to-1 transition of the DPLL Lock Timeout bit in the Status register (STATUS.DPLLLTO) and will generate an interrupt request if INTENSET.DPLLLTO is '1'.

Writing '0' to this bit has no effect.

Writing '1' to this bit clears the DPLL Lock Timeout interrupt flag.

#### Bit 9 – DPLLLCKF: DPLL Lock Fall

This flag is cleared by writing '1' to it.

This flag is set on 0-to-1 transition of the DPLL Lock Fall bit in the Status register (STATUS.DPLLLCKF) and will generate an interrupt request if INTENSET.DPLLLCKF is '1'.

Writing '0' to this bit has no effect.

Writing '1' to this bit clears the DPLL Lock Fall interrupt flag.

**Bit 8 – DPLLLCKR: DPLL Lock Rise**

This flag is cleared by writing '1' to it.

This flag is set on 0-to-1 transition of the DPLL Lock Rise bit in the Status register (STATUS.DPLLLCKR) and will generate an interrupt request if INTENSET.DPLLLCKR is '1'.

Writing '0' to this bit has no effect.

Writing '1' to this bit clears the DPLL Lock Rise interrupt flag.

**Bit 4 – OSC48MRDY: OSC48M Ready**

This flag is cleared by writing '1' to it.

This flag is set on 0-to-1 transition of the OSC48M Ready bit in the Status register (STATUS.OSC48MRDY) and will generate an interrupt request if INTENSET.OSC48MRDY is '1'.

Writing '0' to this bit has no effect.

Writing '1' to this bit clears the OSC48M Ready interrupt flag.

**Bit 1 – CLKFAIL: XOSC Failure Detection**

This flag is cleared by writing '1' to it.

This flag is set on a 0-to-1 transition of the XOSC Clock Failure bit in the Status register (STATUS.CLKFAIL) and will generate an interrupt request if INTENSET.CLKFAIL is '1'.

Writing '0' to this bit has no effect.

Writing '1' to this bit clears the XOSC Clock Fail interrupt flag.

**Bit 0 – XOSCRDY: XOSC Ready**

This flag is cleared by writing '1' to it.

This flag is set on a 0-to-1 transition of the XOSC Ready bit in the Status register (STATUS.XOSCRDY) and will generate an interrupt request if INTENSET.XOSCRDY is '1'.

Writing '0' to this bit has no effect.

Writing '1' to this bit clears the XOSC Ready interrupt flag.

## 21.8.4. Status

**Name:** STATUS  
**Offset:** 0x0C  
**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					DPLLLDRTO	DPLLLTO	DPLLLCKF	DPLLLCKR
Reset					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access				OSC48MRDY		CLKSW	CLKFAIL	XOSCRDY
Reset				R		R	R	R
Reset				0		0	0	0

### Bit 11 – DPLLLDRTO: DPLL Loop Divider Ratio Update Complete

Value	Description
0	DPLL Loop Divider Ratio Update Complete not detected.
1	DPLL Loop Divider Ratio Update Complete detected.

### Bit 10 – DPLLLTO: DPLL Lock Timeout

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

### Bit 9 – DPLLLCKF: DPLL Lock Fall

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

### Bit 8 – DPLLLCKR: DPLL Lock Rise

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

**Bit 4 – OSC48MRDY: OSC48M Ready**

Value	Description
0	OSC48M is not ready.
1	OSC48M is stable and ready to be used as a clock source.

**Bit 2 – CLKSW: XOSC Clock Switch**

Value	Description
0	XOSC is not switched and provides the external clock or crystal oscillator clock.
1	XOSC is switched and provides the safe clock.

**Bit 1 – CLKFAIL: XOSC Clock Failure**

Value	Description
0	No XOSC failure detected.
1	A XOSC failure was detected.

**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.

## 21.8.5. External Multipurpose Crystal Oscillator (XOSC) Control

**Name:** XOSCCTRL  
**Offset:** 0x10  
**Reset:** 0x0080  
**Property:** PAC Write-Protection

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		SWBACK	CFDEN	XTALEN	ENABLE	
Access	R/W	R/W		R/W	R/W	R/W	R/W	
Reset	1	0		0	0	0	0	

### Bits 15:12 – STARTUP[3:0]: Start-Up Time

These bits select start-up time for the oscillator.

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

**Table 21-5. Start-Up Time for External Multipurpose Crystal Oscillator**

STARTUP[3:0]	Number of OSCULP32K Clock Cycles	Number of XOSC Clock Cycles	Approximate Equivalent Time [μs]
0x0	1	3	31
0x1	2	3	61
0x2	4	3	122
0x3	8	3	244
0x4	16	3	488
0x5	32	3	977
0x6	64	3	1953
0x7	128	3	3906
0x8	256	3	7813
0x9	512	3	15625
0xA	1024	3	31250
0xB	2048	3	62500μs
0xC	4096	3	125000
0xD	8192	3	250000
0xE	16384	3	500000
0xF	32768	3	1000000

**Note:**

1. Actual startup time is 1 OSCULP32K cycle + 3 XOSC cycles.
2. The given time neglects the three XOSC cycles before OSCULP32K cycle.

**Bit 11 – AMPGC: Automatic Amplitude Gain Control**

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. Those bits must be properly configured even when the Automatic Amplitude Gain Control is active.

Value	Recommended Max Frequency [MHz]
0x0	2
0x1	4
0x2	8
0x3	16
0x4	30
0x5-0x7	Reserved

**Bit 7 – ONDEMAND: On Demand Control**

The On Demand operation mode allows the oscillator to be enabled or disabled, depending on peripheral clock requests.

If the ONDEMAND bit has been previously written to '1', 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.

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 XOSC is disabled in standby sleep mode if no peripheral requests the clock.
1	The XOSC is not stopped in standby sleep mode. If ONDEMAND=1, the XOSC will be running when a peripheral is requesting the clock. If ONDEMAND=0, the clock source will always be running in standby sleep mode.

#### Bit 4 – SWBACK: Clock Switch Back

This bit controls the XOSC output switch back to the external clock or crystal oscillator in case of clock recovery:

Value	Description
0	The clock switch back is disabled.
1	The clock switch back is enabled. This bit is reset once the XOSC putput clock is switched back to the external clock or crystal oscillator.

#### Bit 3 – CFDEN: Clock Failure Detector Enable

This bit controls the clock failure detector:

Value	Description
0	The Clock Failure Detector is disabled.
1	the Clock Failure Detector 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 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.



## 21.8.6. Clock Failure Detector Prescaler

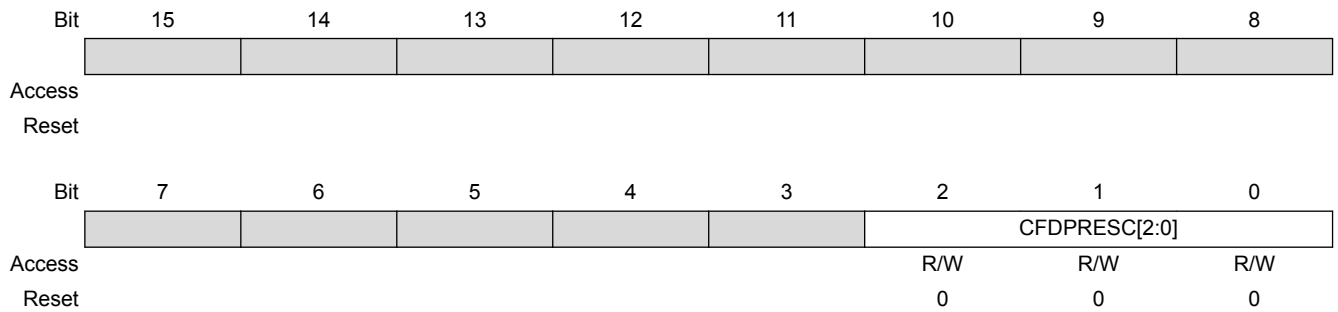
[registerDescription]

**Name:** CFDPRESC

**Offset:** 0x12

**Reset:** 0x00

**Property:** PAC Write-Protection



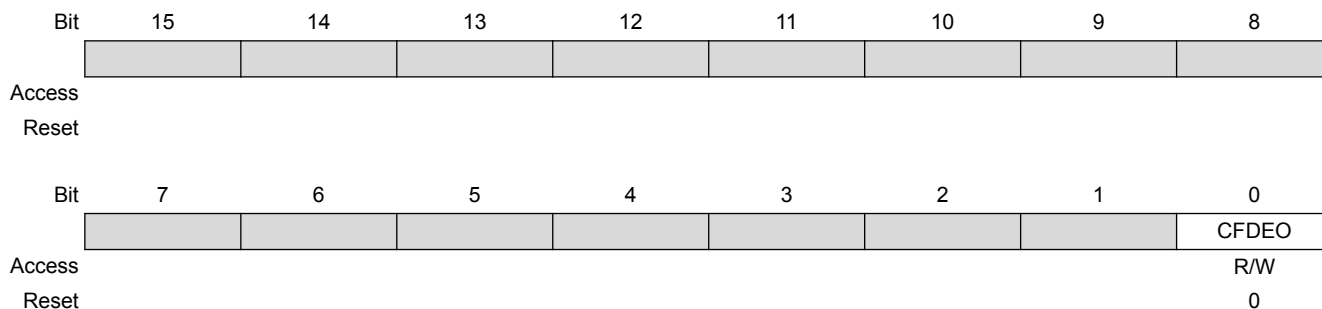
### Bits 2:0 – CFDPRESC[2:0]: Clock Failure Detector Prescaler

These bits select the prescaler for the clock failure detector.

The OSC48M oscillator is used to clock the CFD prescaler. The CFD safe clock frequency is the OSC48M frequency divided by  $2^{\text{CFDPRESC}}$ .

## 21.8.7. Event Control

**Name:** EVCTRL  
**Offset:** 0x13  
**Reset:** 0x00  
**Property:** PAC Write-Protection



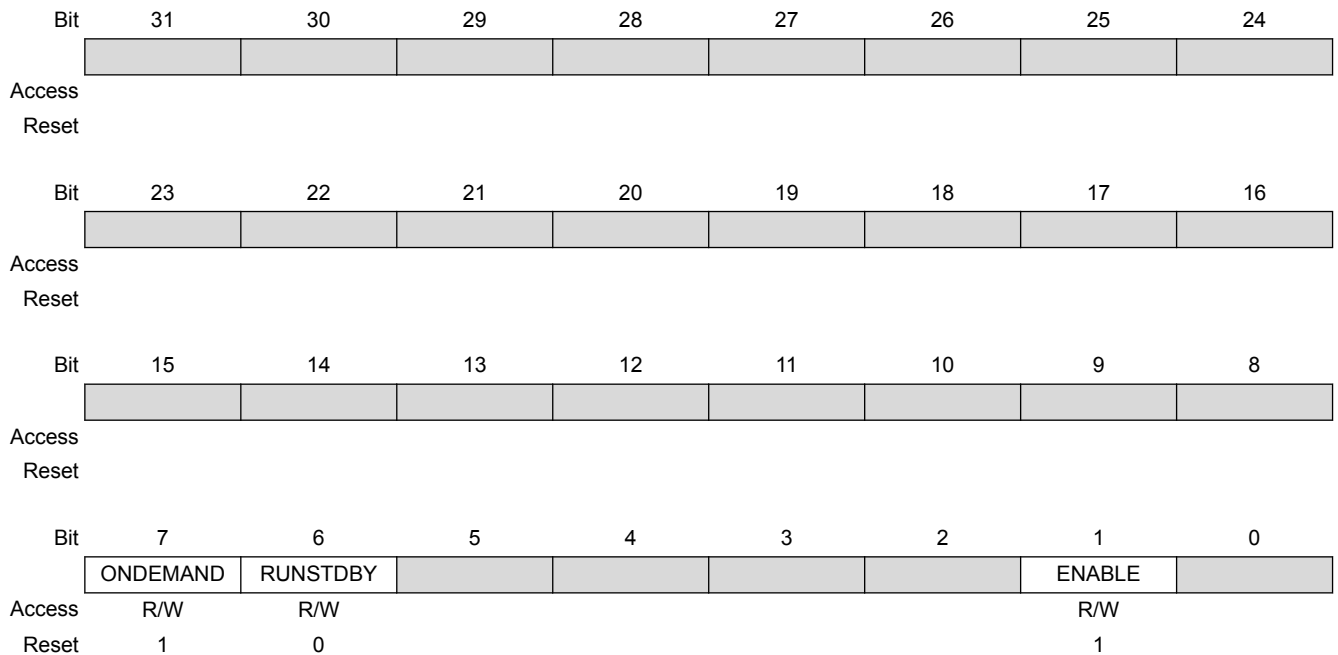
### Bit 0 – CFDEO: Clock Failure Detector Event Out

This bit indicates whether the Clock Failure detector event output is enabled or not and an output event will be generated when the Clock Failure detector detects a clock failure

Value	Description
0	Clock Failure detector event output is disabled and no event will be generated.
1	Clock Failure detector event output is enabled and an event will be generated.

## 21.8.8. 48MHz Internal Oscillator (OSC48M) Control

**Name:** OSC48MCTRL  
**Offset:** 0x14  
**Reset:** 0x82  
**Property:** PAC Write-Protection



### Bit 7 – ONDEMAND: On Demand Control

The On Demand operation mode allows the oscillator to be enabled or disabled depending on peripheral clock requests.

If the ONDEMAND bit has been previously written to '1', 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.

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 OSC48M behaves during standby sleep mode.

Value	Description
0	The OSC48M is disabled in standby sleep mode if no peripheral requests the clock.
1	The OSC48M is not stopped in standby sleep mode. If ONDEMAND=1, the OSC48M will be running when a peripheral is requesting the clock. If ONDEMAND=0, the clock source will always be running in standby sleep mode.

**Bit 1 – ENABLE: Oscillator Enable**

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

## 21.8.9. OSC48M Divider

**Name:** OSC48MDIV

**Offset:** 0x15

**Reset:** 0x0B

**Property:** -

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

### Bits 3:0 – DIV[3:0]: Oscillator Divider Selection

These bits control the oscillator frequency range by adjusting the division ratio. The oscillator frequency is 48MHz divided by DIV+1.

Value	Description
0000	48MHz
0001	24MHz
0010	16MHz
0011	12MHz
0100	9.6MHz
0101	8MHz
0110	6.86MHz
0111	6MHz
1000	5.33MHz
1001	4.8MHz
1010	4.36MHz
1011	4MHz
1100	3.69MHz
1101	3.43MHz
1110	3.2MHz
1111	3MHz

## 21.8.10. OSC48M Startup

**Name:** OSC48MSTUP

**Offset:** 0x16

**Reset:** 0x07

**Property:** -

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

### Bits 2:0 – STARTUP[2:0]: Oscillator Startup Delay

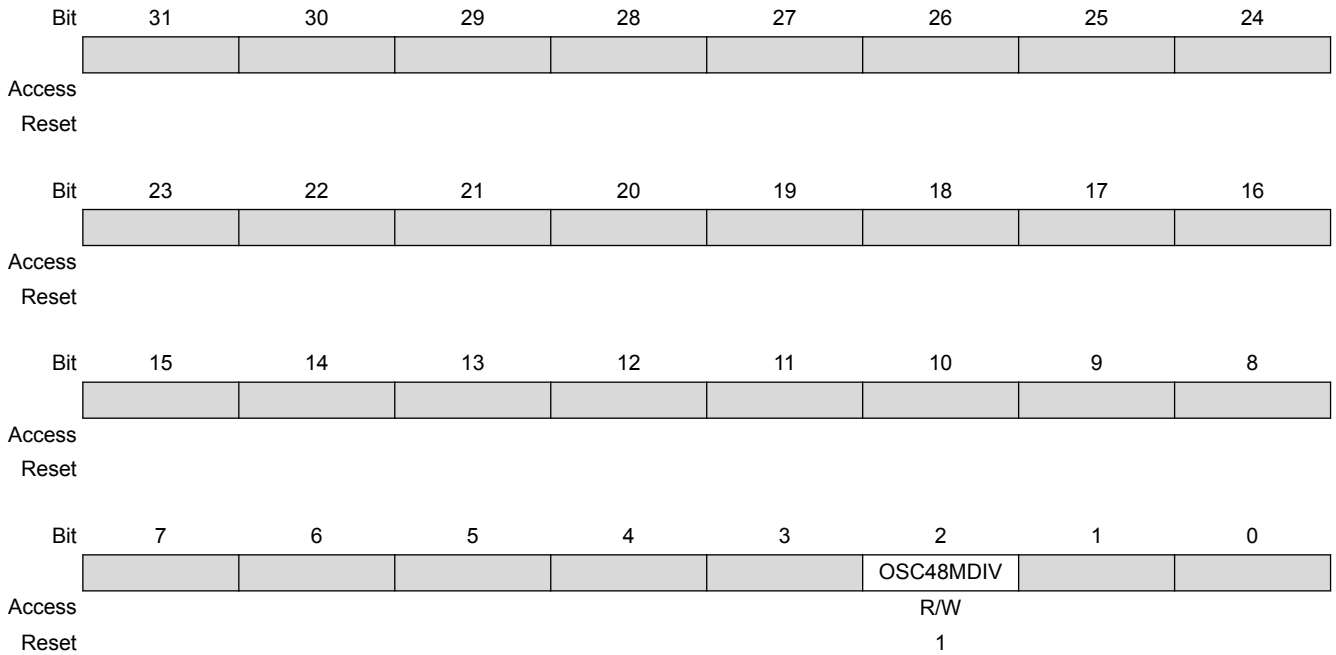
These bits select the oscillator start-up delay in oscillator cycles.

**Table 21-6. Oscillator Divider Selection**

STARTUP[2:0]	Number of OSCM48M Clock Cycles	Approximate Equivalent Time
0x0	8	166ns
0x1	16	333ns
0x2	32	667ns
0x3	64	1.333μs
0x4	128	2.667μs
0x5	256	5.333μs
0x6	512	10.667μs
0x7	1024	21.333μs

### 21.8.11. OSC48M Synchronization Busy

**Name:** OSC48MSYNCBUSY  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** -



#### Bit 2 – OSC48MDIV: Oscillator Divider Synchronization Status

This bit is set when OSC48MDIV register is written.

This bit is cleared when OSC48MDIV synchronization is completed.

Value	Description
0	No synchronized access.
1	Synchronized access is ongoing.

## 21.8.12. DPLL Control A

**Name:** DPLLCTRLA

**Offset:** 0x1C

**Reset:** 0x80

**Property:** PAC Write-Protection, Write-Synchronized (ENABLE)

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

### Bit 7 – ONDEMAND: On Demand Clock Activation

The On Demand operation mode allows the DPLL to be enabled or disabled depending on peripheral clock requests.

If the ONDEMAND bit has been previously written to '1', the DPLL will only be running when requested by a peripheral. If there is no peripheral requesting the DPLL's clock source, the DPLL will be in a disabled state.

If On Demand is disabled the DPLL will always be running when enabled.

In standby sleep mode, the On Demand operation is still active.

Value	Description
0	The DPLL is always on, if enabled.
1	The DPLL is enabled when a peripheral is requesting the DPLL to be used as a clock source. The DPLL is disabled if no peripheral is requesting the clock source.

### Bit 6 – RUNSTDBY: Run in Standby

This bit controls how the DPLL behaves during standby sleep mode:

Value	Description
0	The DPLL is disabled in standby sleep mode if no peripheral requests the clock.
1	The DPLL is not stopped in standby sleep mode. If ONDEMAND=1, the DPLL will be running when a peripheral is requesting the clock. If ONDEMAND=0, the clock source will always be running in standby sleep mode.

### Bit 1 – ENABLE: DPLL Enable

The software operation of enabling or disabling the DPLL takes a few clock cycles, so the DPLLSYNCBUSY.ENABLE status bit indicates when the DPLL is successfully enabled or disabled.

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



### 21.8.13. DPLL Ratio Control

**Name:** DPLL\_RATIO  
**Offset:** 0x20  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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

#### Bits 19:16 – LDRFRAC[3:0]: Loop Divider Ratio Fractional Part

Writing these bits selects the fractional part of the frequency multiplier. Due to synchronization there is a delay between writing these bits and the effect on the DPLL output clock. The value written will read back immediately and the DPLL\_RATIO bit in the DPLL Synchronization Busy register (DPLLSYNCBUSY.DPLL\_RATIO) will be set. DPLLSYNCBUSY.DPLL\_RATIO will be cleared when the operation is completed.

#### Bits 11:0 – LDR[11:0]: Loop Divider Ratio

Writing these bits selects the integer part of the frequency multiplier. The value written to these bits will read back immediately, and the DPLL\_RATIO bit in the DPLL Synchronization busy register (DPLLSYNCBUSY.DPLL\_RATIO), will be set. DPLLSYNCBUSY.DPLL\_RATIO will be cleared when the operation is completed.

## 21.8.14. DPLL Control B

**Name:** DPLLCTRLB  
**Offset:** 0x24  
**Reset:** 0x00  
**Property:** Enable-Protected, PAC Write-Protection

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
Reset			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	
Reset			0		0	0	0	

### Bits 26:16 – DIV[10:0]: Clock Divider

These bits set the XOSC clock division factor and can be calculated with following formula:

$$f_{DIV} = \frac{f_{XOSC}}{2x(DIV + 1)}$$

### Bit 12 – LBYPASS: Lock Bypass

Value	Description
0	DPLL Lock signal drives the DPLL controller internal logic.
1	DPLL Lock signal is always asserted.

### Bits 10:8 – LTIME[2:0]: Lock Time

These bits select the lock time-out value:

Value	Name	Description
0x0	Default	No time-out. Automatic lock.
0x1	Reserved	
0x2	Reserved	
0x3	Reserved	

Value	Name	Description
0x4	8MS	Time-out if no lock within 8ms
0x5	9MS	Time-out if no lock within 9ms
0x6	10MS	Time-out if no lock within 10ms
0x7	11MS	Time-out if no lock within 11ms

#### Bits 5:4 – REFCLK[1:0]: Reference Clock Selection

Write these bits to select the DPLL clock reference:

Value	Name	Description
0x0	XOSC32K	XOSC32K clock reference
0x1	XOSC	XOSC clock reference
0x2	GCLK	GCLK clock reference
0x3	Reserved	

#### Bit 3 – WUF: Wake Up Fast

Value	Description
0	DPLL clock is output after startup and lock time.
1	DPLL clock is output after startup time.

#### Bit 2 – LPEN: Low-Power Enable

Value	Description
0	The low-power mode is disabled. Time to Digital Converter is enabled.
1	The low-power mode is enabled. Time to Digital Converter is disabled. 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:

Value	Name	Description
0x0	DEFAULT	Default filter mode
0x1	LBFILT	Low bandwidth filter
0x2	HBFILT	High bandwidth filter
0x3	HDFILT	High damping filter

## 21.8.15. DPLL Prescaler

**Name:** DPLLPRESC  
**Offset:** 0x28  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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

### Bits 1:0 – PRESC[1:0]: Output Clock Prescaler

These bits define the output clock prescaler setting.

Value	Name	Description
0x0	DIV1	DPLL output is divided by 1
0x1	DIV2	DPLL output is divided by 2
0x2	DIV4	DPLL output is divided by 4
0x3	Reserved	

## 21.8.16. DPLL Synchronization Busy

**Name:** DPLLSYNCBUSY

**Offset:** 0x2C

**Reset:** 0x00

**Property:** –

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

### Bit 3 – DPLLPRESC: DPLL Prescaler Synchronization Status

Value	Description
0	The DPLLPRESC register has been synchronized.
1	The DPLLPRESC register value has changed and its synchronization is in progress.

### Bit 2 – DPLLRRATIO: DPLL Loop Divider Ratio Synchronization Status

Value	Description
0	The DPLLRRATIO register has been synchronized.
1	The DPLLRRATIO register value has changed and its synchronization is in progress.

### Bit 1 – ENABLE: DPLL Enable Synchronization Status

Value	Description
0	The DPLLCTRLA.ENABLE bit has been synchronized.
1	The DPLLCTRLA.ENABLE bit value has changed and its synchronization is in progress.

## 21.8.17. DPLL Status

**Name:** DPLLSTATUS

**Offset:** 0x30

**Reset:** 0x00

**Property:** –

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

### Bit 1 – CLKRDY: Output Clock Ready

Value	Description
0	The DPLL output clock is off.
1	The DPLL output clock is on.

### Bit 0 – LOCK: DPLL Lock status bit

Value	Description
0	The DPLL Lock signal is cleared, when the DPLL is disabled or when the DPLL is trying to reach the target frequency.
1	The DPLL Lock signal is asserted when the desired frequency is reached.

## 22. OSC32KCTRL – 32KHz Oscillators Controller

### 22.1. Overview

The 32KHz Oscillators Controller (OSC32KCTRL) provides a user interface to the 32.768kHz oscillators: XOSC32K, OSC32K, and OSCULP32K.

The OSC32KCTRL sub-peripherals can be enabled, disabled, calibrated, and monitored through interface registers.

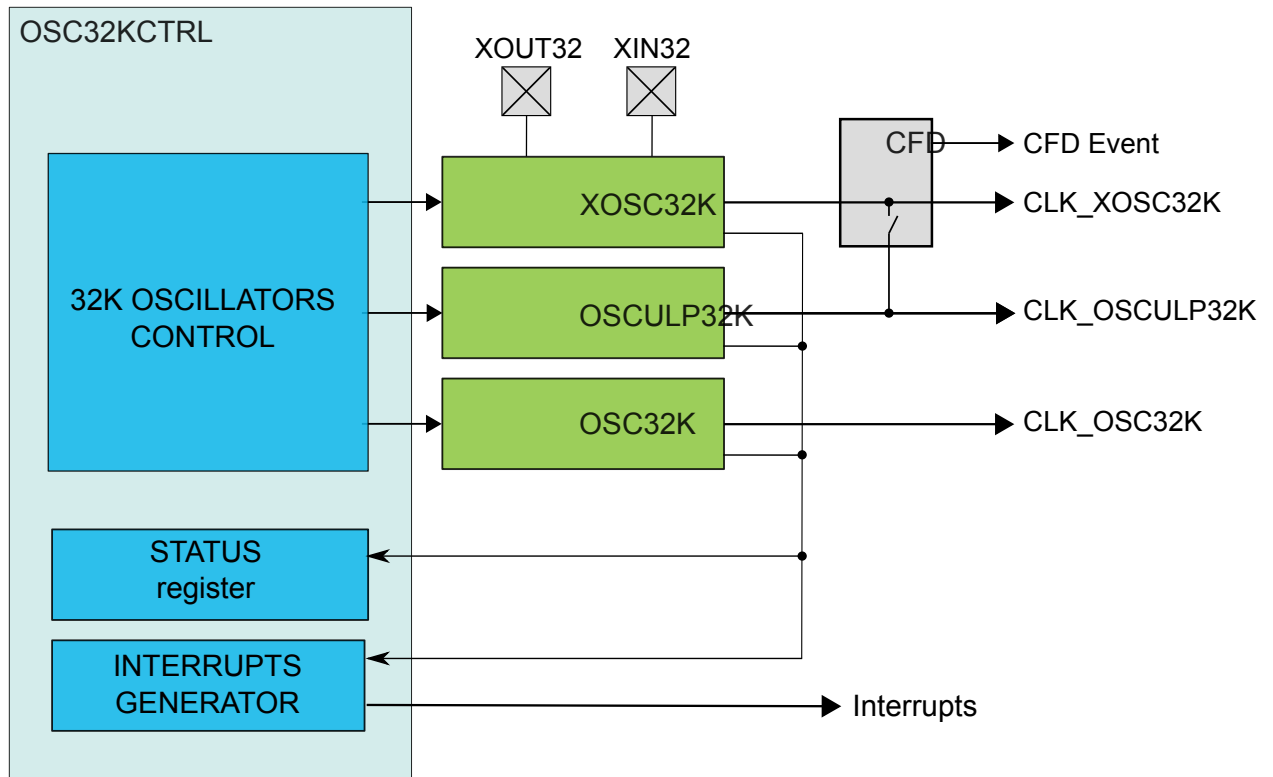
All sub-peripheral statuses are collected in the Status register (STATUS). They can additionally trigger interrupts upon status changes via the INTENSET, INTENCLR, and INTFLAG registers.

### 22.2. Features

- 32.768kHz Crystal Oscillator (XOSC32K)
  - Programmable start-up time
  - Crystal or external input clock on XIN32 I/O
  - Clock failure detection with safe clock switch
  - Clock failure event output
- 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
- 1.024kHz clock outputs available

## 22.3. Block Diagram

Figure 22-1. OSC32KCTRL Block Diagram



## 22.4. Signal Description

Signal	Description	Type
XIN32	Analog Input	32.768kHz Crystal Oscillator or external clock generator input
XOUT32	Analog Output	32.768kHz Crystal Oscillator output

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

## 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

I/O lines are configured by OSC32KCTRL when XOSC32K is enabled, and need no user configuration.

### 22.5.2. Power Management

The OSC32KCTRL will continue to operate in any sleep mode where a 32KHz oscillator is running as source clock. The OSC32KCTRL interrupts can be used to wake up the device from sleep modes.

#### Related Links

[PM – Power Manager](#) on page 162



### 22.5.3. Clocks

The OSC32KCTRL gathers controls for all 32KHz oscillators and provides clock sources to the Generic Clock Controller (GCLK), Real-Time Counter (RTC), and Watchdog Timer (WDT).

The available clock sources are: XOSC32K, OSC32K, and OSCULP32K.

#### Related Links

[Peripheral Clock Masking](#) on page 135

### 22.5.4. Interrupts

The interrupt request lines are connected to the interrupt controller. Using the OSC32KCTRL interrupts requires the interrupt controller to be configured first.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

### 22.5.5. Events

The events of this peripheral are connected to the Event System.

#### Related Links

[EVSYS – Event System](#) on page 474

### 22.5.6. Debug Operation

When the CPU is halted in debug mode, OSC32KCTRL will continue normal operation. If OSC32KCTRL is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

### 22.5.7. 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

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](#) on page 46

### 22.5.8. Analog Connections

The external 32.768kHz crystal must be connected between the XIN32 and XOUT32 pins, along with any required load capacitors. For details on recommended oscillator characteristics and capacitor load, refer to the related links.

#### Related Links

[Electrical Characteristics](#) on page 1105

### 22.5.9. Calibration

The OSC32K calibration value from the production test must be loaded from the NVM Software Calibration Area into the OSC32K register (OSC32K.CALIB) by software to achieve specified accuracy.

#### Related Links

[NVM Software Calibration Area Mapping](#) on page 37

## 22.6. Functional Description

### 22.6.1. Principle of Operation

XOSC32K, OSC32K, and OSCULP32K are configured via OSC32KCTRL control registers. Through this interface, the sub-peripherals are enabled, disabled, or have their calibration values updated.

The STATUS register gathers different status signals coming from the sub-peripherals of OSC32KCTRL. 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.

### 22.6.2. 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

At reset, the XOSC32K is disabled, and the XIN32/XOUT32 pins can either 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, the XIN32 and XOUT32 pins are controlled by the OSC32KCTRL, and GPIO functions are overridden on both pins. When in external clock mode, the only XIN32 pin will be overridden and controlled by the OSC32KCTRL, while the XOUT32 pin can still be used as a GPIO pin.

The XOSC32K is enabled by writing a '1' to the Enable bit in the 32KHz External Crystal Oscillator Control register (XOSC32K.ENABLE=1). The XOSC32K is disabled by writing a '0' to the Enable bit in the 32KHz External Crystal Oscillator Control register (XOSC32K.ENABLE=0).

To enable the XOSC32K as a crystal oscillator, the XTALEN bit in the 32KHz External Crystal Oscillator Control register must be set (XOSC32K.XTALEN=1). If XOSC32K.XTALEN is '0', the external clock input will be enabled.

The XOSC32K 32.768kHz output is enabled by setting the 32KHz Output Enable bit in the 32KHz External Crystal Oscillator Control register (XOSC32K.EN32K=1). The XOSC32K also has a 1.024kHz clock output. This is enabled by setting the 1KHz Output Enable bit in the 32KHz External Crystal Oscillator Control register (XOSC32K.EN1K=1).

It is also possible to lock the XOSC32K configuration by setting the Write Lock bit in the 32KHz External Crystal Oscillator Control register (XOSC32K.WRTLOCK=1). If set, the XOSC32K configuration is locked until a Power-On Reset (POR) is detected.

The XOSC32K will behave differently in different sleep modes based on the settings of XOSC32K.RUNSTDBY, XOSC32K.ONDEMAND, and XOSC32K.ENABLE:

**Table 22-1. XOSC32K Sleep Behavior**

XOSC32K.RUNSTDBY	XOSC32K.ONDEMAND	XOSC32K.ENABLE	Sleep Behavior
-	-	0	Disabled
0	0	1	Always run in IDLE sleep modes.  In STANDBY sleep mode: XOSC32K and CFD are only running if requested by peripheral.
1	0	1	Always run in both IDLE and STANDBY sleep modes.
-	1	1	In IDLE or STANDBY sleep modes: XOSC32K and CFD are only running if requested by a peripheral.

As a crystal oscillator usually requires a very long start-up time, the 32KHz External Crystal Oscillator will keep running across resets when XOSC32K.ONDEMAND=0, except for power-on reset (POR). After a reset or when waking up from a sleep mode where the XOSC32K was disabled, the XOSC32K 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 (XOSC32K.STARTUP) in the 32KHz External 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.

Once the external clock or crystal oscillator is stable and ready to be used as a clock source, the XOSC32K Ready bit in the Status register is set (STATUS.XOSC32KRDY=1). The transition of STATUS.XOSC32KRDY from '0' to '1' generates an interrupt if the XOSC32K Ready bit in the Interrupt Enable Set register is set (INTENSET.XOSC32KRDY=1).

The XOSC32K can be used as a source for Generic Clock Generators (GCLK) or for the Real-Time Counter (RTC). Before enabling the GCLK or the RTC module, the corresponding oscillator output must be enabled (XOSC32K.EN32K or XOSC32K.EN1K) in order to ensure proper operation. In the same way, the GCLK or RTC modules must be disabled before the clock selection is changed. For details on RTC clock configuration, refer also to [Real-Time Counter Clock Selection](#) on page 219.

#### Related Links

[GCLK - Generic Clock Controller](#) on page 114

[RTC – Real-Time Counter](#) on page 283

### 22.6.3. Clock Failure Detection Operation

The Clock Failure Detector (CFD) allows the user to monitor the external clock or crystal oscillator signal provided by the external oscillator (XOSC32K). The CFD detects failing operation of the XOSC32K clock with reduced latency, and allows to switch to a safe clock source in case of clock failure. The user can also switch from the safe clock back to XOSC32K in case of recovery. The safe clock is derived from the OSCULP32K oscillator with a configurable prescaler. This allows to configure the safe clock in order to fulfill the operative conditions of the microcontroller.

In sleep modes, CFD operation is automatically disabled when the external oscillator is not requested to run by a peripheral. See the Sleep Behavior table above when this is the case.

The user interface registers allow to enable, disable, and configure the CFD. The Status register provides status flags on failure and clock switch conditions. The CFD can optionally trigger an interrupt or an event when a failure is detected.

### **Clock Failure Detection**

The CFD is reset only at power-on (POR). The CFD does not monitor the XOSC32K clock when the oscillator is disabled (XOSC32K.ENABLE=0).

Before starting CFD operation, the user must start and enable the safe clock source (OSCULP32K oscillator).

CFD operation is started by writing a '1' to the CFD Enable bit in the External Oscillator Control register (CFDCTRL.CFDEN). After starting or restarting the XOSC32K, the CFD does not detect failure until the start-up time has elapsed. The start-up time is configured by the Oscillator Start-Up Time in the External Multipurpose Crystal Oscillator Control register (XOSC32K.STARTUP). Once the XOSC32K Start-Up Time is elapsed, the XOSC32K clock is constantly monitored.

During a period of 4 safe clocks (monitor period), the CFD watches for a clock activity from the XOSC32K. There must be at least one rising and one falling XOSC32K clock edge during 4 safe clock periods to meet non-failure conditions. If no or insufficient activity is detected, the failure status is asserted: The Clock Failure Detector status bit in the Status register (STATUS.CLKFAIL) and the Clock Failure Detector interrupt flag bit in the Interrupt Flag register (INTFLAG.CLKFAIL) are set. If the CLKFAIL bit in the Interrupt Enable Set register (INTENSET.CLKFAIL) is set, an interrupt is generated as well. If the Event Output enable bit in the Event Control register (EVCTRL.CFDEO) is set, an output event is generated, too.

After a clock failure was issued the monitoring of the XOSC32K clock is continued, and the Clock Failure Detector status bit in the Status register (STATUS.CLKFAIL) reflects the current XOSC32K activity.

### **Clock Switch**

When a clock failure is detected, the XOSC32K clock is replaced by the safe clock in order to maintain an active clock during the XOSC32K clock failure. The safe clock source is the OSCULP32K oscillator clock. Both 32KHz and 1KHz outputs of the XOSC32K are replaced by the respective OSCULP32K 32KHz and 1KHz outputs. The safe clock source can be scaled down by a configurable prescaler to ensure that the safe clock frequency does not exceed the operating conditions selected by the application. When the XOSC32K clock is switched to the safe clock, the Clock Switch bit in the Status register (STATUS.CLKSW) is set.

When the CFD has switched to the safe clock, the XOSC32K is not disabled. If desired, the application must take the necessary actions to disable the oscillator. The application must also take the necessary actions to configure the system clocks to continue normal operations. In the case the application can recover the XOSC32K, the application can switch back to the XOSC32K clock by writing a '1' to Switch Back Enable bit in the Clock Failure Control register (CFDCTRL.SWBACK). Once the XOSC32K clock is switched back, the Switch Back bit (CFDCTRL.SWBACK) is cleared by hardware.

### **Prescaler**

The CFD has an internal configurable prescaler to generate the safe clock from the OSCULP32K oscillator. The prescaler size allows to scale down the OSCULP32K oscillator so the safe clock frequency is not higher than the XOSC32K clock frequency monitored by the CFD. The maximum division factor is 2.

The prescaler is applied on both outputs (32KHz and 1KHz) of the safe clock.

For an external crystal oscillator at 32KHz and the OSCULP32K frequency is 32KHz, the XOSC32K.CFDPRESC should be set to 0 for a safe clock of equal frequency.

### Event

If the Event Output Enable bit in the Event Control register (EVCTRL.CFDEO) is set, the CFD clock failure will be output on the Event Output. When the CFD is switched to the safe clock, the CFD clock failure will not be output on the Event Output.

### Sleep Mode

The CFD is halted depending on configuration of the XOSC32K and the peripheral clock request. For further details, please refer to the Sleep Behavior table above. The CFD interrupt can be used to wake up the device from sleep modes.

#### 22.6.4. 32KHz Internal Oscillator (OSC32K) Operation

The OSC32K provides a tunable, low-speed, and low-power clock source.

At reset, the OSC32K is disabled. It can be enabled by setting the Enable bit in the 32KHz Internal Oscillator Control register (OSC32K.ENABLE=1). The OSC32K is disabled by clearing the Enable bit in the 32KHz Internal Oscillator Control register (OSC32K.ENABLE=0).

The frequency of the OSC32K oscillator is controlled by OSC32K.CALIB, which is a calibration value in the 32KHz Internal Oscillator Calibration bits in the 32KHz Internal Oscillator Control register. The CALIB value must be loaded with production calibration values from the NVM Software Calibration Area. When writing the Calibration bits, the user must wait for the STATUS.OSC32KRDY bit to go high before the new value is committed to the oscillator.

The OSC32K has a 32.768kHz output which is enabled by setting the 32KHz Output Enable bit in the 32KHz Internal Oscillator Control register (OSC32K.EN32K=1). The OSC32K also has a 1.024kHz clock output. This is enabled by setting the 1KHz Output Enable bit in the 32KHz Internal Oscillator Control register (OSC32K.EN1K).

Before using the USB, the Pad Calibration register (PADCAL) must be loaded with production calibration values from the NVM Software Calibration Area.

The OSC32K will behave differently in different sleep modes based on the settings of OSC32K.RUNSTDBY, OSC32K.ONDEMAND, and OSC32K.ENABLE:

**Table 22-2. OSC32K Sleep Behavior**

OSC32K.RUNSTDBY	OSC32K.ONDEMAND	OSC32K.ENABLE	Sleep Behavior
-	-	0	Disabled
0	0	1	Always run in IDLE sleep modes. Run in STANDBY sleep mode only if requested by peripheral.
1	0	1	Always run in IDLE and STANDBY sleep modes.
-	1	1	Run in IDLE or STANDBY sleep modes only if requested by a peripheral.

The OSC32K requires a start-up time. For this reason, OSC32K will keep running across resets when OSC32K.ONDEMAND=0, except for power-on reset (POR).

After such a reset, or when waking up from a sleep mode where the OSC32K was disabled, the OSC32K will need a certain amount of time to stabilize on the correct frequency.

This startup time can be configured by changing the Oscillator Start-Up Time bit group (OSC32K.STARTUP) in the OSC32K Control register. During the start-up time, the oscillator output is masked to ensure that no unstable clock propagates to the digital logic.

Once the external clock or crystal oscillator is stable and ready to be used as a clock source, the OSC32K Ready bit in the Status register is set (STATUS.OSC32KRDY=1). The transition of STATUS.OSC32KRDY from '0' to '1' generates an interrupt if the OSC32K Ready bit in the Interrupt Enable Set register is set (INTENSET.OSC32KRDY=1).

The OSC32K can be used as a source for Generic Clock Generators (GCLK) or for the Real-Time Counter (RTC). Before enabling the GCLK or the RTC module, the corresponding oscillator output must be enabled (OSC32K.EN32K or OSC32K.EN1K) in order to ensure proper operation. In the same way, the GCLK or RTC modules must be disabled before the clock selection is changed.

#### Related Links

[NVM Software Calibration Area Mapping](#) on page 37

[RTC – Real-Time Counter](#) on page 283

[Real-Time Counter Clock Selection](#) on page 219

### 22.6.5. 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 is enabled by default after a power-on reset (POR) and will always run except during POR. The frequency of the OSCULP32K oscillator is controlled by the value in the 32KHz Ultra Low Power Internal Oscillator Calibration bits in the 32KHz Ultra Low Power Internal Oscillator Control register (OSCULP32K.CALIB). This data is used to compensate for process variations.

OSCULP32K.CALIB is automatically loaded from Flash Factory Calibration during start-up. The calibration value can be overridden by the user by writing to OSCULP32K.CALIB.

It is also possible to lock the OSCULP32K configuration by setting the Write Lock bit in the 32KHz Ultra Low Power Internal Oscillator Control register (OSCULP32K.WRTLCK=1). If set, the OSCULP32K configuration is locked until a power-on reset (POR) is detected.

The OSCULP32K can be used as a source for Generic Clock Generators (GCLK) or for the Real-Time Counter (RTC). To ensure proper operation, the GCLK or RTC modules must be disabled before the clock selection is changed.

#### Related Links

[RTC – Real-Time Counter](#) on page 283

[Real-Time Counter Clock Selection](#) on page 219

[GCLK - Generic Clock Controller](#) on page 114

### 22.6.6. Watchdog Timer Clock Selection

The Watchdog Timer (WDT) uses the internal 1.024kHz OSCULP32K output clock. This clock is running all the time and internally enabled when requested by the WDT module.

## Related Links

[WDT – Watchdog Timer](#) on page 263

### 22.6.7. Real-Time Counter Clock Selection

Before enabling the RTC module, the RTC clock must be selected first. All oscillator outputs are valid as RTC clock. The selection is done in the RTC Control register (RTCCTRL). To ensure a proper operation, it is highly recommended to disable the RTC module first, before the RTC clock source selection is changed.

## Related Links

[RTC – Real-Time Counter](#) on page 283

### 22.6.8. Interrupts

The OSC32KCTRL has the following interrupt sources:

- XOSC32KRDY - 32KHz Crystal Oscillator Ready: A 0-to-1 transition on the STATUS.XOSC32KRDY bit is detected
- CLKFAIL - Clock Failure Detector: A 0-to-1 transition on the STATUS.CLKFAIL bit is detected
- OSC32KRDY - 32KHz Internal Oscillator Ready: A 0-to-1 transition on the STATUS.OSC32KRDY bit is detected

All these interrupts are synchronous wake-up source.

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 enabled individually by setting the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by setting 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 OSC32KCTRL is reset. See the INTFLAG register for details on how to clear interrupt flags.

The OSC32KCTRL 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. Refer to the INTFLAG register for details.

**Note:** Interrupts must be globally enabled for interrupt requests to be generated.

## Related Links

[PM – Power Manager](#) on page 162

[Nested Vector Interrupt Controller](#) on page 41

### 22.6.9. Events

The CFD can generate the following output event:

- Clock Failure Detector (CLKFAIL): Generated when the Clock Failure Detector status bit is set in the Status register (STATUS.CLKFAIL). The CFD event is not generated when the Clock Switch bit (STATUS.SWBACK) in the Status register is set.

Writing a '1' to an Event Output bit in the Event Control register (EVCTRL.CFDEO) enables the CFD output event. Writing a '0' to this bit disables the CFD output event. Refer to the Event System chapter for details on configuring the event system.



## 22.7. Register Summary

Offset	Name	Bit Pos.									
0x00	INTENCLR	7:0						CLKFAIL	OSC32KRDY	XOSC32KRD Y	
0x01		15:8									
0x02		23:16									
0x03		31:24									
0x04	INTENSET	7:0						CLKFAIL	OSC32KRDY	XOSC32KRD Y	
0x05		15:8									
0x06		23:16									
0x07		31:24									
0x08	INTFLAG	7:0						CLKFAIL	OSC32KRDY	XOSC32KRD Y	
0x09		15:8									
0x0A		23:16									
0x0B		31:24									
0x0C	STATUS	7:0					CLKSW		OSC32KRDY	XOSC32KRD Y	
0x0D		15:8									
0x0E		23:16									
0x0F		31:24									
0x10 ... 0x13	Reserved										
0x14	XOSC32K	7:0	ONDEMAND	RUNSTDBY		EN1K	EN32K	XTALEN	ENABLE		
0x15		15:8				WRTLOCK		STARTUP[2:0]			
0x16	CFDDCTRL	7:0						CFDPRESC	SWBACK		
0x17	EVCTRL	7:0								CFDEO	
0x18	OSC32K	7:0	ONDEMAND	RUNSTDBY			EN1K	EN32K	ENABLE		
0x19		15:8				WRTLOCK		STARTUP[2:0]			
0x1A		23:16	CALIB[6:0]								
0x1B		31:24									
0x1C	OSCULP32K	7:0									
0x1D		15:8	WRTLOCK				CALIB[4:0]				

## 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.

All registers with write-access can be write-protected optionally by the peripheral access controller (PAC). Optional Write-Protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write-Protection" property in the register description. Write-protection does not apply to accesses through an external debugger.

### Related Links

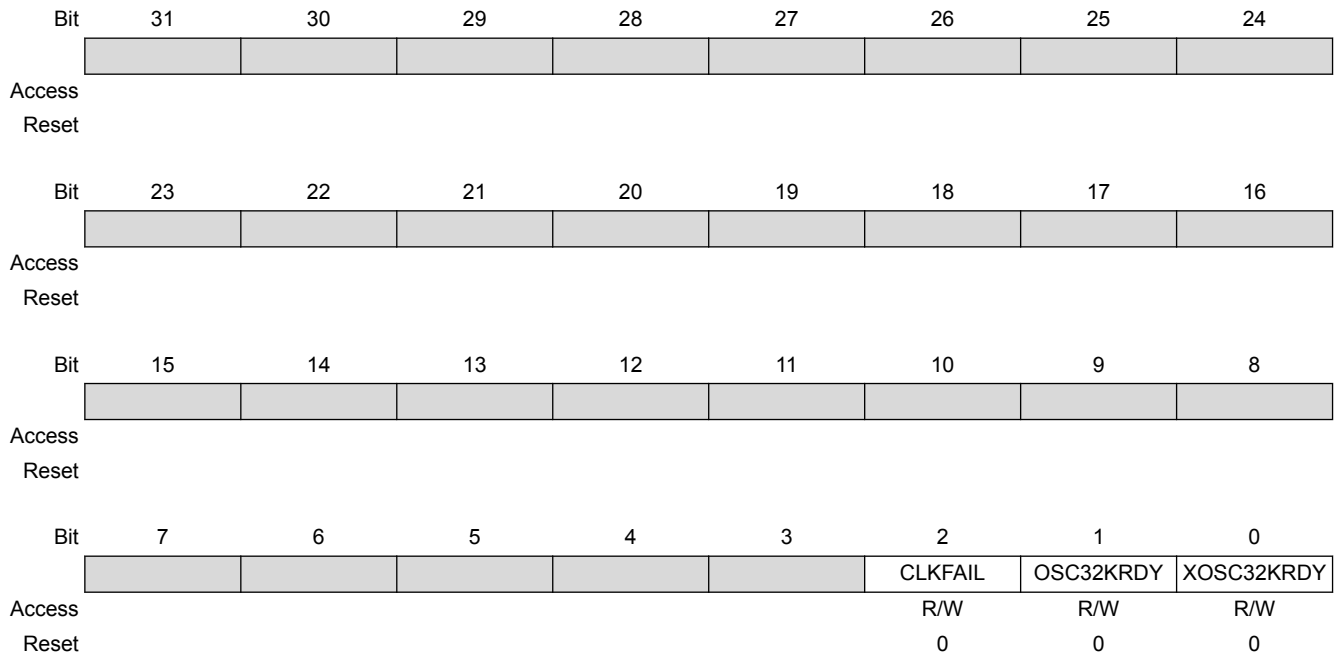




### 22.8.1. Interrupt Enable Clear

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

**Name:** INTENCLR  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



#### Bit 2 – CLKFAIL: XOSC32K Clock Failure Detection Interrupt Enable

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

Writing a '1' to this bit will clear the XOSC32K Clock Failure Interrupt Enable bit, which disables the XOSC32K Clock Failure interrupt.

Value	Description
0	The XOSC32K Clock Failure Detection is disabled.
1	The XOSC32K Clock Failure Detection is enabled. An interrupt request will be generated when the XOSC32K Clock Failure Detection interrupt flag is set.

#### Bit 1 – OSC32KRDY: OSC32K Ready Interrupt Enable

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

Writing a '1' 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.

**Bit 0 – XOSC32KRDY: XOSC32K Ready Interrupt Enable**

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

Writing a '1' 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.

## 22.8.2. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x04

**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
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
Access						R/W	R/W	R/W
Reset						0	0	0

### Bit 2 – CLKFAIL: XOSC32K Clock Failure Detection Interrupt Enable

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

Writing a '1' to this bit will set the XOSC32K Clock Failure Interrupt Enable bit, which enables the XOSC32K Clock Failure interrupt.

Value	Description
0	The XOSC32K Clock Failure Detection is disabled.
1	The XOSC32K Clock Failure Detection is enabled. An interrupt request will be generated when the XOSC32K Clock Failure Detection interrupt flag is set.

### Bit 1 – OSC32KRDY: OSC32K Ready Interrupt Enable

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

Writing a '1' 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.

**Bit 0 – XOSC32KRDY: XOSC32K Ready Interrupt Enable**

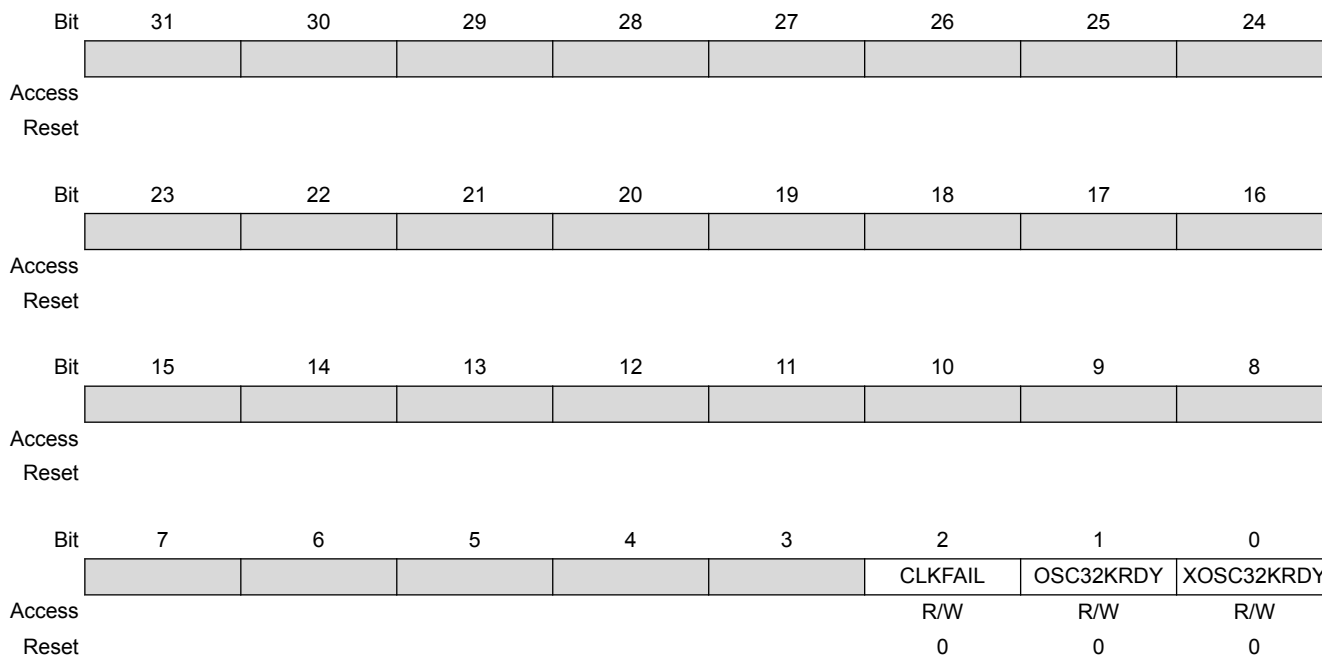
Writing a '0' to this bit has no effect.

Writing a '1' 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.

### 22.8.3. Interrupt Flag Status and Clear

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



#### Bit 2 – CLKFAIL: XOSC32K Clock Failure Detection

This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the XOSC32K Clock Failure Detection bit in the Status register (STATUS.CLKFAIL) and will generate an interrupt request if INTENSET.CLKFAIL is '1'.

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

Writing a '1' to this bit will clear the XOSC32K Clock Failure Detection flag.

#### Bit 1 – OSC32KRDY: OSC32K Ready

This flag is cleared by writing a '1' to it.

This flag is set by a zero-to-one transition of the OSC32K Ready bit in the Status register (STATUS.OSC32KRDY), and will generate an interrupt request if INTENSET.OSC32KRDY=1.

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

Writing a '1' to this bit clears the OSC32K Ready interrupt flag.

#### Bit 0 – XOSC32KRDY: XOSC32K Ready

This flag is cleared by writing a '1' to it.

This flag is set by a zero-to-one transition of the XOSC32K Ready bit in the Status register (STATUS.XOSC32KRDY), and will generate an interrupt request if INTENSET.XOSC32KRDY=1.

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

Writing a '1' to this bit clears the XOSC32K Ready interrupt flag.

## 22.8.4. Status

**Name:** STATUS  
**Offset:** 0x0C  
**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
Access					R		R	R
Reset					0		0	0

### Bit 3 – CLKSW: XOSC32K Clock Switch

Value	Description
0	XOSC32K is not switched and provided the crystal oscillator.
1	XOSC32K is switched to be provided by the safe clock.

### Bit 3 – CLKFAIL: XOSC32K Clock Failure Detector

Value	Description
0	XOSC32K is passing failure detection.
1	XOSC32K is not passing failure detection.

### Bit 1 – OSC32KRDY: OSC32K Ready

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

### Bit 0 – XOSC32KRDY: XOSC32K Ready



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

## 22.8.5. 32KHz External Crystal Oscillator (XOSC32K) Control

**Name:** XOSC32K  
**Offset:** 0x14  
**Reset:** 0x00000080  
**Property:** PAC Write-Protection

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		EN1K	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, effectively freezing 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 22-3. Start-Up Time for 32KHz External Crystal Oscillator**

STARTUP[2:0]	Number of OSCULP32K Clock Cycles	Number of XOSC32K Clock Cycles	Approximate Equivalent Time [ms]
0x0	2048	3	62.592
0x1	4096	3	125.092
0x2	16384	3	500.092
0x3	32768	3	100.0092
0x4	65536	3	200.0092
0x5	131072	3	400.0092
0x6	262144	3	800.0092
0x7	-	-	Reserved

#### Note:

1. Actual Start-Up time is 1 OSCULP32K cycle + 3 XOSC32K cycles.
2. The given time assumes an XTAL frequency of 32.768kHz.

**Bit 7 – ONDEMAND: On Demand Control**

This bit controls how the XOSC32K behaves when a peripheral clock request is detected. For details, refer to [XOSC32K Sleep Behavior](#).

**Bit 6 – RUNSTDBY: Run in Standby**

This bit controls how the XOSC32K behaves during standby sleep mode. For details, refer to [XOSC32K Sleep Behavior](#).

**Bit 4 – EN1K: 1KHz Output Enable**

Value	Description
0	The 1KHz output is disabled.
1	The 1KHz output is enabled.

**Bit 3 – EN32K: 32KHz Output Enable**

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**

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

## 22.8.6. Clock Failure Detector Control

**Name:** CFDDCTRL  
**Offset:** 0x16  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

### Bit 2 – CFDPRESC: Clock Failure Detector Prescaler

This bit selects the prescaler for the Clock Failure Detector.

Value	Description
0	The CFD safe clock frequency is the OSCULP32K frequency
1	The CFD safe clock frequency is the OSCULP32K frequency divided by 2

### Bit 2 – CFDEN: Clock Failure Detector Enable

This bit selects the Clock Failure Detector state.

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

### Bit 1 – SWBACK: Clock Switch Back

This bit controls the XOSC32K output switch back to the external clock or crystal scillator in case of clock recovery.

Value	Description
0	The clock switch is disabled.
1	The clock switch is enabled. This bit is reset when the XOSC32K output is switched back to the external clock or crystal oscillator.

## 22.8.7. Event Control

**Name:** EVCTRL  
**Offset:** 0x17  
**Reset:** 0x0  
**Property:** PAC Write-Protection

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

### Bit 0 – CFDEO: Clock Failure Detector Event Out

This bit controls whether the Clock Failure Detector event output is enabled and an event will be generated when the CFD detects a clock failure.

Value	Description
0	Clock Failure Detector Event output is disabled, no event will be generated.
1	Clock Failure Detector Event output is enabled, an event will be generated.

## 22.8.8. 32KHz Internal Oscillator (OSC32K) Control

**Name:** OSC32K  
**Offset:** 0x18  
**Reset:** 0x0000 0080 (Writing action by User required)  
**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		CALIB[6:0]						
Reset		-	-	-	-	-	-	-
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	R/W	
Reset	1	0			0	0	0	

### Bits 22:16 – CALIB[6:0]: Oscillator Calibration

These bits control the oscillator calibration. The calibration values must be loaded by the user from the NVM Software Calibration Area.

### Bit 12 – WRTLOCK: Write Lock

This bit locks the OSC32K register for future writes, effectively freezing 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 start-up counter.

**Table 22-4. Start-Up Time for 32KHz Internal Oscillator**

STARTUP[2:0]	Number of OSC32K clock cycles	Approximate Equivalent Time [ms]
0x0	3	0.092
0x1	4	0.122
0x2	6	0.183

STARTUP[2:0]	Number of OSC32K clock cycles	Approximate Equivalent Time [ms]
0x3	10	0.305
0x4	18	0.549
0x5	34	1.038
0x6	66	2.014
0x7	130	3.967

**Note:**

1. Start-up time is given by STARTUP + three OSC32K cycles.
2. The given time assumes an XTAL frequency of 32.768kHz.

**Bit 7 – ONDEMAND: On Demand Control**

This bit controls how the OSC32K behaves when a peripheral clock request is detected. For details, refer to [OSC32K Sleep Behavior](#).

**Bit 6 – RUNSTDBY: Run in Standby**

This bit controls how the OSC32K behaves during standby sleep mode. For details, refer to [OSC32K Sleep Behavior](#).

**Bit 3 – EN1K: 1KHz Output Enable**

Value	Description
0	The 1KHz output is disabled.
1	The 1KHz output is enabled.

**Bit 2 – EN32K: 32KHz Output Enable**

Value	Description
0	The 32KHz output is disabled.
1	The 32KHz output is enabled.

**Bit 1 – ENABLE: Oscillator Enable**

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

## 22.8.9. 32KHz Ultra Low Power Internal Oscillator (OSCULP32K) Control

**Name:** OSCULP32K  
**Offset:** 0x1C  
**Reset:** 0x0000XX06  
**Property:** Write-Protected

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

### Bit 15 – 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 12:8 – CALIB[4:0]: Oscillator Calibration

These bits control the oscillator calibration.

These bits are loaded from Flash Calibration at startup.



## 23. SUPC – Supply Controller

### 23.1. Overview

The Supply Controller (SUPC) manages the voltage reference and power supply of the device.

The SUPC controls the voltage regulators for the core (VDDCORE) domain. It sets the voltage regulators according to the sleep modes, or the user configuration. In active mode, the voltage regulators can be selected on the fly between LDO (low-dropout) type regulator or Buck converter.

The SUPC embeds two Brown-Out Detectors. BODVDD monitors the voltage applied to the device (VDD) and BODCORE monitors the internal voltage to the core (VDDCORE). The BOD can monitor the supply voltage continuously (continuous mode) or periodically (sampling mode).

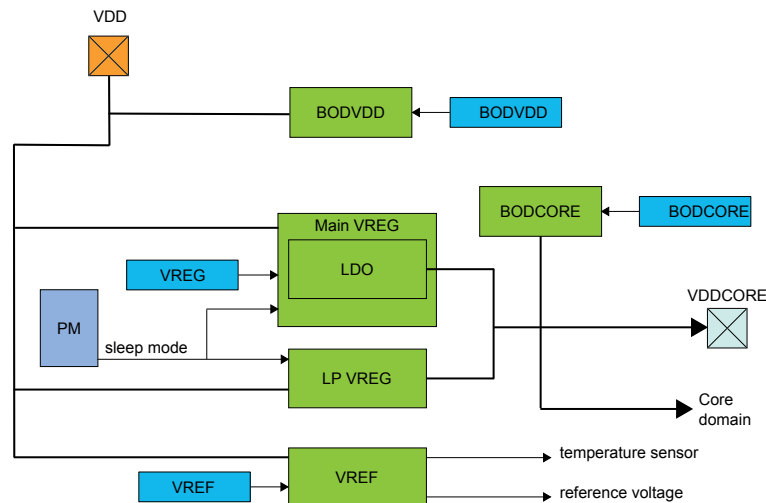
The SUPC generates also a selectable reference voltage and a voltage dependent on the temperature which can be used by analog modules like the ADC.

### 23.2. Features

- Voltage Regulator System
  - Main voltage regulator: LDO in active mode (MAINVREG)
  - Low Power voltage regulator in mode (LPVREG)
- Voltage Reference System
  - Reference voltage for ADC and DAC
  - Temperature sensor
- VDD Brown-Out Detector (BODVDD)
  - Programmable threshold
  - Threshold value loaded from NVM User Row at startup
  - Triggers resets or interrupts. Action loaded from NVM User Row
  - Operating modes:
    - Continuous mode
    - Sampled mode for low power applications with programmable sample frequency
  - Hysteresis value from Flash User Calibration
- VDDCORE Brown-Out Detector (BODCORE)
  - 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 with programmable sample frequency
  - Hysteresis value loaded from NVM User Row at startup

## 23.3. Block Diagram

Figure 23-1. SUPC Block Diagram



## 23.4. Signal Description

Signal Name	Type	Description
OUT[1:0]	Digital Output	SUPC Outputs
PSOK	Digital Input	Main Power Supply OK

One signal can be mapped on several pins.

### Related Links

[I/O Multiplexing and Considerations](#) on page 27

## 23.5. Product Dependencies

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

### 23.5.1. I/O Lines

I/O lines are configured by SUPC either when the SUPC output (signal OUT) is enabled or when the PSOK input is enabled. The I/O lines need no user configuration.

### 23.5.2. Power Management

The SUPC can operate in all sleep modes.

### Related Links

[PM – Power Manager](#) on page 162

### 23.5.3. Clocks

The SUPC bus clock (CLK\_SUPC\_APB) can be enabled and disabled in the Main Clock module.

A 32KHz clock, asynchronous to the user interface clock (CLK\_SUPC\_APB), is required to run BODVDD and BODCORE in sampled mode. Due to this asynchronicity, writing to certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) on page 243 for further details.

#### Related Links

[OSC32KCTRL – 32KHz Oscillators Controller](#) on page 211

[Peripheral Clock Masking](#) on page 135

#### 23.5.4. DMA

Not applicable.

#### 23.5.5. Interrupts

The interrupt request lines are connected to the interrupt controller. Using the SUPC interrupts requires the interrupt controller to be configured first.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

#### 23.5.6. Events

Not applicable.

#### 23.5.7. Debug Operation

When the CPU is halted in debug mode, the SUPC continues normal operation. If the SUPC 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 a debugger connection is detected by the system, BODVDD and BODCORE resets will be blocked.

#### 23.5.8. Register Access Protection

Registers with write-access can be write-protected optionally by the peripheral access controller (PAC).

**Note:** Not all registers with write-access can be write-protected.

PAC Write-Protection is not available for the following registers:

- Interrupt Flag Status and Clear register (INTFLAG)

Optional PAC Write-Protection is denoted by the "PAC Write-Protection" property in each individual register description.

#### Related Links

[PAC - Peripheral Access Controller](#) on page 46

#### 23.5.9. Analog Connections

Not applicable.

### 23.6. Functional Description

#### 23.6.1. Voltage Regulator System Operation

##### 23.6.1.1. Enabling, Disabling, and Resetting

The LDO main voltage regulator is enabled after any Reset. The main voltage regulator (MAINVREG) can be disabled by writing the Enable bit in the VREG register (VREG.ENABLE) to zero. The main voltage regulator output supply level is automatically defined by the sleep mode selected in the Power Manager module.

#### Related Links

[PM – Power Manager](#) on page 162

### 23.6.1.2. Initialization

After a Reset, the LDO voltage regulator supplying VDDCORE is enabled.

### 23.6.1.3. Sleep Mode Operation

In mode, the low power voltage regulator (LPVREG) is used to supply VDDCORE.

#### Related Links

[Sleep Mode Controller](#) on page 164

## 23.6.2. Voltage Reference System Operation

### 23.6.2.1. Initialization

The voltage reference output and the temperature sensor are disabled after any Reset.

### 23.6.2.2. Enabling, Disabling, and Resetting

The voltage reference output is enabled/disabled by setting/clearing the Voltage Reference Output Enable bit in the Voltage Reference register (VREF.VREFOE).

The temperature sensor is enabled/disabled by setting/clearing the Temperature Sensor Enable bit in the Voltage Reference register (VREF.TSEN).

**Note:** It is not recommended to enable both voltage reference output and temperature sensor at the same time - only the voltage reference output will be present at both ADC inputs.

### 23.6.2.3. Selecting a Voltage Reference

The SEL bit group in the VREF register (VREF.SEL) selects the reference voltage to be applied to analog modules, e.g. the ADC.

### 23.6.2.4. Sleep Mode Operation

The Voltage Reference output during sleep mode can be configured using the Run in Standby bit and the On Demand bit in the VREF register (VREF.RUNSTDBY, VREF.ONDEMAND), see table VREF Sleep Mode Operation.

When the temperature sensor is enabled (VREF.TSEN=1), it will be always active in all sleep modes *including* standby sleep mode.

**Table 23-1. VREF Sleep Mode Operation**

VREF.VREFOE	VREF.ONDEMAND	VREF.RUNSTDBY	Sleep behavior
0	-	-	Disable
1	0	0	Always run in all sleep modes <i>except</i> standby sleep mode
1	0	1	Always run in all sleep modes <i>including</i> standby sleep mode
1	1	0	Only run if requested by the ADC, in all sleep modes <i>except</i> standby sleep mode
1	1	1	Only run if requested by the ADC, in all sleep modes <i>including</i> standby sleep mode

## 23.6.3. Brown-Out Detectors

### 23.6.3.1. Initialization

Before a Brown Out Detector (BODCORE or BODVDD) is enabled, it must be configured, as outlined by the following:

- Set the BOD threshold level (BODnn.LEVEL)
- Set the configuration in active, standby, backup modes (BODnn.ACTCDG, BODnn.STDBYCFG, BODnn.BKUP)
- Set the prescaling value if the BOD will run in sampling mode (BODnn.PSEL)
- Set the action and hysteresis (BODnn.ACTION and BODnn.HYST)

The BODCORE and BODVDD registers are Enable-Protected, meaning that they can only be written when the respective BOD is disabled (BODCORE.ENABLE=0 and SYNCBUSY.BODCOREEN=0, or BODVDD.ENABLE=0 and SYNCBUSY.BODVDDEN=0, respectively). As long as the Enable bit is '1', any writes to Enable-Protected registers will be discarded, and an APB error will be generated. The Enable bits are not Enable-Protected.

### 23.6.3.2. Enabling, Disabling, and Resetting

After power or user reset, the BODVDD and BODCORE register values are loaded from the NVM User Row.

The BODs are enabled by writing a '1' to the Enable bit in the respective BOD control register (BODCORE.ENABLE or BODVDD.ENABLE). The BODs are disabled by writing a '0' to the respective BODnn.ENABLE.

#### Related Links

[NVM User Row Mapping](#) on page 36

### 23.6.3.3. VDD Brown-Out Detector (BODVDD)

The VDD Brown-Out Detector (BODVDD) is able to monitor either the VDD or the VBAT supply and compares the voltage with the brown-out threshold level set in the BODVDD Level field (BODVDD.LEVEL) in the BODVDD register.

The Voltage Monitored bit in the BODVDD Control register (BODVDD.VMON) selects which supply is monitored in active and standby mode. In hibernate and backup mode, BODVDD will always monitor the supply of the backup domain, i.e. either VDD or VBAT.

When VDD crosses below the brown-out threshold level, the BODVDD can generate either an interrupt or a Reset, depending on the BODVDD Action bit field (BODVDD.ACTION).

The BODVDD detection status can be read from the BODVDD Detection bit in the Status register (STATUS.BODVDDDET).

At start-up or at Power-On Reset (POR), the BODVDD register values are loaded from the NVM User Row.

#### Related Links

[NVM User Row Mapping](#) on page 36

### 23.6.3.4. VDDCORE Brown-Out Detector (BODCORE)

The VDDCORE Brown-Out Detector (BODCORE) monitors the VDDCORE supply and compares the voltage with the brown-out threshold level set in the BODCORE Threshold Level bit field in the BODCORE register (BODCORE.LEVEL).

When VDDCORE crosses below the brown-out threshold level, the BODCORE can generate either an interrupt or a reset.

The BODCORE detection status can be read from the BODCORE Detection bit (STATUS.BODCOREDET) in the STATUS register.

At start-up including power-on reset (POR), the BODCORE register values are loaded from the NVM User Row.

## Related Links

[NVM User Row Mapping](#) on page 36

### 23.6.3.5. Continuous Mode

Continuous mode is the default mode for both BODCORE and BODVDD.

The *BODVDD* is continuously monitoring the VDD supply voltage if it is enabled (BODVDD.ENABLE=1) and if the BODVDD Configuration bit in the BODVDD register is cleared (BODVDD.ACTCFG=0 for active mode, BODVDD.STDBYCFG=0 for standby mode).

The *BODCORE* is continuously monitoring the VDDCORE supply voltage if it is enabled (BODCORE.ENABLE=1) and if the BODCORE Configuration bit in the BODCORE register is cleared (BODCORE.ACTCFG=0 and/or BODCORE.STDBYCFG, respectively).

### 23.6.3.6. Sampling Mode

The Sampling Mode is a low-power mode where the BODVDD or BODCORE are being repeatedly enabled on a sampling clock's ticks. The BODVDD (or BODCORE) will monitor the supply voltage (or core 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 in Active mode for each BOD by writing the respective ACTCFG bit (BODCORE.ACTCFG=1 and BODVDD.ACTCFG=1). Sampling mode is enabled in Standby mode by writing to the respective STDBYCFG bit (BODnn.STBYCFG=1). The frequency of the clock ticks ( $F_{clk\text{sampling}}$ ) is controlled by the Prescaler Select bit groups in the respective BOD registers (BODVDD.PSEL and BODCORE.PSEL, respectively).

$$F_{clk\text{sampling}} = \frac{F_{clk\text{prescaler}}}{2^{(PSEL + 1)}}$$

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

As the sampling clock is different from the APB clock domain, synchronization among the clocks is necessary. See also [Synchronization](#) on page 243.

### 23.6.3.7. Hysteresis

A hysteresis on the trigger threshold of a BOD will reduce the sensitivity to ripples on the monitored voltage.

Enabling the BODVDD hysteresis by writing the Hysteresis bit in the BODVDD register (BODVDD.HYST) to '1' will add hysteresis to the BODVDD threshold level. BODCORE.HYST=1 will add hysteresis to the BODCORE threshold level.

The hysteresis functionality can be used in both Continuous and Sampling Mode.

### 23.6.3.8. Sleep Mode Operation

#### Standby Mode

The BODVDD and the BODCORE can be used in standby mode if the BOD is enabled and the corresponding Run in Standby bit is written to '1' (BODVDD.RUNSTDBY, BODCORE.RUNSTDBY).

Both BODs can be configured to work in either Continuous or Sampling Mode by writing a '1' to the respective Configuration in Standby Sleep Mode bit (BODVDD.STDBYCFG, BODCORE.STDBYCFG).

## 23.6.4. Interrupts

The SUPC has the following interrupt sources, which are either synchronous or asynchronous wake-up sources:

- BODVDD Ready (BODVDDRDY), synchronous
- BODVDD Detection (BODVDDDET), asynchronous
- BODVDD Synchronization Ready (BVDDSRDY), synchronous
- BODCORE Ready (BODCORERDY), synchronous
- BODCORE Detection (BODCOREDET), asynchronous
- BODCORE Synchronization Ready (BODCORESRDY), synchronous

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 SUPC is reset. See the INTFLAG register for details on how to clear interrupt flags. The SUPC 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.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

[Sleep Mode Controller](#) on page 164

### 23.6.5. Synchronization

The prescaler counters that are used to trigger brown-out detections operate asynchronously from the peripheral bus. As a consequence, the BODCORE and BODVDD Enable bits (BODCORE.ENABLE and BODVDD.ENABLE, respectively) need synchronization when written.

The Write-Synchronization of the Enable bit is triggered by writing a '1' to the Enable bit of the BODCORE or the BODVDD Control register. The Synchronization Ready bit (STATUS.BCORESRDY or STATUS.BVDDSRDY) in the STATUS register will be cleared when the Write-Synchronization starts, and set again when the Write-Synchronization is complete. Writing to the same register while the Write-Synchronization is ongoing (STATUS.BVDDSRDY or STATUS.BCORESRDY are '0') will generate an error without stalling the APB bus.

## 23.7. Register Summary

Offset	Name	Bit Pos.								
0x00	INTENCLR	7:0			BCORES RDY	BODCORE DE T	BODCOR ERD Y	BVDDSRDY	BODVDDDET	BODVDDR DY
0x01		15:8								
0x02		23:16								
0x03		31:24								
0x04	INTENSET	7:0			BCORES RDY	BODCORE DE T	BODCOR ERD Y	BVDDSRDY	BODVDDDET	BODVDDR DY
0x05		15:8								
0x06		23:16								
0x07		31:24								
0x08	INTFLAG	7:0			BCORES RDY	BODCORE DE T	BODCOR ERD Y	BVDDSRDY	BODVDDDET	BODVDDR DY
0x09		15:8								
0x0A		23:16								
0x0B		31:24								
0x0C	STATUS	7:0			BCORES RDY	BODCORE DE T	BODCOR ERD Y	BVDDSRDY	BODVDDDET	BODVDDR DY
0x0D		15:8								
0x0E		23:16								
0x0F		31:24								
0x10	BODVDD	7:0		RUNSTDBY	STDBYCFG	ACTION[1:0]		HYST	ENABLE	
0x11		15:8	PSEL[3:0]							ACTCFG
0x12		23:16	LEVEL[5:0]							
0x13		31:24								
0x14	BODCORE	7:0		RUNSTDBY	STDBYCFG	ACTION[1:0]		HYST	ENABLE	
0x15		15:8	PSEL[3:0]							ACTCFG
0x16		23:16	LEVEL[5:0]							
0x17		31:24								
0x18	VREG	7:0		RUNSTDBY					ENABLE	
0x19		15:8								
0x1A		23:16								
0x1B		31:24								
0x1C	VREF	7:0	ONDEMAND	RUNSTDBY				VREFOE	TSEN	
0x1D		15:8								
0x1E		23:16	SEL[3:0]							
0x1F		31:24								

## 23.8. Register Description

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



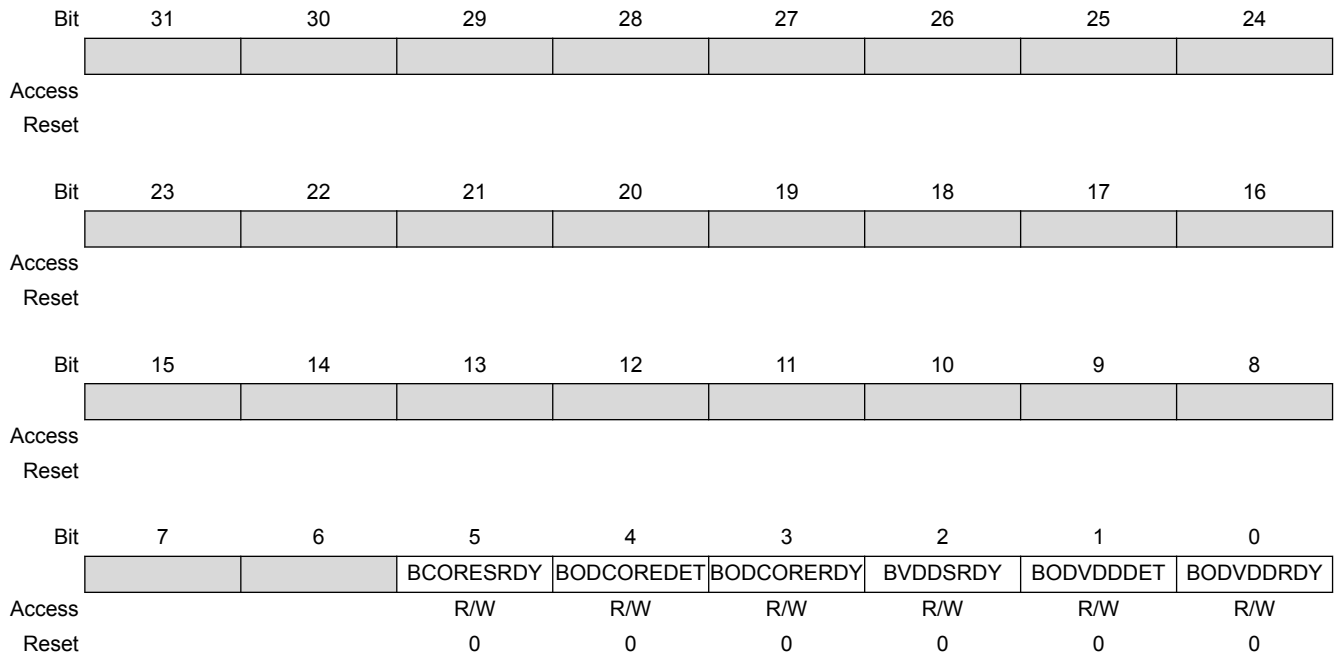
Some registers are optionally write-protected by the Peripheral Access Controller (PAC). PAC Write-protection is denoted by the "PAC Write-Protection" property in each individual register description. Refer to [Register Access Protection](#) on page 239 for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the "Write-Synchronized" or the "Read-Synchronized" property in each individual register description. Refer to [Synchronization](#) on page 243 for details.

### 23.8.1. Interrupt Enable Clear

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

**Name:** INTENCLR  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



#### Bit 5 – BCORESRDY: BODCORE Synchronization Ready Interrupt Enable

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

Writing a '1' to this bit will clear the BODCORE Synchronization Ready Interrupt Enable bit, which disables the BODCORE Synchronization Ready interrupt.

Value	Description
0	The BODCORE Synchronization Ready interrupt is disabled.
1	The BODCORE Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the BODCORE Synchronization Ready Interrupt flag is set.

#### Bit 4 – BODCOREDET: BODCORE Detection Interrupt Enable

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

Writing a '1' to this bit will clear the BODCORE Detection Interrupt Enable bit, which disables the BODCORE Detection interrupt.

Value	Description
0	The BODCORE Detection interrupt is disabled.
1	The BODCORE Detection interrupt is enabled, and an interrupt request will be generated when the BODCORE Detection Interrupt flag is set.

### Bit 3 – BODCORERDY: BODCORE Ready Interrupt Enable

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

Writing a '1' to this bit will clear the BODCORE Ready Interrupt Enable bit, which disables the BODCORE Ready interrupt.

Value	Description
0	The BODCORE Ready interrupt is disabled.
1	The BODCORE Ready interrupt is enabled and an interrupt request will be generated when the BODCORE Ready Interrupt flag is set.

### Bit 2 – BVDDSRDY: BODVDD Synchronization Ready Interrupt Enable

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

Writing a '1' to this bit will clear the BODVDD Synchronization Ready Interrupt Enable bit, which disables the BODVDD Synchronization Ready interrupt.

Value	Description
0	The BODVDD Synchronization Ready interrupt is disabled.
1	The BODVDD Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the BODVDD Synchronization Ready Interrupt flag is set.

### Bit 1 – BODVDDDET: BODVDD Detection Interrupt Enable

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

Writing a '1' to this bit will clear the BODVDD Detection Interrupt Enable bit, which disables the BODVDD Detection interrupt.

Value	Description
0	The BODVDD Detection interrupt is disabled.
1	The BODVDD Detection interrupt is enabled, and an interrupt request will be generated when the BODVDD Detection Interrupt flag is set.

### Bit 0 – BODVDDRDY: BODVDD Ready Interrupt Enable

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

Writing a '1' to this bit will clear the BODVDD Ready Interrupt Enable bit, which disables the BODVDD Ready interrupt.

Value	Description
0	The BODVDD Ready interrupt is disabled.
1	The BODVDD Ready interrupt is enabled, and an interrupt request will be generated when the BODVDD Ready Interrupt flag is set.

### 23.8.2. Interrupt Enable Set

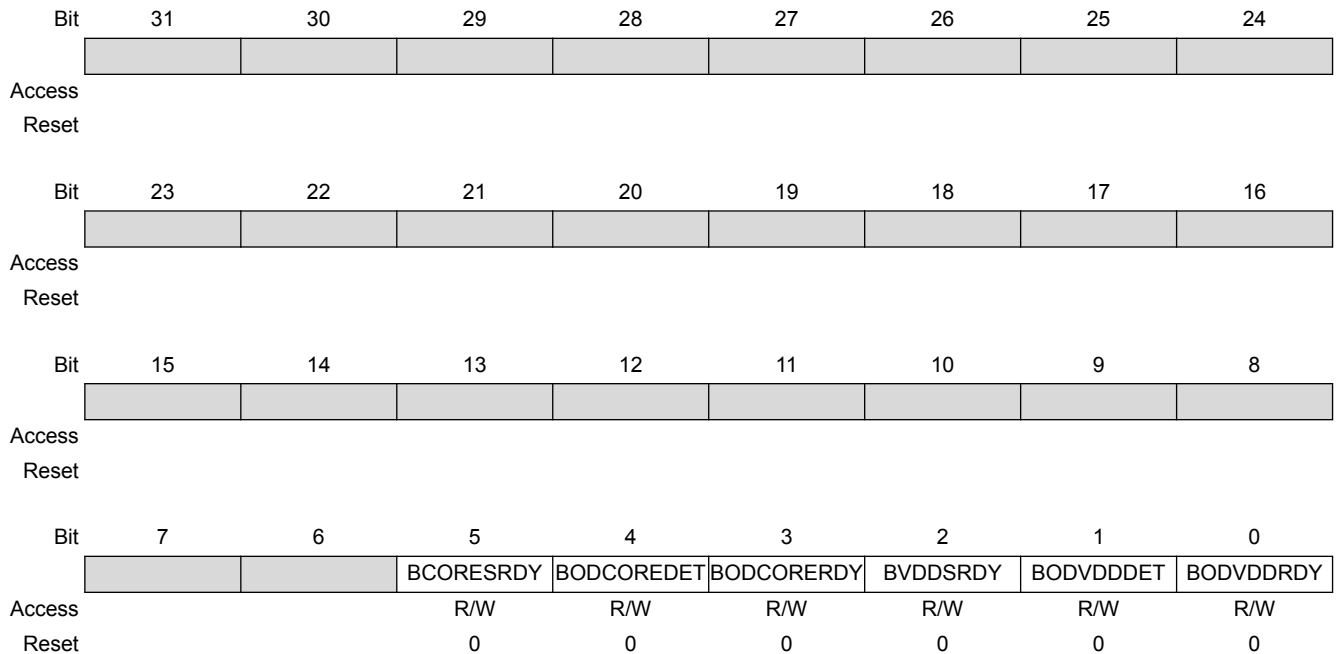
This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

**Name:** INTENSET

**Offset:** 0x04

**Reset:** 0x00000000

**Property:** PAC Write-Protection



#### Bit 5 – BCOESRDY: BODCORE Synchronization Ready Interrupt Enable

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

Writing a '1' to this bit will set the BODCORE Synchronization Ready Interrupt Enable bit, which enables the BODCORE Synchronization Ready interrupt.

Value	Description
0	The BODCORE Synchronization Ready interrupt is disabled.
1	The BODCORE Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the BODCORE Synchronization Ready Interrupt flag is set.

#### Bit 4 – BODCOREDET: BODCORE Detection Interrupt Enable

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

Writing a '1' to this bit will set the BODCORE Detection Interrupt Enable bit, which enables the BODCORE Detection interrupt.

Value	Description
0	The BODCORE Detection interrupt is disabled.
1	The BODCORE Detection interrupt is enabled and an interrupt request will be generated when the BODCORE Detection Interrupt Flag is set.

### Bit 3 – BODCORERDY: BODCORE Ready Interrupt Enable

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

Writing a '1' to this bit will set the BODCORE Ready Interrupt Enable bit, which enables the BODCORE Ready interrupt.

Value	Description
0	The BODCORE Ready interrupt is disabled.
1	The BODCORE Ready interrupt is enabled, and an interrupt request will be generated when the BODCORE Ready Interrupt flag is set.

### Bit 2 – BVDDSRDY: BODVDD Synchronization Ready Interrupt Enable

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

Writing a '1' to this bit will set the BODVDD Synchronization Ready Interrupt Enable bit, which enables the BODVDD Synchronization Ready interrupt.

Value	Description
0	The BODVDD Synchronization Ready interrupt is disabled.
1	The BODVDD Synchronization Ready interrupt is enabled, and an interrupt request will be generated when the BODVDD Synchronization Ready Interrupt flag is set.

### Bit 1 – BODVDDDET: BODVDD Detection Interrupt Enable

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

Writing a '1' to this bit will set the BODVDD Detection Interrupt Enable bit, which enables the BODVDD Detection interrupt.

Value	Description
0	The BODVDD Detection interrupt is disabled.
1	The BODVDD Detection interrupt is enabled, and an interrupt request will be generated when the BODVDD Detection Interrupt flag is set.

### Bit 0 – BODVDDRDY: BODVDD Ready Interrupt Enable

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

Writing a '1' to this bit will set the BODVDD Ready Interrupt Enable bit, which enables the BODVDD Ready interrupt.

Value	Description
0	The BODVDD Ready interrupt is disabled.
1	The BODVDD Ready interrupt is enabled, and an interrupt request will be generated when the BODVDD Ready Interrupt flag is set.

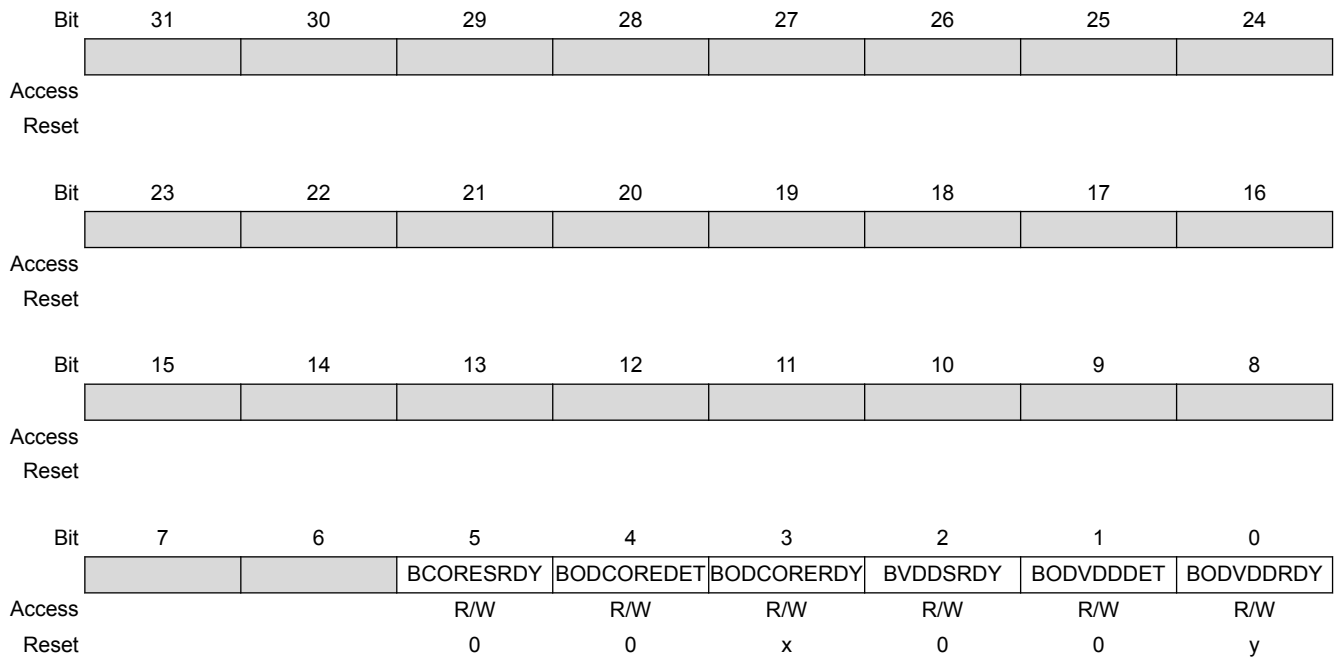
### 23.8.3. Interrupt Flag Status and Clear

**Name:** INTFLAG

**Offset:** 0x08

**Reset:** 0x0000010X - X= determined from NVM User Row (0xX=0bx00y)

**Property:** -



#### Bit 5 – BCORESRDY: BODCORE Synchronization Ready

This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the BODCORE Synchronization Ready bit in the Status register (STATUS.BCORESRDY) and will generate an interrupt request if INTENSET.BCORESRDY=1.

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

Writing a '1' to this bit clears the BODCORE Synchronization Ready interrupt flag.

#### Bit 4 – BODCOREDET: BODCORE Detection

This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the BODCORE Detection bit in the Status register (STATUS.BODCOREDET) and will generate an interrupt request if INTENSET.BODCOREDET=1.

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

Writing a '1' to this bit clears the BODCORE Detection interrupt flag.

#### Bit 3 – BODCORERDY: BODCORE Ready

This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the BODCORE Ready bit in the Status register (STATUS.BODCORERDY) and will generate an interrupt request if INTENSET.BODCORERDY=1.

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

Writing a '1' to this bit clears the BODCORE Ready interrupt flag.

The BODCORE can be enabled at startup from Flash User Row.

**Bit 2 – BVDDSRDY: BODVDD Synchronization Ready**

This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the BODVDD Synchronization Ready bit in the Status register (STATUS.BVDDSRDY) and will generate an interrupt request if INTENSET.BVDDSRDY=1.

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

Writing a '1' to this bit clears the BODVDD Synchronization Ready interrupt flag.

**Bit 1 – BODVDDDET: BODVDD Detection**

This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the BODVDD Detection bit in the Status register (STATUS.BODVDDDET) and will generate an interrupt request if INTENSET.BODVDDDET=1.

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

Writing a '1' to this bit clears the BODVDD Detection interrupt flag.

**Bit 0 – BODVDDRDY: BODVDD Ready**

This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the BODVDD Ready bit in the Status register (STATUS.BODVDDRDY) and will generate an interrupt request if INTENSET.BODVDDRDY=1.

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

Writing a '1' to this bit clears the BODVDD Ready interrupt flag.

The BODVDD can be enabled.

## 23.8.4. Status

**Name:** STATUS  
**Offset:** 0x0C  
**Reset:** Determined from NVM User Row  
**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
Access			R	R	R	R	R	R
Reset			0	0	x	0	0	y

### Bit 5 – BCORESRDY: BODCORE Synchronization Ready

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

### Bit 4 – BODCOREDET: BODCORE Detection

Value	Description
0	No BODCORE detection.
1	BODCORE has detected that the core power supply is going below the BODCORE reference value.

### Bit 3 – BODCORERDY: BODCORE Ready

The BODCORE can be enabled at start-up from NVM User Row.

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

### Bit 2 – BVDDSRDY: BODVDD Synchronization Ready



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

**Bit 1 – BODVDDDET: BODVDD Detection**

Value	Description
0	No BODVDD detection.
1	BODVDD has detected that the I/O power supply is going below the BODVDD reference value.

**Bit 0 – BODVDDRDY: BODVDD Ready**

The BODVDD can be enabled at start-up from NVM User Row.

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

### 23.8.5. VDD Brown-Out Detector (BODVDD) Control

**Name:** BODVDD

**Offset:** 0x10

**Reset:** Determined from NVM User Row

**Property:** Write-Synchronized, Enable-Protected, PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access			LEVEL[5:0]					
Reset			x	x	x	x	x	x
Bit	15	14	13	12	11	10	9	8
Access	PSEL[3:0]							ACTCFG
Reset	0	0	0	0				0
Bit	7	6	5	4	3	2	1	0
Access		RUNSTDBY	STDBYCFG	ACTION[1:0]		HYST	ENABLE	
Reset		0	0	y	y	0	z	

#### Bits 21:16 – LEVEL[5:0]: BODVDD Threshold Level on VDD

This field sets the triggering voltage threshold for the BODVDD when the BODVDD monitors VDD except in backup sleep mode.

These bits are loaded from NVM User Row at start-up.

This field is not synchronized.

#### Bits 15:12 – PSEL[3:0]: Prescaler Select

Selects the prescaler divide-by output for the BODVDD sampling mode. The input clock comes from the OSCULP32K 1KHz output.

Value	Name	Description
0x0	DIV2	Divide clock by 2
0x1	DIV4	Divide clock by 4
0x2	DIV8	Divide clock by 8
0x3	DIV16	Divide clock by 16
0x4	DIV32	Divide clock by 32
0x5	DIV64	Divide clock by 64
0x6	DIV128	Divide clock by 128

Value	Name	Description
0x7	DIV256	Divide clock by 256
0x8	DIV512	Divide clock by 512
0x9	DIV1024	Divide clock by 1024
0xA	DIV2048	Divide clock by 2048
0xB	DIV4096	Divide clock by 4096
0xC	DIV8192	Divide clock by 8192
0xD	DIV16384	Divide clock by 16384
0xE	DIV32768	Divide clock by 32768
0xF	DIV65536	Divide clock by 65536

#### Bit 8 – ACTCFG: BODVDD Configuration in Active Sleep Mode

This field is not synchronized.

Value	Description
0	In active mode, the BODVDD operates in continuous mode.
1	In active mode, the BODVDD operates in sampling mode.

#### Bit 6 – RUNSTDBY: Run in Standby

This bit is not synchronized.

Value	Description
0	In standby sleep mode, the BODVDD is disabled.
1	In standby sleep mode, the BODVDD is enabled.

#### Bit 5 – STDBYCFG: BODVDD Configuration in Standby Sleep Mode

If the RUNSTDBY bit is set to '1', the STDBYCFG bit sets the BODVDD configuration in standby sleep mode.

This field is not synchronized.

Value	Description
0	In standby sleep mode, the BODVDD is enabled and configured in continuous mode.
1	In standby sleep mode, the BODVDD is enabled and configured in sampling mode.

#### Bits 4:3 – ACTION[1:0]: BODVDD Action

These bits are used to select the BODVDD action when the supply voltage crosses below the BODVDD threshold.

These bits are loaded from NVM User Row at start-up.

This field is not synchronized.

Value	Name	Description
0x0	NONE	No action
0x1	RESET	The BODVDD generates a reset
0x2	INT	The BODVDD generates an interrupt
0x3	-	Reserved

#### Bit 2 – HYST: Hysteresis

This bit indicates whether hysteresis is enabled for the BODVDD threshold voltage:

This field is not synchronized.

Value	Description
0	No hysteresis.
1	Hysteresis enabled.

#### Bit 1 – ENABLE: Enable

This bit is loaded from NVM User Row at start-up.

This bit is not enable-protected.

Value	Description
0	BODVDD is disabled.
1	BODVDD is enabled.

### 23.8.6. VDDCORE Brown-Out Detector (BODCORE) Control

**Name:** BODCORE

**Offset:** 0x14

**Reset:** 0x00XX00YZ - determined by NVM User Row (0xXX=0bxxxxxx, 0xYZ=0x000yy0z0)

**Property:** Write-Synchronized, Enable-Protected, PAC Write-Protection

Bit	31	30	29	28	27	26	25	24	
Access									
Reset									
Bit	23	22	21	20	19	18	17	16	
Access			LEVEL[5:0]						
Reset			x	x	x	x	x	x	
Bit	15	14	13	12	11	10	9	8	
Access	PSEL[3:0]								ACTCFG
Reset	0	0	0	0				0	
Bit	7	6	5	4	3	2	1	0	
Access		RUNSTDBY	STDBYCFG	ACTION[1:0]		HYST	ENABLE		
Reset		0	0	y	y	0	z		

#### Bits 21:16 – LEVEL[5:0]: BODCORE Threshold Level

This field sets the triggering voltage threshold for the BODCORE.

These bits are loaded from NVM User Row at start-up.

This field is not synchronized.

#### Bits 15:12 – PSEL[3:0]: Prescaler Select

Selects the prescaler divide-by output for the BODCORE Sampling mode. The input clock comes from the OSCULP32K 1KHz output.

Value	Name	Description
0x0	DIV2	Divide clock by 2
0x1	DIV4	Divide clock by 4
0x2	DIV8	Divide clock by 8
0x3	DIV16	Divide clock by 16
0x4	DIV32	Divide clock by 32
0x5	DIV64	Divide clock by 64
0x6	DIV128	Divide clock by 128
0x7	DIV256	Divide clock by 256

Value	Name	Description
0x8	DIV512	Divide clock by 512
0x9	DIV1024	Divide clock by 1024
0xA	DIV2048	Divide clock by 2048
0xB	DIV4096	Divide clock by 4096
0xC	DIV8192	Divide clock by 8192
0xD	DIV16384	Divide clock by 16384
0xE	DIV32768	Divide clock by 32768
0xF	DIV65536	Divide clock by 65536

#### Bit 8 – ACTCFG: BODCORE Configuration in Active Sleep Mode

This field is not synchronized.

Value	Description
0	In active mode, the BODCORE operates in continuous mode.
1	In active mode, the BODCORE operates in sampling mode.

#### Bit 6 – RUNSTDBY: Run in Standby

This bit is not synchronized.

Value	Description
0	In standby sleep mode, the BODCORE is disabled.
1	In standby sleep mode, the BODCORE is enabled.

#### Bit 5 – STDBYCFG: BODCORE Configuration in Standby Sleep Mode

If the RUNSTDBY bit is set to 1, the STDBYCFG bit sets the BODCORE configuration in standby sleep mode.

This field is not synchronized.

Value	Description
0	In standby sleep mode, the BODCORE is enabled and configured in continuous mode.
1	In standby sleep mode, the BODCORE is enabled and configured in sampling mode.

#### Bits 4:3 – ACTION[1:0]: BODCORE Action

These bits are used to select the BODCORE action when the supply voltage crosses below the BODCORE threshold.

These bits are loaded from NVM User Row at start-up.

This field is not synchronized.

Value	Name	Description
0x0	NONE	No action.
0x1	RESET	The BODCORE generates a reset.

Value	Name	Description
0x2	INT	The BODCORE generates an interrupt.
0x3	-	Reserved

**Bit 2 – HYST: Hysteresis**

This bit indicates whether hysteresis is enabled for the BODCORE threshold voltage:

This bit is not synchronized.

Value	Description
0	No hysteresis.
1	Hysteresis enabled.

**Bit 1 – ENABLE: Enable**

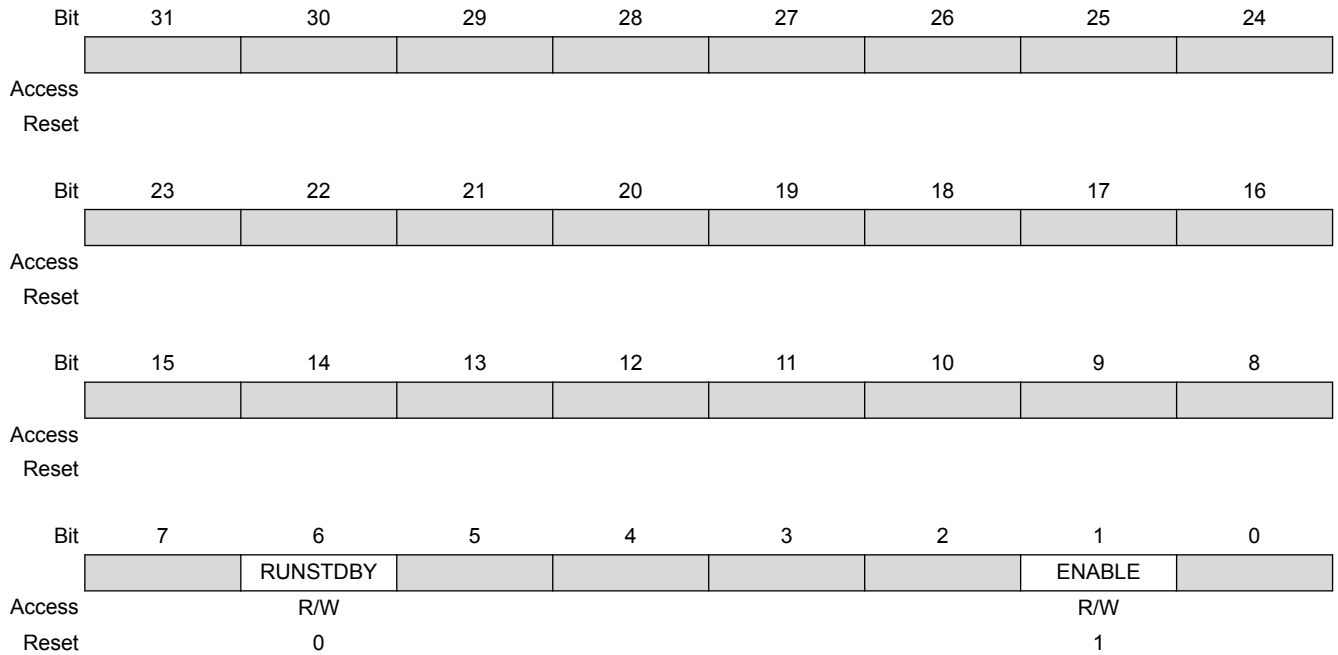
This bit is loaded from NVM User Row at start-up.

This bit is not enable-protected.

Value	Description
0	BODCORE is disabled.
1	BODCORE is enabled.

### 23.8.7. Voltage Regulator System (VREG) Control

**Name:** VREG  
**Offset:** 0x18  
**Reset:** 0x00000002  
**Property:** PAC Write-Protection



#### Bit 6 – RUNSTDBY: Run in Standby

Value	Description
0	The voltage regulator is in low power mode in Standby sleep mode.
1	The voltage regulator is in normal mode in Standby sleep mode.

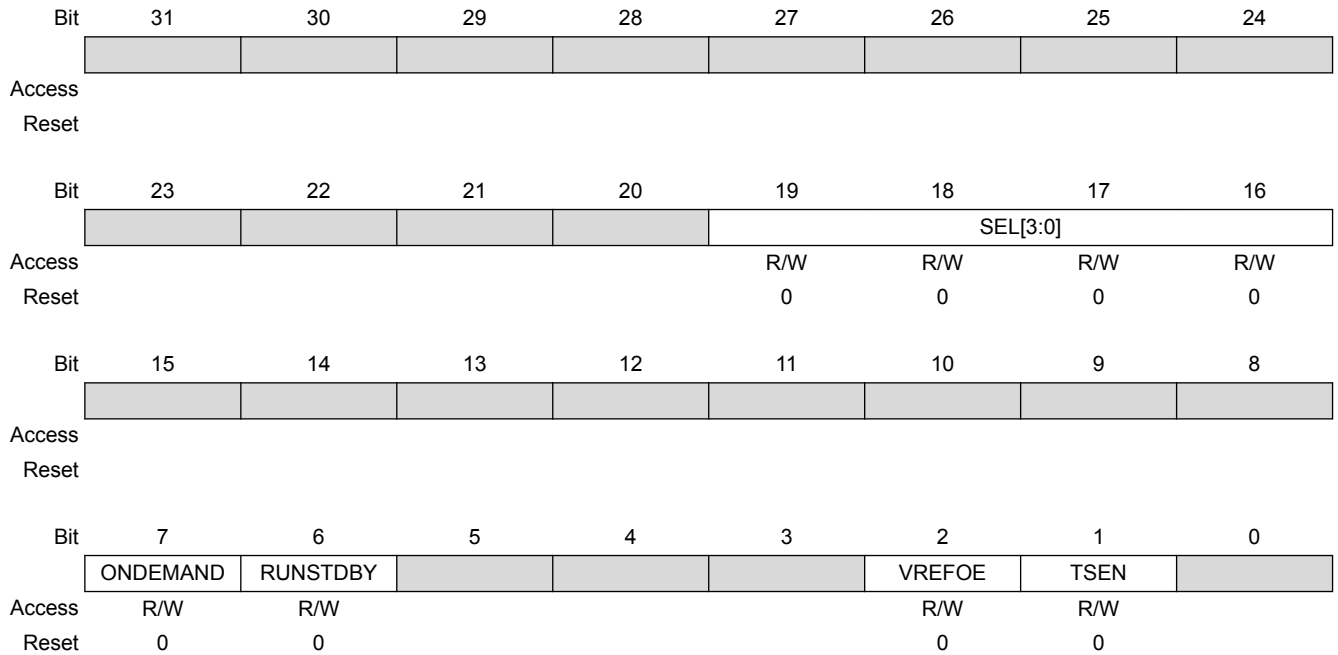
#### Bit 1 – ENABLE: Enable

Value	Description
0	The voltage regulator is disabled.
1	The voltage regulator is enabled.



### 23.8.8. Voltage References System (VREF) Control

**Name:** VREF  
**Offset:** 0x1C  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



#### Bits 19:16 – SEL[3:0]: Voltage Reference Selection

These bits select the Voltage Reference for the ADC/DAC.

Value	Name	Description
0x0	1V0	1.0V voltage reference typical value
0x1	1V1	1.1V voltage reference typical value
0x2	1V2	1.2V voltage reference typical value
0x3	1V25	1.25V voltage reference typical value
0x4	2V0	2.0V voltage reference typical value
0x5	2V2	2.2V voltage reference typical value
0x6	2V4	2.4V voltage reference typical value
0x7	2V5	2.5V voltage reference typical value
Others		Reserved

#### Bit 7 – ONDEMAND: On Demand Control

The On Demand operation mode allows the voltage reference or the temperature sensor to be enabled or disabled, depending on other peripheral requests.

In On Demand operation mode, i.e. if VREF.ONDEMAND=1, the voltage reference or the temperature sensor will be running only when requested by a peripheral. If there is no peripheral requesting the voltage reference (resp. the temperature sensor), the voltage reference (resp. the temperature sensor) will be disabled.

If On Demand is disabled, the voltage reference and the temperature sensor will always be running when enabled.

In standby sleep mode, the On Demand operation is still active if the VREF.RUNSTDBY=1. If VREF.RUNSTDBY=0, the voltage reference and the temperature sensor are disabled in standby sleep mode.

Value	Description
0	The voltage reference and the temperature sensor are always on, if enabled.
1	The voltage reference or the temperature sensor is enabled, when a peripheral is requesting it. The voltage reference or the temperature sensor is disabled if no peripheral is requesting it.

#### Bit 6 – RUNSTDBY: Run In Standby

The bit controls how the voltage reference or the temperature sensor behaves during standby sleep mode.

Value	Description
0	The voltage reference and the temperature sensor are halted during standby sleep mode.
1	The voltage reference or the temperature sensor is not stopped in standby sleep mode. If VREF.ONDEMAND=1, the voltage reference or the temperature sensor will be running when a peripheral is requesting it. If VREF.ONDEMAND=0, the voltage reference or the temperature sensor will always be running in standby sleep mode.

#### Bit 2 – VREFOE: Voltage Reference Output Enable

Value	Description
0	The Voltage Reference output is not available as an ADC input channel.
1	The Voltage Reference output is routed to an ADC input channel.

#### Bit 1 – TSEN: Temperature Sensor Enable

Value	Description
0	Temperature Sensor is disabled.
1	Temperature Sensor is enabled and routed to an ADC input channel.

## 24. WDT – Watchdog Timer

### 24.1. 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.

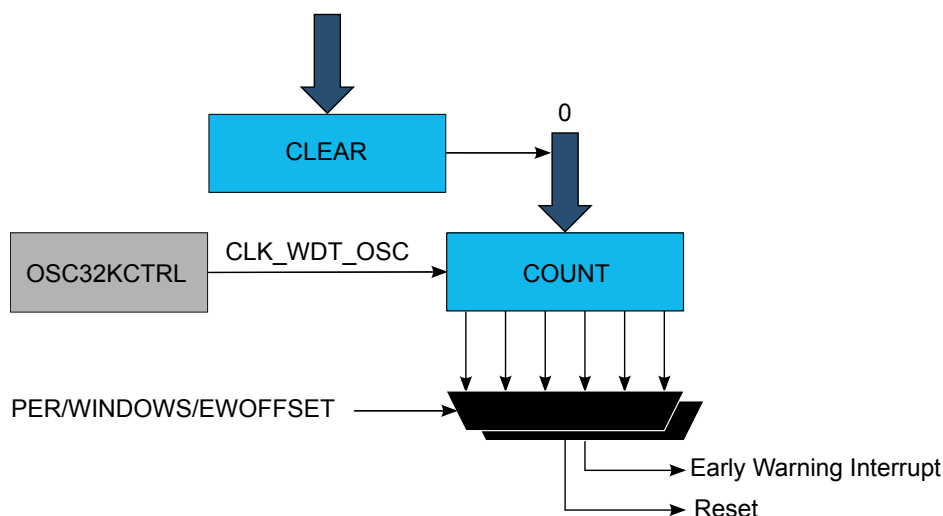
When enabled, the WDT will run in active mode and all sleep modes. It is asynchronous and runs from a CPU-independent clock source. The WDT will continue operation and issue a system reset or interrupt even if the main clocks fail.

### 24.2. 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
  - Window mode
- Selectable time-out periods
  - From 8 cycles to 16,384 cycles in Normal mode
  - From 16 cycles to 32,768 cycles in Window mode
- Always-On capability

## 24.3. Block Diagram

Figure 24-1. WDT 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 below.

### 24.5.1. I/O Lines

Not applicable.

### 24.5.2. 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](#) on page 162

### 24.5.3. Clocks

The WDT bus clock (CLK\_WDT\_APB) can be enabled and disabled (masked) in the Main Clock module (MCLK).

A 1KHz oscillator clock (CLK\_WDT\_OSC) is required to clock the WDT internal counter. This clock must be configured and enabled in the 32KHz Oscillator Controller (OSC32KCTRL) before using the WDT.

CLK\_WDT\_OSC is normally sourced from the clock of the internal ultra-low-power oscillator, OSCULP32K. Due to the ultra-low-power design, the oscillator is not very accurate, and so the exact time-out period may vary from device to device. This variation must be kept in mind when designing software that uses the WDT to ensure that the time-out periods used are valid for all devices.

The counter clock CLK\_WDT\_OSC is asynchronous to the bus clock (CLK\_WDT\_APB). Due to this asynchronicity, writing to certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) on page 269 for further details.

#### Related Links

[Peripheral Clock Masking](#) on page 135

[OSC32KCTRL – 32KHz Oscillators Controller](#) on page 211

#### 24.5.4. DMA

Not applicable.

#### 24.5.5. 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](#) on page 41

[Overview](#) on page 41

[Interrupt Line Mapping](#) on page 41

#### 24.5.6. Events

Not applicable.

#### 24.5.7. Debug Operation

When the CPU is halted in debug mode the WDT will halt normal operation.

#### 24.5.8. 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

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.

#### 24.5.9. Analog Connections

Not applicable.

## 24.6. Functional Description

### 24.6.1. 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 A register (CTRLA) and the Interrupt Enable register (handled by INTENCLR/INTENSET) determine the mode of operation:

**Table 24-1. WDT Operating Modes**

CTRLA.ENABLE	CTRLA.WEN	Interrupt Enable	Mode
0	x	x	Stopped
1	0	0	Normal mode
1	0	1	Normal mode with Early Warning interrupt
1	1	0	Window mode
1	1	1	Window mode with Early Warning interrupt

## 24.6.2. Basic Operation

### 24.6.2.1. Initialization

The following bits are enable-protected, meaning that they can only be written when the WDT is disabled (CTRLA.ENABLE=0):

- Control A register (CTRLA), except the Enable bit (CTRLA.ENABLE)
- Configuration register (CONFIG)
- Early Warning Interrupt Control register (EWCTRL)

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'.

The WDT can be configured 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 desired, the Window Enable bit in the Control A register must be set (CTRLA.WEN=1) and the Window Period bits in the Configuration register (CONFIG.WINDOW) must be defined.

Enable-protection is denoted by the "Enable-Protected" property in the register description.

### 24.6.2.2. Configurable Reset Values

After a Power-on Reset, some registers will be loaded with initial values from the NVM User Row.

This includes the following bits and bit groups:

- Enable bit in the Control A register, CTRLA.ENABLE
- Always-On bit in the Control A register, CTRLA.ALWAYSON
- Watchdog Timer Windows Mode Enable bit in the Control A register, CTRLA.WEN
- Watchdog Timer Windows Mode Time-Out Period bits in the Configuration register, CONFIG.WINDOW
- Time-Out Period bits in the Configuration register, CONFIG.PER
- Early Warning Interrupt Time Offset bits in the Early Warning Interrupt Control register, EWCTRL.EWOFFSET

#### Related Links

[NVM User Row Mapping](#) on page 36

### 24.6.2.3. Enabling, Disabling, and Resetting

The WDT is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The WDT is disabled by writing a '0' to CTRLA.ENABLE.

The WDT can be disabled only if the Always-On bit in the Control A register (CTRLA.ALWAYSON) is '0'.

#### 24.6.2.4. 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 A register (CTRLA.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.

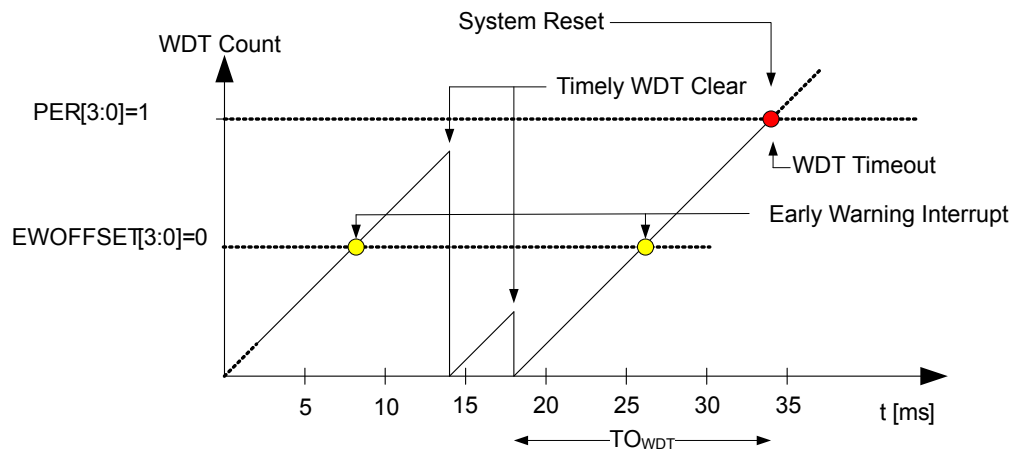
The WDT is cleared and a new WDT time-out period is started by writing 0xA5 to the Clear register (CLEAR). Writing any other value than 0xA5 to CLEAR will issue an immediate system reset.

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 24-2. Normal-Mode Operation



#### 24.6.2.5. 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.

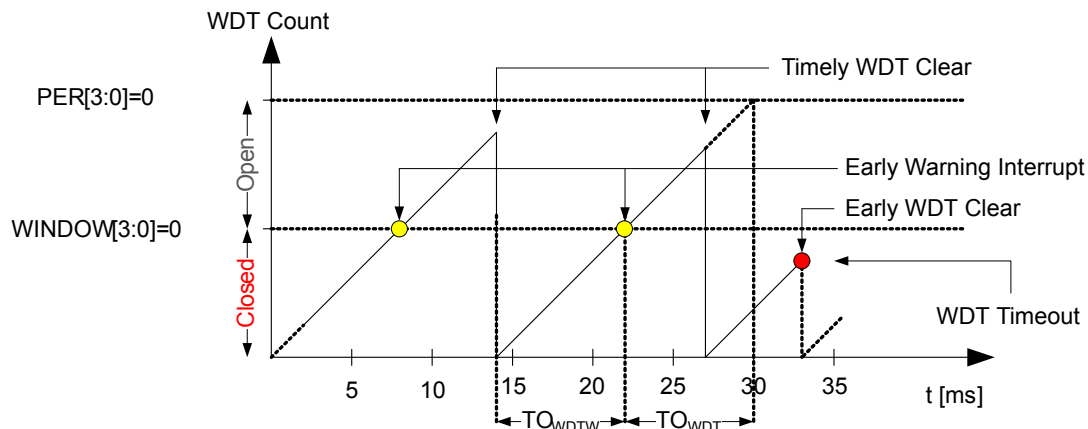
Both parameters  $TO_{WDTW}$  and  $TO_{WDT}$  are periods in a range from 8ms to 16s, so the total duration of the WDT time-out period is the sum of the two parameters.

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 24-3. Window-Mode Operation**



### 24.6.3. DMA Operation

Not applicable.

### 24.6.4. Interrupts

The WDT has the following interrupt source:

- Early Warning (EW): Indicates that the counter is approaching the time-out condition.
  - This interrupt is an asynchronous wake-up source.

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](#) on page 279 register description 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:** Interrupts must be globally enabled for interrupt requests to be generated.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

[Overview](#) on page 41

[Interrupt Line Mapping](#) on page 41

[PM – Power Manager](#) on page 162

[Sleep Mode Controller](#) on page 164

### 24.6.5. Events

Not applicable.



### 24.6.6. Sleep Mode Operation

The WDT will continue to operate in any sleep mode where the source clock is active except backup mode. The WDT interrupts can be used to wake up the device from a sleep mode. An interrupt request will be generated after the wake-up if the Interrupt Controller is configured accordingly. Otherwise the CPU will wake up directly, without triggering an interrupt. In this case, the CPU will continue executing from the instruction following the entry into sleep.

### 24.6.7. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

The following registers are synchronized when written:

- Enable bit in Control A register (CTRLA.ENABLE)
- Window Enable bit in Control A register (CTRLA.WEN)
- Always-On bit in control Control A (CTRLA.ALWAYSON)

The following registers are synchronized when read:

- Watchdog Clear register (CLEAR)

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.

### 24.6.8. Additional Features

#### 24.6.8.1. Always-On Mode

The Always-On mode is enabled by setting the Always-On bit in the Control A register (CTRLA.ALWAYSON=1). When the Always-On mode is enabled, the WDT runs continuously, regardless of the state of CTRLA.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 CTRLA.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 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 WDT Operating Modes With Always-On shows the operation of the WDT for CTRLA.ALWAYSON=1.

**Table 24-2. 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

#### 24.6.8.2. Early Warning

The Early Warning interrupt notifies that the WDT is approaching its time-out condition. 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 CLK\_WDT\_OSC clocks before the interrupt is generated, relative to 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. Consequently, 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, the Early Warning interrupt can be used to wake up and clear the Watchdog Timer, after which the system can perform other tasks or return to sleep mode.

If the WDT is operating in Normal mode with CONFIG.PER = 0x2 and EWCTRL.EWOFFSET = 0x1, the Early Warning interrupt is generated 16 CLK\_WDT\_OSC clock cycles after the start of the time-out period. The time-out system reset is generated 32 CLK\_WDT\_OSC clock cycles after the start of the watchdog time-out period.

## 24.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	ALWAYSON					WEN	ENABLE	
0x01	CONFIG	7:0	WINDOW[3:0]			PER[3:0]				
0x02	EWCTRL	7:0				EWOFFSET[3:0]				
0x03	Reserved									
0x04	INTENCLR	7:0							EW	
0x05	INTENSET	7:0							EW	
0x06	INTFLAG	7:0							EW	
0x07	Reserved									
0x08	SYNCBUSY	7:0			CLEAR	ALWAYSON	WEN	ENABLE		
0x09		15:8								
0x0A		23:16								
0x0B		31:24								
0x0C	CLEAR	7:0	CLEAR[7:0]							

## 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.

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](#) on page 265.

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](#) on page 269.

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.

## 24.8.1. Control A

**Name:** CTRLA

**Offset:** 0x00

**Reset:** N/A - Loaded from NVM User Row at startup

**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	ALWAYSON					WEN	ENABLE	
Access	R/W					R/W	R/W	
Reset	-					-	-	

### Bit 7 – ALWAYSON: Always-On

This bit allows the WDT to run continuously. After being set, this bit cannot be written to '0', and the WDT will remain enabled until a power-on Reset is received. When this bit is '1', the Control A register (CTRLA), 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 '0' to this bit has no effect.

This bit is not Enable-Protected.

This bit is loaded from NVM User Row at startup.

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

This bit enables Window mode. It can only be written if the peripheral is disabled unless CTRLA.ALWAYSON=1. The initial value of this bit is loaded from Flash Calibration.

This bit is loaded from NVM User Row at startup.

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. It can only be written if CTRLA.ALWAYSON=0.

Due to synchronization, there is delay between 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 Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

This bit is not Enable-Protected.

This bit is loaded from NVM User Row at startup.

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

## 24.8.2. Configuration

**Name:** CONFIG  
**Offset:** 0x01  
**Reset:** Loaded from NVM User Row at startup  
**Property:** PAC Write-Protection, Enable-Protected

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	-	-	-	-	-	-	-	-

### 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 cycles of the 1.024kHz CLK\_WDT\_OSC clock.

These bits are loaded from NVM User Row at startup.

Value	Name	Description
0x0	CYC8	8 clock cycles
0x1	CYC16	16 clock cycles
0x2	CYC32	32 clock cycles
0x3	CYC64	64 clock cycles
0x4	CYC128	128 clock cycles
0x5	CYC256	256 clock cycles
0x6	CYC512	512 clock cycles
0x7	CYC1024	1024 clock cycles
0x8	CYC2048	2048 clock cycles
0x9	CYC4096	4096 clock cycles
0xA	CYC8192	8192 clock cycles
0xB	CYC16384	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 1.024kHz CLK\_WDTOSC clock cycles. In Window mode operation, these bits define the open window period.

These bits are loaded from NVM User Row at startup.

Value	Name	Description
0x0	CYC8	8 clock cycles
0x1	CYC16	16 clock cycles
0x2	CYC32	32 clock cycles

Value	Name	Description
0x3	CYC64	64 clock cycles
0x4	CYC128	128 clock cycles
0x5	CYC256	256 clock cycles
0x6	CYC512	512 clock cycles
0x7	CYC1024	1024 clock cycles
0x8	CYC2048	2048 clock cycles
0x9	CYC4096	4096 clock cycles
0xA	CYC8192	8192 clock cycles
0xB	CYC16384	16384 clock cycles
0xC - 0xF	-	Reserved

### 24.8.3. Early Warning Control

**Name:** EWCTRL  
**Offset:** 0x02  
**Reset:** N/A - Loaded from NVM User Row at startup  
**Property:** PAC Write-Protection, Enable-Protected

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

#### Bits 3:0 – EWOFFSET[3:0]: Early Warning Interrupt Time Offset

These bits determine the number of GCLK\_WDT clock cycles between the start of the watchdog time-out period and the generation of the Early Warning interrupt. These bits are loaded from NVM User Row at startup.

Value	Name	Description
0x0	CYC8	8 clock cycles
0x1	CYC16	16 clock cycles
0x2	CYC32	32 clock cycles
0x3	CYC64	64 clock cycles
0x4	CYC128	128 clock cycles
0x5	CYC256	256 clock cycles
0x6	CYC512	512 clock cycles
0x7	CYC1024	1024 clock cycles
0x8	CYC2048	2048 clock cycles
0x9	CYC4096	4096 clock cycles
0xA	CYC8192	8192 clock cycles
0xB	CYC16384	16384 clock cycles
0xC - 0xF	-	Reserved



#### 24.8.4. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x04

**Reset:** 0x00

**Property:** PAC Write-Protection

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

##### Bit 0 – EW: Early Warning Interrupt Enable

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

Writing a '1' to this bit clears the Early Warning Interrupt Enable bit, which disables the Early Warning interrupt.

Value	Description
0	The Early Warning interrupt is disabled.
1	The Early Warning interrupt is enabled.

### 24.8.5. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x05

**Reset:** 0x00

**Property:** PAC Write-Protection

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

#### Bit 0 – EW: Early Warning Interrupt Enable

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

Writing a '1' to this bit sets the Early Warning Interrupt Enable bit, which enables the Early Warning interrupt.

Value	Description
0	The Early Warning interrupt is disabled.
1	The Early Warning interrupt is enabled.

## 24.8.6. Interrupt Flag Status and Clear

**Name:** INTFLAG

**Offset:** 0x06

**Reset:** 0x00

**Property:** N/A

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

### Bit 0 – EW: Early Warning

This flag is cleared by writing a '1' to it.

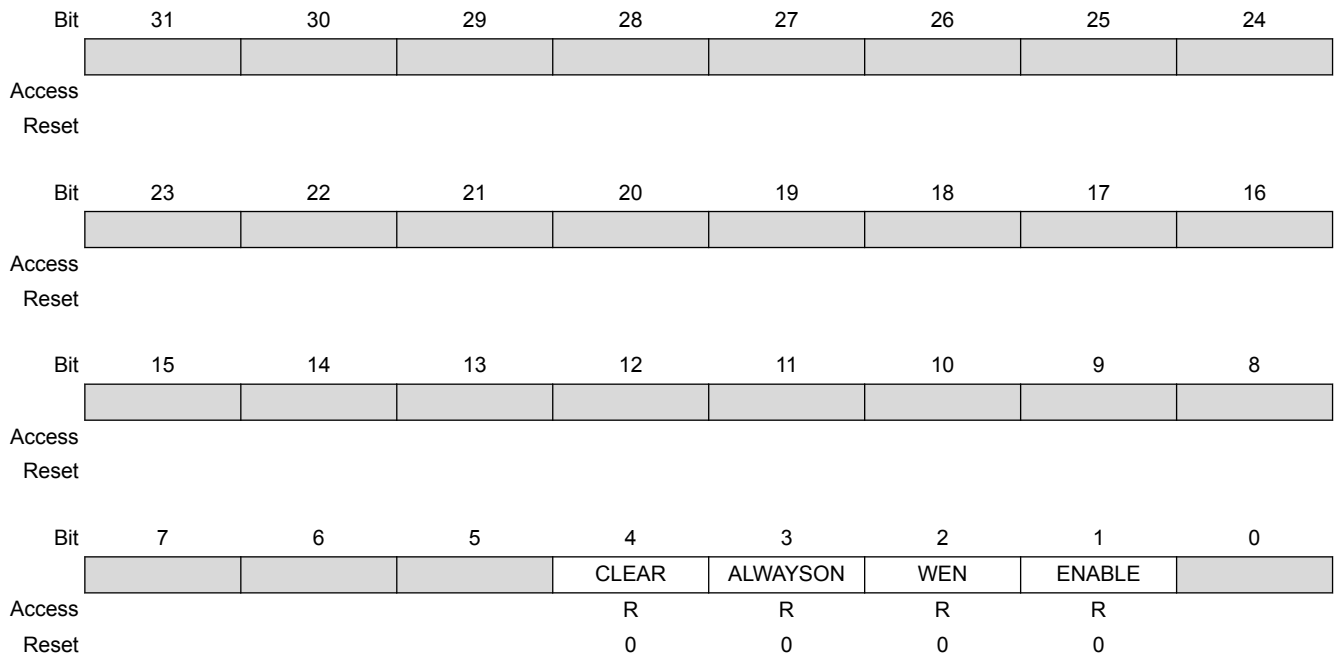
This flag is set when an Early Warning interrupt occurs, as defined by the EWOFFSET bit group in EWCTRL.

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

Writing a '1' to this bit clears the Early Warning interrupt flag.

## 24.8.7. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** Read-Only



### Bit 4 – CLEAR: Clear Synchronization Busy

Value	Description
0	Write synchronization of the CLEAR register is complete.
1	Write synchronization of the CLEAR register is ongoing.

### Bit 3 – ALWAYS ON: Always-On Synchronization Busy

Value	Description
0	Write synchronization of the CTRLA.ALWAYSON bit is complete.
1	Write synchronization of the CTRLA.ALWAYSON bit is ongoing.

### Bit 2 – WEN: Window Enable Synchronization Busy

Value	Description
0	Write synchronization of the CTRLA.WEN bit is complete.
1	Write synchronization of the CTRLA.WEN bit is ongoing.

### Bit 1 – ENABLE: Enable Synchronization Busy

Value	Description
0	Write synchronization of the CTRLA.ENABLE bit is complete.
1	Write synchronization of the CTRLA.ENABLE bit is ongoing.

## 24.8.8. Clear

**Name:** CLEAR  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	CLEAR[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 – CLEAR[7:0]: Watchdog Clear

In Normal mode, writing 0xA5 to this register during the watchdog time-out period will clear the Watchdog Timer and the watchdog time-out period is restarted.

In Window mode, any writing attempt to this register before the time-out period started (i.e., during  $TO_{WDTW}$ ) will issue an immediate system Reset. Writing 0xA5 during the time-out period  $TO_{WDT}$  will clear the Watchdog Timer and the complete time-out sequence (first  $TO_{WDTW}$  then  $TO_{WDT}$ ) is restarted.

In both modes, writing any other value than 0xA5 will issue an immediate system Reset.

## 25. RTC – Real-Time Counter

### 25.1. 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 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.

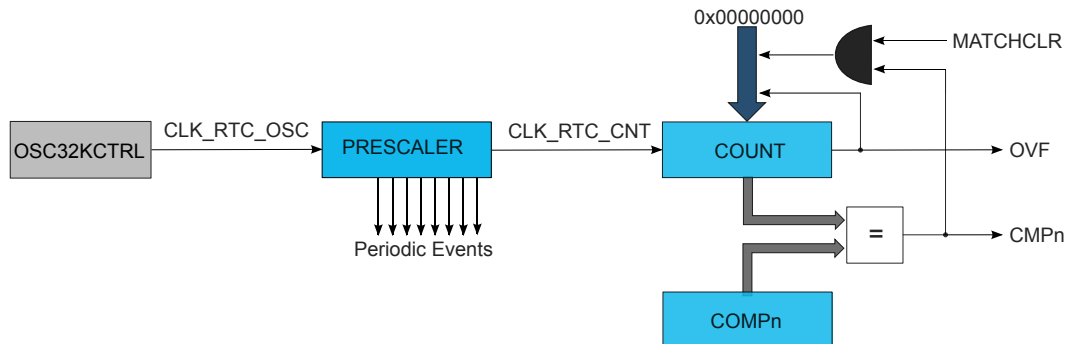
The 10-bit programmable prescaler can scale down the clock source. By this, a wide range of resolutions and time-out periods can be configured. With a 32.768kHz clock source, the minimum counter tick interval is 30.5 $\mu$ s, and time-out periods can range up to 36 hours. For a counter tick interval of 1s, the maximum time-out period is more than 136 years.

### 25.2. Features

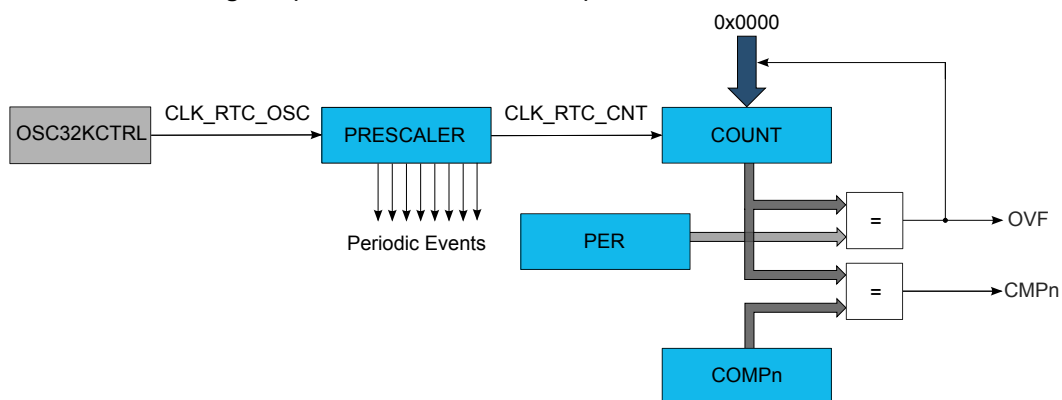
- 32-bit counter with 10-bit prescaler
- Multiple clock sources
- 32-bit or 16-bit counter mode
- Clock/Calendar mode
  - Time in seconds, minutes, and hours (12/24)
  - Date in day of month, month, and year
  - Leap year correction
- Digital prescaler correction/tuning for increased accuracy
- Overflow, alarm/compare match and prescaler interrupts and events
  - Optional clear on alarm/compare match

### 25.3. Block Diagram

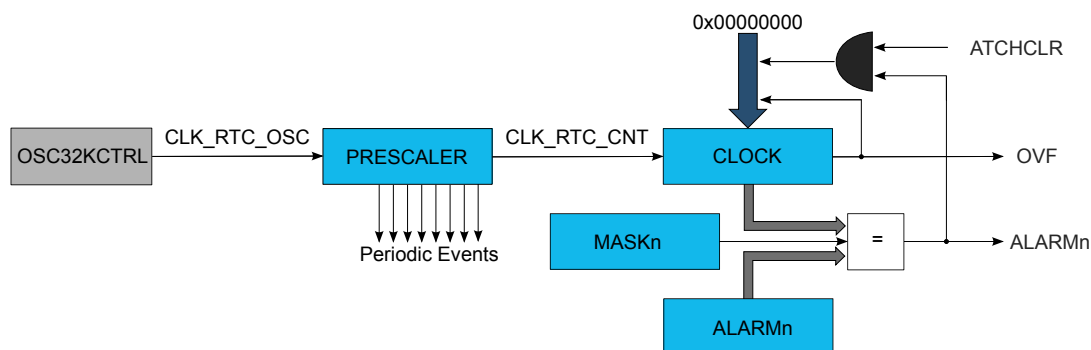
Figure 25-1. RTC Block Diagram (Mode 0 — 32-Bit Counter)



**Figure 25-2. RTC Block Diagram (Mode 1 — 16-Bit Counter)**



**Figure 25-3. RTC Block Diagram (Mode 2 — Clock/Calendar)**



## 25.4. Signal Description

Not applicable.

## 25.5. Product Dependencies

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

### 25.5.1. I/O Lines

Not applicable.

### 25.5.2. 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 A register (CTRLA.SWRST=1).

#### Related Links

[PM – Power Manager](#) on page 162

[PM – Power Manager](#) on page 162



### 25.5.3. Clocks

The RTC bus clock (CLK\_RTC\_APB) can be enabled and disabled in the Main Clock module MCLK, and the default state of CLK\_RTC\_APB can be found in Peripheral Clock Masking section.

A 32KHz or 1KHz oscillator clock (CLK\_RTC\_OSC) is required to clock the RTC. This clock must be configured and enabled in the 32KHz oscillator controller (OSC32KCTRL) before using the RTC.

This oscillator clock is asynchronous to the bus clock (CLK\_RTC\_APB). Due to this asynchronicity, writing to certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) on page 289 for further details.

#### Related Links

[OSC32KCTRL – 32KHz Oscillators Controller](#) on page 211

[Peripheral Clock Masking](#) on page 135

### 25.5.4. DMA

Not applicable.

#### Related Links

[DMAC – Direct Memory Access Controller](#) on page 334

### 25.5.5. Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the RTC interrupt requires the Interrupt Controller to be configured first.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

### 25.5.6. Events

The events are connected to the *Event System*.

#### Related Links

[EVSYS – Event System](#) on page 474

### 25.5.7. 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 [DBGCTRL](#) on page 298 for details.

### 25.5.8. Register Access Protection

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

- Interrupt Flag Status and Clear (INTFLAG) register

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

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

#### Related Links

[PAC - Peripheral Access Controller](#) on page 46

### 25.5.9. 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.

## 25.6. Functional Description

### 25.6.1. 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

### 25.6.2. Basic Operation

#### 25.6.2.1. Initialization

The following bits are enable-protected, meaning that they can only be written when the RTC is disabled (CTRLA.ENABLE=0):

- Operating Mode bits in the Control A register (CTRLA.MODE)
- Prescaler bits in the Control A register (CTRLA.PRESCALER)
- Clear on Match bit in the Control A register (CTRLA.MATCHCLR)
- Clock Representation bit in the Control A register (CTRLA.CLKREP)

The following register is enable-protected

- Event Control register (EVCTRL)

Enable-protected bits and registers can be changed only when the RTC is disabled (CTRLA.ENABLE=0). If the RTC is enabled (CTRLA.ENABLE=1), these operations are necessary: first write CTRLA.ENABLE=0 and check whether the write synchronization has finished, then change the desired bit field value. Enable-protected bits 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 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:

$$f_{\text{CLK\_RTC\_CNT}} = \frac{f_{\text{CLK\_RTC\_OSC}}}{2^{\text{PRESCALER}}}$$

The frequency of the oscillator clock, CLK\_RTC\_OSC, is given by  $f_{\text{CLK\_RTC\_OSC}}$ , and  $f_{\text{CLK\_RTC\_CNT}}$  is the frequency of the internal prescaled RTC clock, CLK\_RTC\_CNT.

#### 25.6.2.2. Enabling, Disabling, and Resetting

The RTC is enabled by setting the Enable bit in the Control A register (CTRLA.ENABLE=1). The RTC is disabled by writing CTRLA.ENABLE=0.

The RTC is reset by setting the Software Reset bit in the Control A register (CTRLA.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.

### 25.6.2.3. 32-Bit Counter (Mode 0)

When the RTC Operating Mode bits in the Control A register are zero (CTRLA.MODE=00), the counter operates in 32-bit Counter mode. The block diagram of this mode is shown in [Figure 25-1 RTC Block Diagram \(Mode 0 — 32-Bit Counter\)](#) on page 283. 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 A register (CTRLA.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 the prescaler events. Note that when CTRLA.MATCHCLR is '1', INTFLAG.CMP0 and INTFLAG.OVF will both be set simultaneously on a compare match with COMP0.

### 25.6.2.4. 16-Bit Counter (Mode 1)

When the RTC Operating Mode bits in the Control A register (CTRLA.MODE) are 1, the counter operates in 16-bit Counter mode as shown in [Figure 25-2 RTC Block Diagram \(Mode 1 — 16-Bit Counter\)](#) on page 284. 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.

### 25.6.2.5. Clock/Calendar (Mode 2)

When the RTC Operating Mode bit field in the Control A register (CTRLA.MODE) is '2', the counter operates in Clock/Calendar mode, as shown in [Figure 25-3 RTC Block Diagram \(Mode 2 — Clock/Calendar\)](#) on page 284. 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 A register (CTRLA.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 31 of year 63, and then wrap to 00:00:00 January 1 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.ALARM0) is set on the next 0-to-1 transition of CLK\_RTC\_CNT.

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 A register (CTRLA.MATCHCLR) is set, 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 it would be possible with the prescaler events only (see [Periodic Intervals](#) on page 290).

**Note:** When CTRLA.MATCHCLR is 1, INTFLAG.ALARM0 and INTFLAG.OVF will both be set simultaneously on an alarm match with ALARM0.

### 25.6.3. DMA Operation

Not applicable.

### 25.6.4. Interrupts

The RTC has the following interrupt sources:

- Overflow (OVF): Indicates that the counter has reached its top value and wrapped to zero.
- Compare (CMPn): Indicates a match between the counter value and the compare register.
- Alarm (ALARM): Indicates a match between the clock value and the alarm register.
- Period n (PERn): The corresponding bit in the prescaler has toggled. Refer to [Periodic Intervals](#) on page 290 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 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](#) on page 41

### 25.6.5. Events

The RTC can generate the following output events:

- Overflow (OVF): Generated when the counter has reached its top value and wrapped to zero.
- Compare (CMPn): Indicates a match between the counter value and the compare register.
- Alarm (ALARM): Indicates a match between the clock value and the alarm register.
- Period n (PERn): The corresponding bit in the prescaler has toggled. Refer to [Periodic Intervals](#) on page 290 for details.

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](#) on page 474

### 25.6.6. 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.

### 25.6.7. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

The following bits are synchronized when written:

- Software Reset bit in Control A register, CTRLA.SWRST
- Enable bit in Control A register, CTRLA.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

The following registers are synchronized when read:

- The Counter Value register, COUNT, if the Counter Read Sync Enable bit in CTRLA (CTRLA.COUNTSYNC) is '1'
- The Clock Value register, CLOCK, if the Clock Read Sync Enable bit in CTRLA (CTRLA.CLOCKSYNC) is '1'

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](#) on page 110

### 25.6.8. Additional Features

#### 25.6.8.1. Periodic Intervals

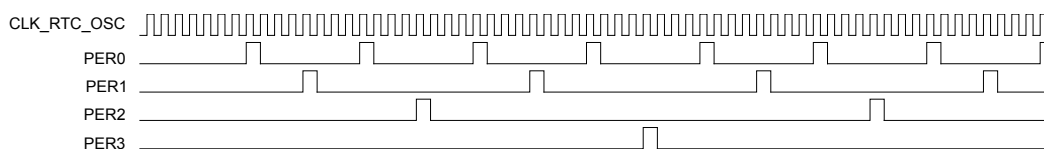
The RTC prescaler can generate interrupts and events at periodic intervals, allowing flexible system tick creation. Any of the upper eight bits of the prescaler (bits 2 to 9) can be the source of an interrupt/event. When one of the eight Periodic Event Output bits in the Event Control register (EVCTRL.PEREO[n=0..7]) 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:

$$f_{\text{PERIODIC}(n)} = \frac{f_{\text{CLK\_RTC\_OSC}}}{2^{n+3}}$$

$f_{\text{CLK\_RTC\_OSC}}$  is the frequency of the internal prescaler clock CLK\_RTC\_OSC, and n is the position of the EVCTRL.PEREO[n] 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 CTRLA.PRESCALER is zero. Then, no periodic events will be generated.

**Figure 25-4. Example Periodic Events**



#### 25.6.8.2. 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 CTRLA.PRESCALER is greater than 1.

The digital correction circuit adds or subtracts cycles from the RTC prescaler to adjust the frequency in approximately 1ppm steps. Digital correction is achieved by adding or skipping a single count in the prescaler once every 4096 CLK\_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:

$$\text{Correction in ppm} = \frac{\text{FREQCORR.VALUE}}{4096 \cdot 240} \cdot 10^6 \text{ ppm}$$

This results in a resolution of 1.017ppm.

The Sign bit in the Frequency Correction register (FREQCORR.SIGN) determines the direction of the correction. A positive value will speed up the frequency, and a negative value will slow down the frequency.

Digital correction also affects the generation of the periodic events from the prescaler. When the correction is applied at the end of the correction cycle period, the interval between the previous periodic event and the next occurrence may also be shortened or lengthened depending on the correction value.

## 25.7. Register Summary - COUNT32

Table 25-1. Register Summary - Mode 0 Registers

Offset	Name	Bit pos.								
0x00	CTRLA on page 292	7:0	MATCHCLR					MODE[1:0]	ENABLE	SWRST
0x01		15:8	COUNTSYNC					PRESCALER[3:0]		
0x02	Reserved									
0x03	Reserved									
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
0x05		15:8	OVFEO							CMPEO0
0x06		23:16								
0x07		31:24								
0x08	INTENCLR	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x09		15:8	OVF							COMP0
0x0A	INTENSET	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x0B		15:8	OVF							COMP0
0x0C	INTFLAG	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x0D		15:8	OVF							COMP0
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	Reserved									
0x10	SYNCBUSY	7:0			COMP0		COUNT	FREQCORR	ENABLE	SWRST
0x11		15:8	COUNTSYNC							
0x12		23:16								
0x13		31:24					GP3	GP2	GP1	GP0
0x14	FREQCORR	7:0	SIGN				VALUE[6:0]			
0x15	Reserved									
0x16	Reserved									
0x17	Reserved									
0x18-0x1B	COUNT	31:0	COUNT[31:0]							
0x1C-0x1F	Reserved									
0x20-0x23	COMP0	31:0	COMP[31:0]							

## 25.8. Register Description - COUNT32

This Register Description section is valid if the RTC is in COUNT32 mode (CTRLA.MODE=0).

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.



## 25.8.1. Control A in COUNT32 mode (CTRLA.MODE=0)

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x0000

**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	COUNTSYNC				PRESCALER[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
	MATCHCLR				MODE[1:0]		ENABLE	SWRST
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

### Bit 15 – COUNTSYNC: COUNT Read Synchronization Enable

The COUNT register requires synchronization when reading. Disabling the synchronization will prevent reading valid values from the COUNT register.

This bit is not enable-protected.

Value	Description
0	COUNT read synchronization is disabled
1	COUNT read synchronization is enabled

### 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). Periodic events and interrupts are not available when the prescaler is off. These bits are not synchronized.

Value	Name	Description
0x0	OFF	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x2	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x3	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x4	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x5	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x6	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x7	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x8	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x9	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0xA	DIV512	CLK_RTC_CNT = GCLK_RTC/512



Value	Name	Description
0xB	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xC-0xF	-	Reserved

#### Bit 7 – MATCHCLR: Clear on Match

This bit defines if the counter is cleared or not on a match.

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

This bit group defines the operating mode of the RTC.

This bit is not synchronized.

Value	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 a delay between writing CTRLA.ENABLE and until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable 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
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 RTC (except DBGCTRL) to their initial state, and the RTC 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 between writing CTRLA.SWRST and until the reset is complete. CTRLA.SWRST will be cleared when the reset is complete.

Value	Description
0	There is not reset operation ongoing
1	The reset operation is ongoing

## 25.8.2. Event Control in COUNT32 mode (CTRLA.MODE=0)

**Name:** EVCTRL  
**Offset:** 0x04  
**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
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
Access	R/W							R/W
Reset	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

### Bit 15 – OVFE0: 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 7:0 – PEREOn: Periodic Interval n Event Output Enable [n = 7..0]

Value	Description
0	Periodic Interval n event is disabled and will not be generated.
1	Periodic Interval n event is enabled and will be generated.

### 25.8.3. Interrupt Enable Clear in COUNT32 mode (CTRLA.MODE=0)

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

**Name:** INTENCLR

**Offset:** 0x08

**Reset:** 0x0000

**Property:** PAC Write-Protection

Bit	15	14	13	12	11	10	9	8
	OVF							CMP0
Access	R/W							R/W
Reset	0							0

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – 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.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

#### Bit 8 – CMP0: Compare 0 Interrupt Enable

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

Writing a '1' to this bit will clear the Compare 0 Interrupt Enable bit, which disables the Compare interrupt.

Value	Description
0	The Compare 0 interrupt is disabled.
1	The Compare 0 interrupt is enabled.

#### Bits 7:0 – PERn: Periodic Interval n Interrupt Enable [n = 7..0]

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

Writing a '1' to this bit will clear the Periodic Interval n Interrupt Enable bit, which disables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

#### 25.8.4. Interrupt Enable Set - COUNT32 (CTRLA.MODE=0)

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

**Name:** INTENSET

**Offset:** 0x0A

**Reset:** 0x0000

**Property:** PAC Write-Protection

Bit	15	14	13	12	11	10	9	8
	OVF							CMP0
Access	R/W							R/W
Reset	0							0

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – 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.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

##### Bit 8 – CMP0: Compare 0 Interrupt Enable

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

Writing a '1' to this bit will set the Compare 0 Interrupt Enable bit, which enables the Compare 0 interrupt.

Value	Description
0	The Compare 0 interrupt is disabled.
1	The Compare 0 interrupt is enabled.

##### Bits 7:0 – PERn: Periodic Interval n Interrupt Enable [n = 7..0]

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

Writing a '1' to this bit will set the Periodic Interval n Interrupt Enable bit, which enables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

### 25.8.5. Interrupt Flag Status and Clear in COUNT32 mode (CTRLA.MODE=0)

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

**Name:** INTFLAG

**Offset:** 0x0C

**Reset:** 0x0000

**Property:** -

Bit	15	14	13	12	11	10	9	8
	OVF							CMP0
Access	R/W							R/W
Reset	0							0

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – OVF: Overflow

This flag is cleared by writing a '1' 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 '1'.

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

Writing a '1' to this bit clears the Overflow interrupt flag.

#### Bit 8 – CMP0: Compare 0

This flag is cleared by writing a '1' 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.COMP0 is one.

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

Writing a '1' to this bit clears the Compare 0 interrupt flag.

#### Bits 7:0 – PERn: Periodic Interval n [n = 7..0]

This flag is cleared by writing a '1' to the flag.

This flag is set on the 0-to-1 transition of prescaler bit [n+2], and an interrupt request will be generated if INTENCLR/SET.PERx is one.

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

Writing a '1' to this bit clears the Periodic Interval n interrupt flag.

## 25.8.6. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x0E  
**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

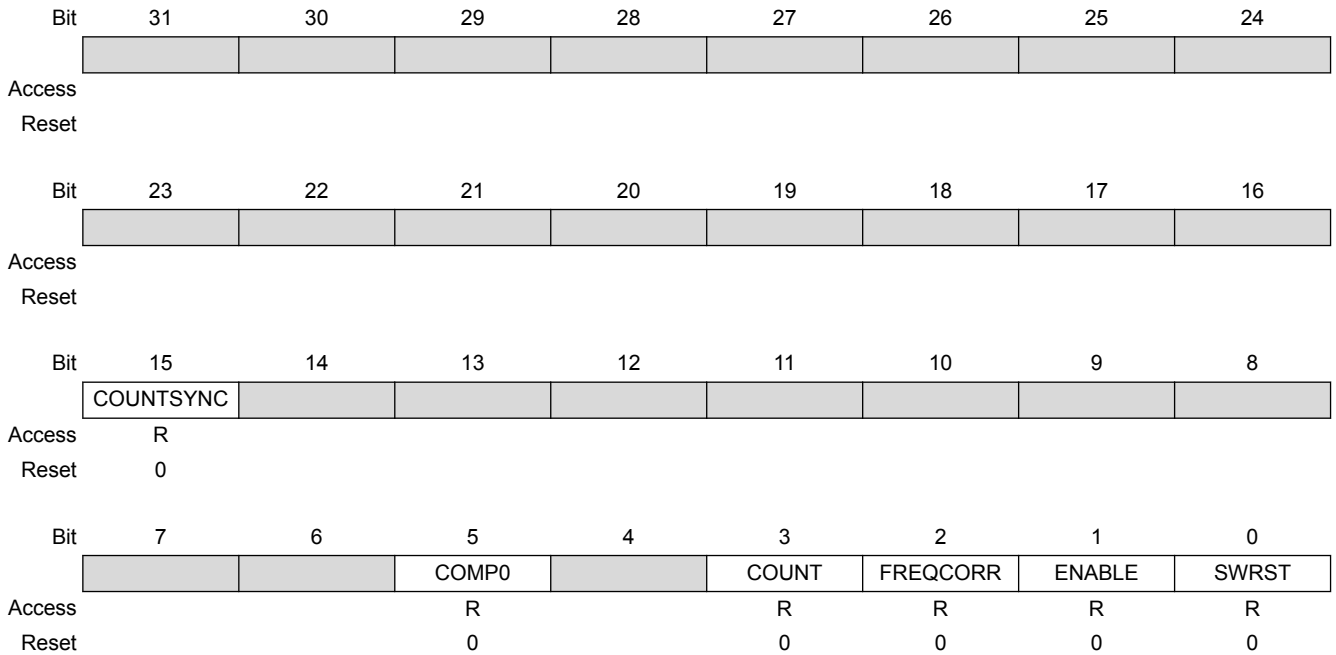
This bit is not reset by a software reset.

This bit controls the functionality when the CPU is halted by an external debugger.

Value	Description
0	The RTC is halted when the CPU is halted by an external debugger.
1	The RTC continues normal operation when the CPU is halted by an external debugger.

### 25.8.7. Synchronization Busy in COUNT32 mode (CTRLA.MODE=0)

**Name:** SYNCBUSY  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** -



#### Bit 15 – COUNTSYNC: Count Read Sync Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.COUNTSYNC bit is complete.
1	Write synchronization for CTRLA.COUNTSYNC bit is ongoing.

#### Bit 5 – COMP0: Compare 0 Synchronization Busy Status

Value	Description
0	Write synchronization for COMP0 register is complete.
1	Write synchronization for COMP0 register is ongoing.

#### Bit 3 – COUNT: Count Value Synchronization Busy Status

Value	Description
0	Read/write synchronization for COUNT register is complete.
1	Read/write synchronization for COUNT register is ongoing.

#### Bit 2 – FREQCORR: Frequency Correction Synchronization Busy Status

Value	Description
0	Read/write synchronization for FREQCORR register is complete.
1	Read/write synchronization for FREQCORR register is ongoing.

**Bit 1 – ENABLE: Enable Synchronization Busy Status**

Value	Description
0	Read/write synchronization for CTRLA.ENABLE bit is complete.
1	Read/write synchronization for CTRLA.ENABLE bit is ongoing.

**Bit 0 – SWRST: Software Reset Synchronization Busy Status**

Value	Description
0	Read/write synchronization for CTRLA.SWRST bit is complete.
1	Read/write synchronization for CTRLA.SWRST bit is ongoing.



## 25.8.8. Frequency Correlation

**Name:** FREQCORR

**Offset:** 0x14

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	SIGN		VALUE[5:0]					
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 – SIGN: Correction Sign

Value	Description
0	The correction value is positive, i.e., frequency will be increased.
1	The correction value is negative, i.e., frequency will be decreased.

### Bits 5:0 – VALUE[5:0]: Correction Value

These bits define the amount of correction applied to the RTC prescaler.

Value	Description
0	Correction is disabled and the RTC frequency is unchanged.
1 - 127	The RTC frequency is adjusted according to the value.

### 25.8.9. Counter Value in COUNT32 mode (CTRLA.MODE=0)

**Name:** COUNT

**Offset:** 0x18

**Reset:** 0x00000000

**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 define the value of the 32-bit RTC counter in mode 0.

## 25.8.10. Compare n Value in COUNT32 mode (CTRLA.MODE=0)

**Name:** COMPn  
**Offset:** 0x20 + n\*0x4 [n=0..3]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, 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 CTRLA.MATCHCLR is one.

## 25.9. Register Summary - COUNT16

Table 25-2. Register Summary - Mode

Offset	Name	Bit pos.								
0x00	CTRLA	7:0						MODE[1:0]	ENABLE	SWRST
0x01		15:8	COUNTSYNC					PRESCALER[3:0]		
0x02	Reserved									
0x03	Reserved									
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
0x05		15:8	OVFEO						CMPEO1	CMPEO0
0x06		23:16								
0x07		31:24								
0x08	INTENCLR	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x09		15:8	OVF						CMP1	CMP0

Offset	Name	Bit pos.								
0x0A	INTENSET	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x0B		15:8	OVF						CMP1	CMP0
0x0C	INTFLAG	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x0D		15:8	OVF						CMP1	CMP0
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	Reserved									
0x10	SYNDBUSY	7:0		COMP1	COMP0	PRE	COUNT	FREQCORR	ENABLE	SWRST
0x11		15:8	COUNTSYNC							
0x12		23:16								
0x13		31:24								
0x14	FREQCORR	7:0	SIGN	VALUE[6:0]						
0x15	Reserved									
0x16	Reserved									
0x17	Reserved									
0x18-0x19	COUNT	15:0	COUNT[15:0]							
0x1A-0x1B	Reserved									
0x1C-0x1D	PER	15:0	PER[15:0]							
0x1E-0x1F	Reserved									
0x20-0x21	COMP0	15:0	COMP[15:0]							
0x22-0x23	COMP1	15:0	COMP[15:0]							

## 25.10. Register Description - COUNT16

This Register Description section is valid if the RTC is in COUNT16 mode (CTRLA.MODE=1).

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.

### 25.10.1. Control A in COUNT16 mode (CTRLA.MODE=1)

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x0000

**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	COUNTSYNC					PRESCALER[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
					MODE[1:0]		ENABLE	SWRST
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### Bit 15 – COUNTSYNC: COUNT Read Synchronization Enable

The COUNT register requires synchronization when reading. Disabling the synchronization will prevent reading valid values from the COUNT register.

This bit is not enable-protected.

Value	Description
0	COUNT read synchronization is disabled
1	COUNT read synchronization is enabled

#### 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). Periodic events and interrupts are not available when the prescaler is off. These bits are not synchronized.

Value	Name	Description
0x0	OFF	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x2	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x3	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x4	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x5	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x6	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x7	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x8	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x9	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0xA	DIV512	CLK_RTC_CNT = GCLK_RTC/512

Value	Name	Description
0xB	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xC-0xF	-	Reserved

### Bits 3:2 – MODE[1:0]: Operating Mode

This field defines the operating mode of the RTC. This bit is not synchronized.

Value	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 CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable 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
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 RTC (except DBGCTRL) to their initial state, and the RTC 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 will be cleared when the reset is complete.

Value	Description
0	There is not reset operation ongoing
1	The reset operation is ongoing

## 25.10.2. Event Control in COUNT16 mode (CTRLA.MODE=1)

**Name:** EVCTRL  
**Offset:** 0x04  
**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
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
Access	R/W						R/W	R/W
Reset	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

### Bit 15 – OVFE0: 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 – CMPEOn: Compare n Event Output Enable [n = 1..0]

Value	Description
0	Compare n event is disabled and will not be generated.
1	Compare n event is enabled and will be generated for every compare match.

### Bits 7:0 – PEREOn: Periodic Interval n Event Output Enable [n = 7..0]

Value	Description
0	Periodic Interval n event is disabled and will not be generated. [n = 7..0]
1	Periodic Interval n event is enabled and will be generated. [n = 7..0]

### 25.10.3. Interrupt Enable Clear in COUNT16 mode (CTRLA.MODE=1)

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

**Name:** INTENCLR

**Offset:** 0x08

**Reset:** 0x0000

**Property:** PAC Write-Protection

Bit	15	14	13	12	11	10	9	8
	OVF						CMP1	CMP0
Access	R/W						R/W	R/W
Reset	0						0	0

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – 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.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

#### Bits 9:8 – CMPn: Compare n Interrupt Enable [n = 1..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Compare n Interrupt Enable bit, which disables the Compare n interrupt.

Value	Description
0	The Compare n interrupt is disabled.
1	The Compare n interrupt is enabled.

#### Bits 7:0 – PERn: Periodic Interval n Interrupt Enable [n = 7..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Periodic Interval n Interrupt Enable bit, which disables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.



#### 25.10.4. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x0A

**Reset:** 0x0000

**Property:** PAC Write-Protection

Bit	15	14	13	12	11	10	9	8
	OVF						CMP1	CMP0
Access	R/W						R/W	R/W
Reset	0						0	0
Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – 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.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

##### Bits 9:8 – CMPn: Compare n Interrupt Enable [n = 1..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Compare n Interrupt Enable bit, which and enables the Compare n interrupt.

Value	Description
0	The Compare n interrupt is disabled.
1	The Compare n interrupt is enabled.

##### Bits 7:0 – PERn: Periodic Interval n Interrupt Enable [n = 7..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Periodic Interval n Interrupt Enable bit, which enables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

### 25.10.5. Interrupt Flag Status and Clear in COUNT16 mode (CTRLA.MODE=1)

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.

**Name:** INTFLAG

**Offset:** 0x0C

**Reset:** 0x0000

**Property:** -

Bit	15	14	13	12	11	10	9	8
	OVF						CMP1	CMP0
Access	R/W						R/W	R/W
Reset	0						0	0

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – OVF: Overflow

This flag is cleared by writing a '1' 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 '1'.

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

Writing a '1' to this bit clears the Overflow interrupt flag.

#### Bits 9:8 – CMPn: Compare n [n = 1..0]

This flag is cleared by writing a '1' 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.COMPx is one.

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

Writing a '1' to this bit clears the Compare n interrupt flag.

#### Bits 7:0 – PERn: Periodic Interval n [n = 7..0]

This flag is cleared by writing a '1' to the flag.

This flag is set on the 0-to-1 transition of prescaler bit [n+2], and an interrupt request will be generated if INTENCLR/SET.PERx is one.

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

Writing a '1' to this bit clears the Periodic Interval n interrupt flag.

## 25.10.6. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x0E  
**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

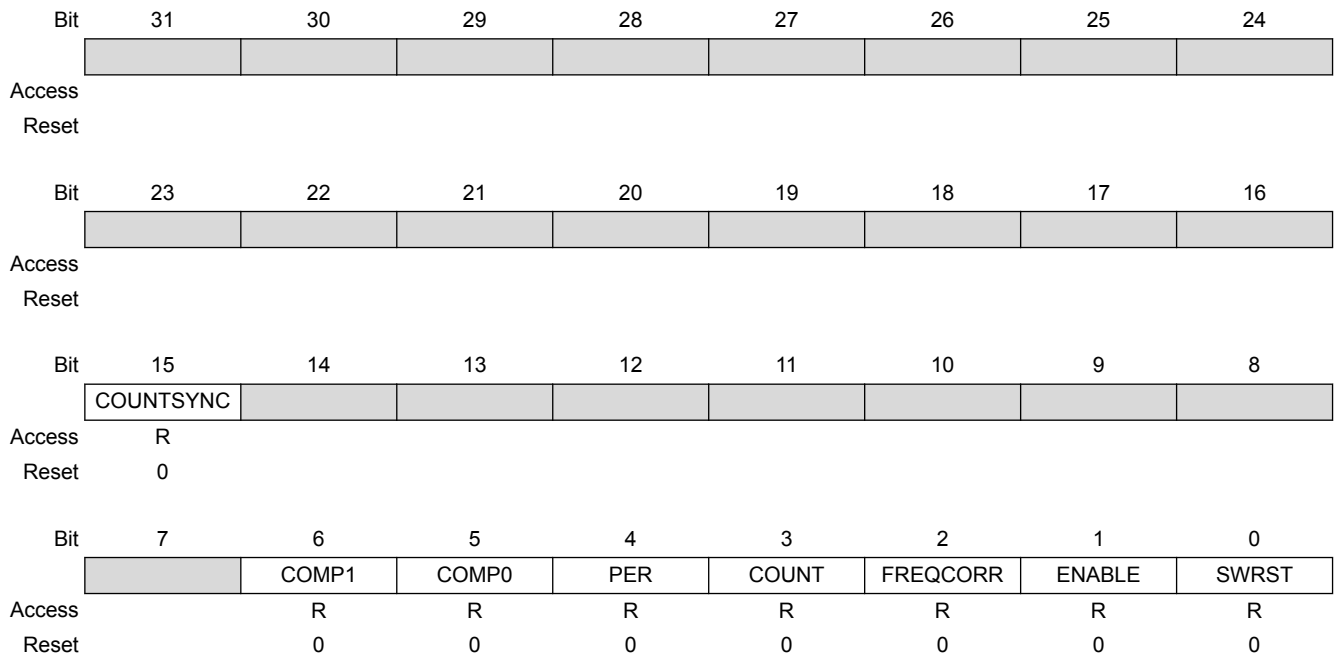
This bit is not reset by a software reset.

This bit controls the functionality when the CPU is halted by an external debugger.

Value	Description
0	The RTC is halted when the CPU is halted by an external debugger.
1	The RTC continues normal operation when the CPU is halted by an external debugger.

### 25.10.7. Synchronization Busy in COUNT16 mode (CTRLA.MODE=1)

**Name:** SYNCBUSY  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** -



#### Bit 15 – COUNTSYNC: Count Read Sync Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.COUNTSYNC bit is complete.
1	Write synchronization for CTRLA.COUNTSYNC bit is ongoing.

#### Bits 6:5 – COMPn: Compare n Synchronization Busy Status [n = 1..0]

Value	Description
0	Write synchronization for COMPn register is complete.
1	Write synchronization for COMPn register is ongoing.

#### Bit 4 – PER: Period Synchronization Busy Status

Value	Description
0	Write synchronization for PER register is complete.
1	Write synchronization for PER register is ongoing.

#### Bit 3 – COUNT: Count Value Synchronization Busy Status

Value	Description
0	Read/write synchronization for COUNT register is complete.
1	Read/write synchronization for COUNT register is ongoing.

**Bit 2 – FREQCORR: Frequency Correction Synchronization Busy Status**

Value	Description
0	Read/write synchronization for FREQCORR register is complete.
1	Read/write synchronization for FREQCORR register is ongoing.

**Bit 1 – ENABLE: Enable Synchronization Busy Status**

Value	Description
0	Read/write synchronization for CTRLA.ENABLE bit is complete.
1	Read/write synchronization for CTRLA.ENABLE bit is ongoing.

**Bit 0 – SWRST: Software Reset Synchronization Busy Status**

Value	Description
0	Read/write synchronization for CTRLA.SWRST bit is complete.
1	Read/write synchronization for CTRLA.SWRST bit is ongoing.

## 25.10.8. Frequency Correlation

**Name:** FREQCORR

**Offset:** 0x14

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	SIGN		VALUE[5:0]					
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 – SIGN: Correction Sign

Value	Description
0	The correction value is positive, i.e., frequency will be increased.
1	The correction value is negative, i.e., frequency will be decreased.

### Bits 5:0 – VALUE[5:0]: Correction Value

These bits define the amount of correction applied to the RTC prescaler.

Value	Description
0	Correction is disabled and the RTC frequency is unchanged.
1 - 127	The RTC frequency is adjusted according to the value.

### 25.10.9. Counter Value in COUNT16 mode (CTRLA.MODE=1)

**Name:** COUNT

**Offset:** 0x18

**Reset:** 0x0000

**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 define the value of the 16-bit RTC counter in COUNT16 mode (CTRLA.MODE=1).

### 25.10.10. Counter Period in COUNT16 mode (CTRLA.MODE=1)

**Name:** PER  
**Offset:** 0x18  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, 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 in COUNT16 mode (CTRLA.MODE=1).



### 25.10.11. Compare n Value in COUNT16 mode (CTRLA.MODE=1)

**Name:** COMPn  
**Offset:** 0x20 + n\*0x2 [n=0..5]  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, 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.

## 25.11. Register Summary - CLOCK

Table 25-3. Register Summary - Mode 2 Registers

Offset	Name	Bit pos.								
0x00	CTRLA	7:0	MATCHCLR	CLKREP			MODE[1:0]		ENABLE	SWRST
0x01		15:8	CLOCKSYNC				PRESCALER[3:0]			
0x02	Reserved									
0x03	Reserved									
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
0x05		15:8	OVFEO						ALARMEO0	
0x06		23:16								
0x07		31:24								
0x08	INTENCLR		PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x09		7:0	OVF						ALARM0	
0x0A	INTENSET		PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x0B		7:0	OVF						ALARM0	
0x0C	INTFLAG		PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
0x0D		7:0	OVF						ALARM0	
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	Reserved									
0x10	SYNDBUSY	7:0			ALARM0		CLOCK	FREQCORR	ENABLE	SWRST
0x11		15:8	CLOCKSYNC				MASK0			
0x12		23:16								
0x13		31:24								

Offset	Name	Bit pos.								
0x14	FREQCORR	7:0	SIGN	VALUE[6:0]						
0x15	Reserved									
0x16	Reserved									
0x17	Reserved									
0x18	CLOCK	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-0x1F	Reserved									
0x20	ALARM0	7:0	MINUTE[1:0]	SECOND[5:0]						
0x21		15:8	HOUR[3:0]			MINUTE[5:2]				
0x22		23:16	MONTH[1:0]	DAY[4:0]				HOUR[4]		
0x23		31:24	YEAR[5:0]					MONTH[3:2]		
0x24	MASK0	7:0					SEL[2:0]			
0x25	Reserved									
0x26	Reserved									
0x27	Reserved									

## 25.12. Register Description - CLOCK

This Register Description section is valid if the RTC is in Clock/Calendar mode (CTRLA.MODE=2).

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.

## 25.12.1. Control A in Clock/Calendar mode (CTRLA.MODE=2)

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x0000

**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	CLOCKSYNC				PRESCALER[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
	MATCHCLR	CLKREP			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

### Bit 15 – CLOCKSINC: CLOCK Read Synchronization Enable

The CLOCK register requires synchronization when reading. Disabling the synchronization will prevent reading valid values from the CLOCK register.

This bit is not enable-protected.

Value	Description
0	CLOCK read synchronization is disabled
1	CLOCK read synchronization is enabled

### 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). Periodic events and interrupts are not available when the prescaler is off. These bits are not synchronized.

Value	Name	Description
0x0	OFF	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x2	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x3	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x4	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x5	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x6	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x7	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x8	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x9	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0xA	DIV512	CLK_RTC_CNT = GCLK_RTC/512

Value	Name	Description
0xB	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xC-0xF	-	Reserved

#### Bit 7 – MATCHCLR: Clear on Match

This bit is valid only in Mode 0 (COUNT32) and Mode 2 (CLOCK). 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

This field defines the operating mode of the RTC. This bit is not synchronized.

Value	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 CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable 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
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 RTC, except DBGCTRL, to their initial state, and the RTC 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 will be cleared when the reset is complete.

Value	Description
0	There is not reset operation ongoing
1	The reset operation is ongoing

## 25.12.2. Event Control in Clock/Calendar mode (CTRLA.MODE=2)

**Name:** EVCTRL  
**Offset:** 0x04  
**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
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
Access	R/W							R/W
Reset	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

### Bit 15 – OVFE0: 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 – ALARMO0: 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 compare match.

### Bits 7:0 – PEREO<sub>n</sub>: Periodic Interval n Event Output Enable [n = 7..0]

Value	Description
0	Periodic Interval n event is disabled and will not be generated.
1	Periodic Interval n event is enabled and will be generated.

### 25.12.3. Interrupt Enable Clear in Clock/Calendar mode (CTRLA.MODE=2)

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

**Name:** INTENCLR  
**Offset:** 0x08  
**Reset:** 0x0000  
**Property:** PAC Write-Protection

Bit	15	14	13	12	11	10	9	8
	OVF							ALARM0
Access	R/W							R/W
Reset	0							0
Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – 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.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

#### Bit 8 – ALARM0: Alarm 0 Interrupt Enable

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Alarm 0 Interrupt Enable bit, which disables the Alarm interrupt.

Value	Description
0	The Alarm 0 interrupt is disabled.
1	The Alarm 0 interrupt is enabled.

#### Bits 7:0 – PERn: Periodic Interval n Interrupt Enable [n = 7..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Periodic Interval n Interrupt Enable bit, which disables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

#### 25.12.4. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x0A

**Reset:** 0x0000

**Property:** PAC Write-Protection

Bit	15	14	13	12	11	10	9	8
	OVF							ALARM0
Access	R/W							R/W
Reset	0							0

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – 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.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

##### Bit 8 – ALARM0: Alarm 0 Interrupt Enable

Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Alarm 0 Interrupt Enable bit, which enables the Alarm 0 interrupt.

Value	Description
0	The Alarm 0 interrupt is disabled.
1	The Alarm 0 interrupt is enabled.

##### Bits 7:0 – PERn: Periodic Interval n Interrupt Enable [n = 7..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Periodic Interval n Interrupt Enable bit, which enables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.



### 25.12.5. Interrupt Flag Status and Clear in Clock/Calendar mode (CTRLA.MODE=2)

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.

**Name:** INTFLAG

**Offset:** 0x0C

**Reset:** 0x0000

**Property:** -

Bit	15	14	13	12	11	10	9	8
	OVF							ALARM0
Access	R/W							R/W
Reset	0							0

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/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 – OVF: Overflow

This flag is cleared by writing a '1' 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 '1'.

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

Writing a '1' to this bit clears the Overflow interrupt flag.

#### Bit 8 – ALARM0: Alarm 0

This flag is cleared by writing a '1' 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.ALARM0 is one.

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

Writing a '1' to this bit clears the Alarm 0 interrupt flag.

#### Bits 7:0 – PERn: Periodic Interval n [n = 7..0]

This flag is cleared by writing a '1' to the flag.

This flag is set on the 0-to-1 transition of prescaler bit [n+2], and an interrupt request will be generated if INTENCLR/SET.PERx is '1'.

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

Writing a '1' to this bit clears the Periodic Interval n interrupt flag.

## 25.12.6. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x0E  
**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 RTC is halted when the CPU is halted by an external debugger.
1	The RTC continues normal operation when the CPU is halted by an external debugger.

## 25.12.7. Synchronization Busy in Clock/Calendar mode (CTRLA.MODE=2)

**Name:** SYNCBUSY  
**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								
Reset								
Bit	15	14	13	12	11	10	9	8
Access	R				R			
Reset	0				0			
Bit	7	6	5	4	3	2	1	0
Access			R		R	R	R	R
Reset			0		0	0	0	0

### Bit 15 – CLOCKSINC: Clock Read Sync Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.CLOCKSINC bit is complete.
1	Write synchronization for CTRLA.CLOCKSINC bit is ongoing.

### Bit 11 – MASK0: Mask 0 Synchronization Busy Status

Value	Description
0	Write synchronization for MASK0 register is complete.
1	Write synchronization for MASK0 register is ongoing.

### Bit 5 – ALARM0: Alarm 0 Synchronization Busy Status

Value	Description
0	Write synchronization for ALARM0 register is complete.
1	Write synchronization for ALARM0 register is ongoing.

### Bit 3 – COUNT: Count Value Synchronization Busy Status

Value	Description
0	Read/write synchronization for COUNT register is complete.
1	Read/write synchronization for COUNT register is ongoing.

**Bit 2 – FREQCORR: Frequency Correction Synchronization Busy Status**

Value	Description
0	Read/write synchronization for FREQCORR register is complete.
1	Read/write synchronization for FREQCORR register is ongoing.

**Bit 1 – ENABLE: Enable Synchronization Busy Status**

Value	Description
0	Read/write synchronization for CTRLA.ENABLE bit is complete.
1	Read/write synchronization for CTRLA.ENABLE bit is ongoing.

**Bit 0 – SWRST: Software Reset Synchronization Busy Status**

Value	Description
0	Read/write synchronization for CTRLA.SWRST bit is complete.
1	Read/write synchronization for CTRLA.SWRST bit is ongoing.

## 25.12.8. Frequency Correlation

**Name:** FREQCORR

**Offset:** 0x14

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	SIGN		VALUE[5:0]					
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 – SIGN: Correction Sign

Value	Description
0	The correction value is positive, i.e., frequency will be increased.
1	The correction value is negative, i.e., frequency will be decreased.

### Bits 5:0 – VALUE[5:0]: Correction Value

These bits define the amount of correction applied to the RTC prescaler.

Value	Description
0	Correction is disabled and the RTC frequency is unchanged.
1 - 127	The RTC frequency is adjusted according to the value.

## 25.12.9. Clock Value in Clock/Calendar mode (CTRLA.MODE=2)

**Name:** CLOCK

**Offset:** 0x18

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Write-Synchronized, Read-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 CTRLA.CLKREP=0, the Hour bit group is in 24-hour format, with values 0-23. When CTRLA.CLKREP=1, HOUR[3:0] has values 1-12, and HOUR[4] represents AM (0) or PM (1).

### Bits 11:6 – MINUTE[5:0]: Minute

0 – 59

**Bits 5:0 – SECOND[5:0]: Second**  
0 – 59

### 25.12.10. Alarm n Value in Clock/Calendar mode (CTRLA.MODE=2)

The 32-bit value of ALARMn is continuously compared with the 32-bit CLOCK value, based on the masking set by MASKn.SEL. When a match occurs, the Alarm n 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 CTRLA.MATCHCLR is one.

**Name:** ALARMn

**Offset:** 0x20 + n\*0x8 [n=0..3]

**Reset:** 0x00000000

**Property:** PAC Write-Protection, 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 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.



### 25.12.11. Alarm n Mask in Clock/Calendar mode (CTRLA.MODE=2)

**Name:** MASKn  
**Offset:** 0x24 + n\*0x8 [n=0..3]  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	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.

Value	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

## 26. DMAC – Direct Memory Access Controller

### 26.1. 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.

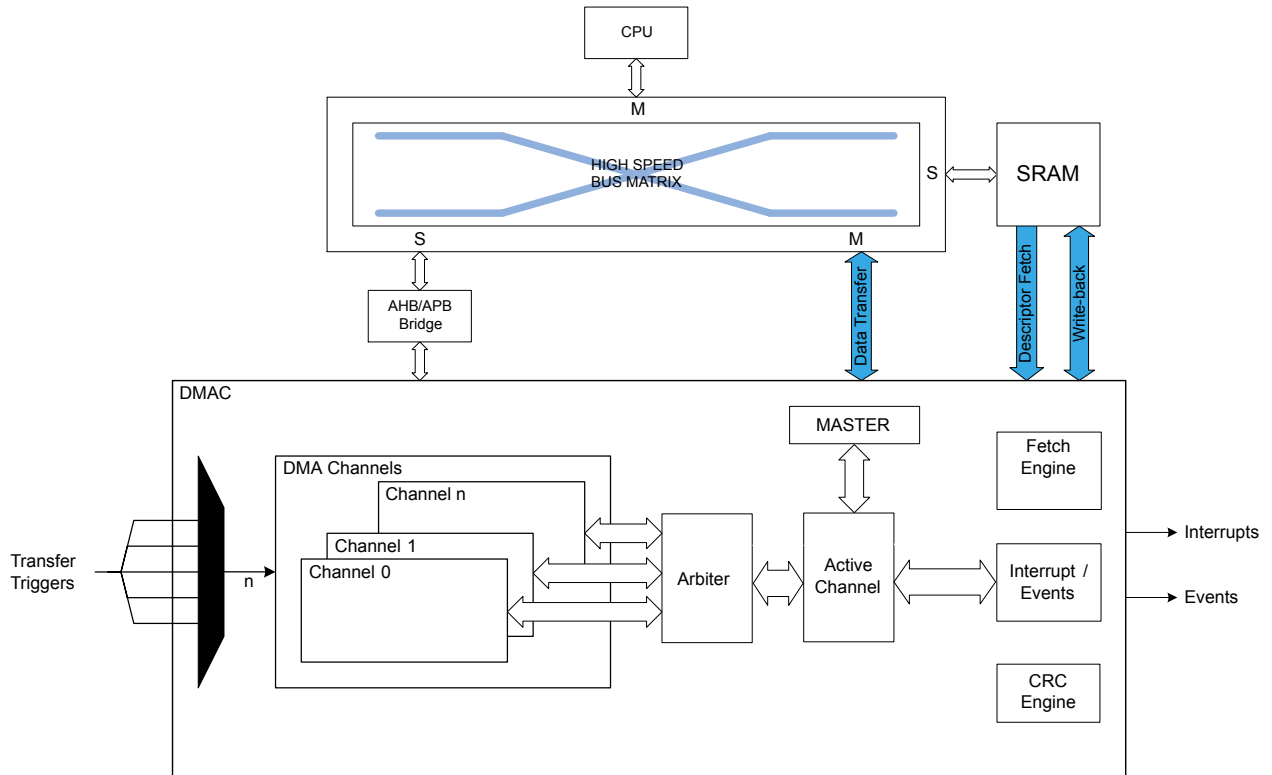
### 26.2. Features

- 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 12channels

- Enable 12 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<sup>®</sup> 802.3)

## 26.3. Block Diagram

Figure 26-1. DMAC 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 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](#) on page 162

### 26.5.3. Clocks

The DMAC bus clock (CLK\_DMACH\_APB) must be configured and enabled in the Main Clock module before using the DMAC.

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](#) on page 135

#### 26.5.4. DMA

Not applicable.

#### 26.5.5. Interrupts

The interrupt request line is connected to the interrupt controller. Using the DMACH interrupt requires the interrupt controller to be configured first.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

#### 26.5.6. Events

The events are connected to the event system.

#### Related Links

[EVSYS – Event System](#) on page 474

#### 26.5.7. Debug Operation

When the CPU is halted in debug mode the DMACH will halt normal operation. The DMACH can be forced to continue operation during debugging. Refer to [DBGCTRL](#) on page 368 for details.

#### 26.5.8. Register Access Protection

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

- Interrupt 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](#) on page 46

#### 26.5.9. Analog Connections

Not applicable.

## 26.6. Functional Description

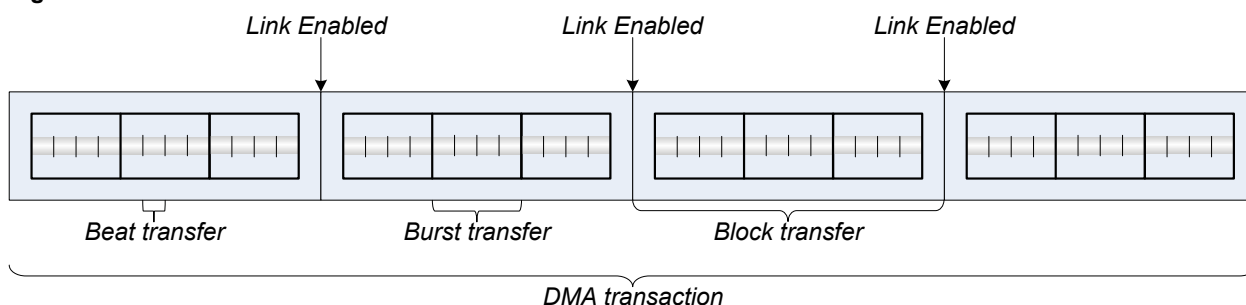
### 26.6.1. Principle of Operation

The DMACH consists of a DMA module and a CRC module.

#### 26.6.1.1. DMA

The DMACH can transfer data between memories and peripherals without interaction from the CPU. The data transferred by the DMACH are called transactions, and these transactions can be split into smaller data transfers. Figure 'DMA Transfer Sizes' shows the relationship between the different transfer sizes:

**Figure 26-2. 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. In contrast to the burst transfer, a block transfer can be interrupted.
- Transaction: The DMAC can link several transfer descriptors by having the first descriptor pointing to the second and so forth, as shown in 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](#) on page 340.

The figure above shows several block transfers linked together, which are called linked descriptors. For further information about linked descriptors, refer to [Linked Descriptors](#) on page 348.

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.

### 26.6.1.2. 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](#) on page 354 for details.

## 26.6.2. Basic Operation

### 26.6.2.1. 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.LVLENx=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)

#### Related Links

[BASEADDR](#) on page 380

[CHCTRLA](#) on page 383

[CHCTRLB](#) on page 384

[CRCCHKSUM](#) on page 366

[CRCCTRL](#) on page 363

[CTRL](#) on page 361

[WRBADDR](#) on page 381

[BTCTRL](#) on page 393

[BTCNT](#) on page 396

[DSTADDR](#) on page 398

[SRCADDR](#) on page 397

#### 26.6.2.2. 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.

#### 26.6.2.3. 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.

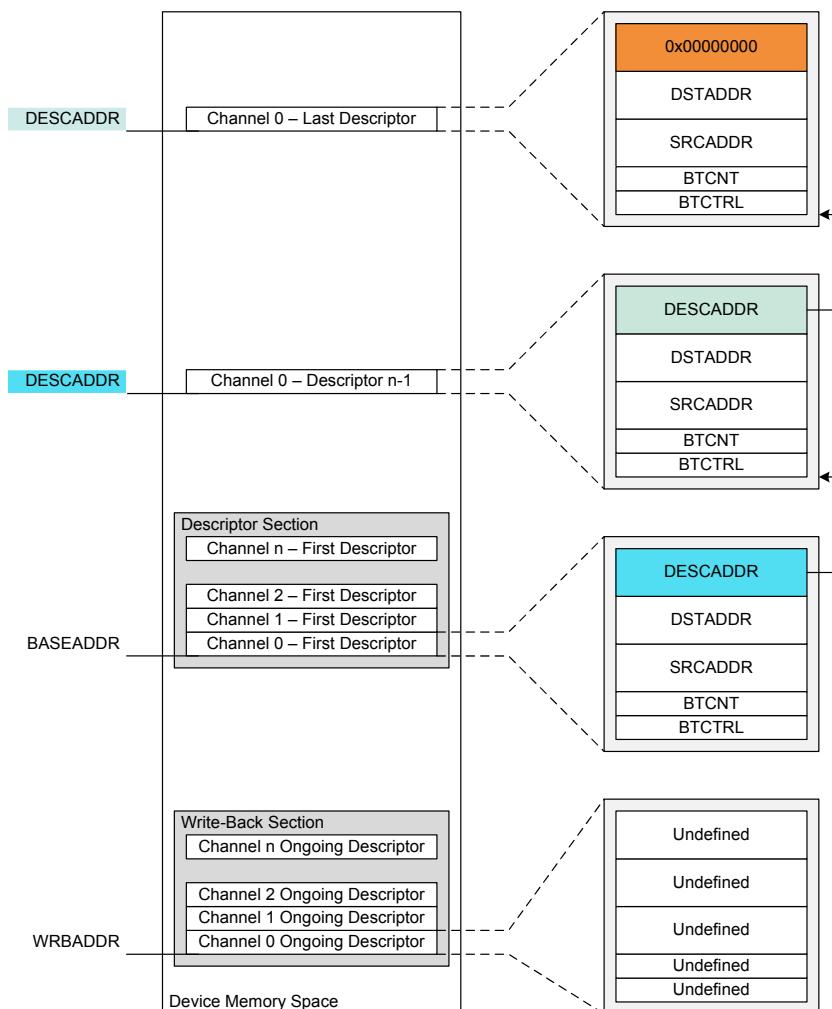
All transfer descriptors must reside in SRAM. The addresses stored in the Descriptor Memory Section Base Address ([BASEADDR](#)) and Write-Back Memory Section Base Address ([WRBADDR](#)) registers tell the DMAC where to find the descriptor memory section and the write-back memory section.



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](#) on page 348.

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](#) on page 348.

**Figure 26-3. 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.

#### 26.6.2.4. 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.

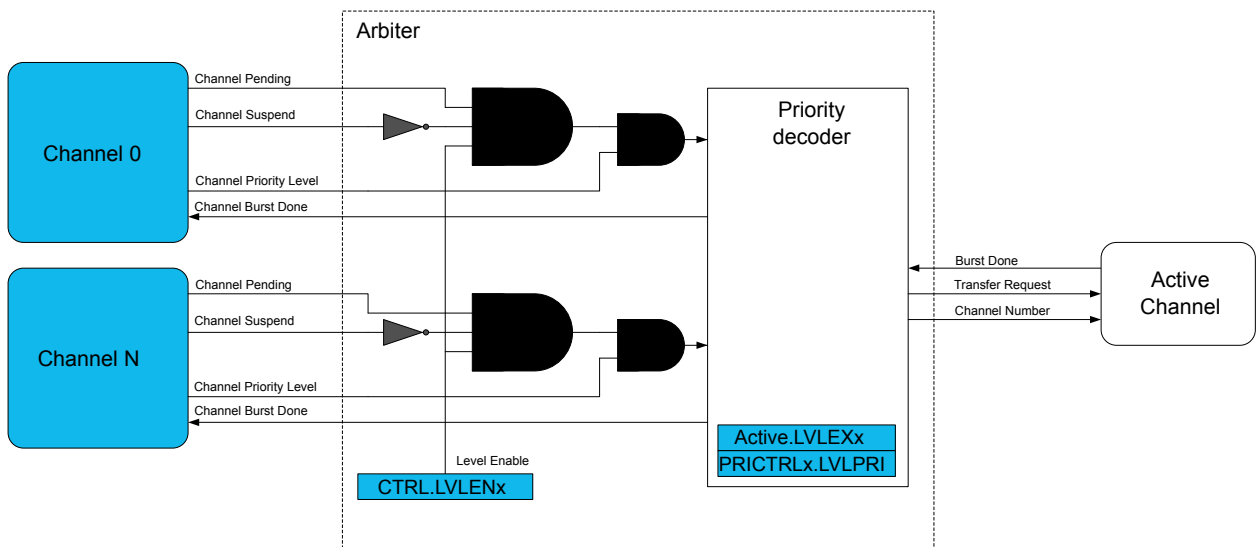
If the upcoming burst transfer is the first for the transfer request, the corresponding Busy Channel x bit in the Busy Channels register will be set (`BUSYCH.BUSYCHx=1`), and it will remain '1' for the subsequent granted burst transfers.

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 26-4. 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`).

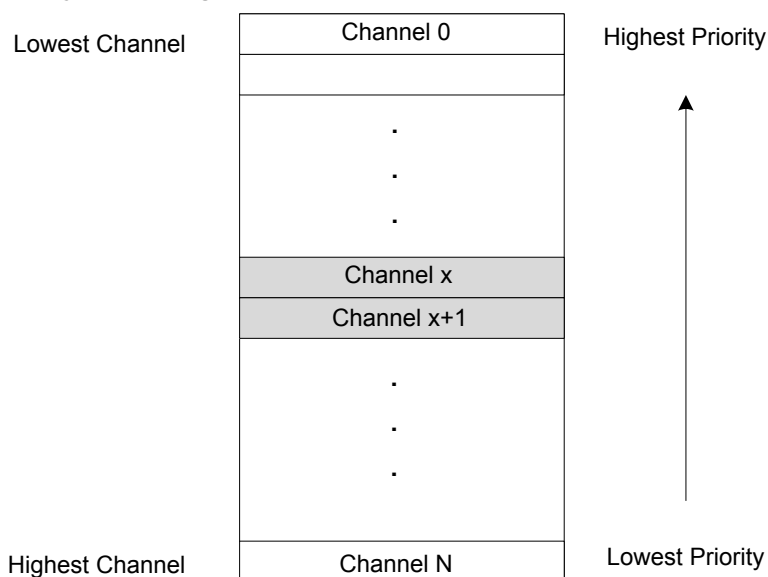
Each DMA channel supports a 4-level priority scheme. The priority level for a channel is configured by writing to the Channel Arbitration Level bit group in the Channel Control B register (CHCTRLB.LVL). As long as all priority levels are enabled, a channel with a lower priority level number will have priority over a channel with a higher 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.

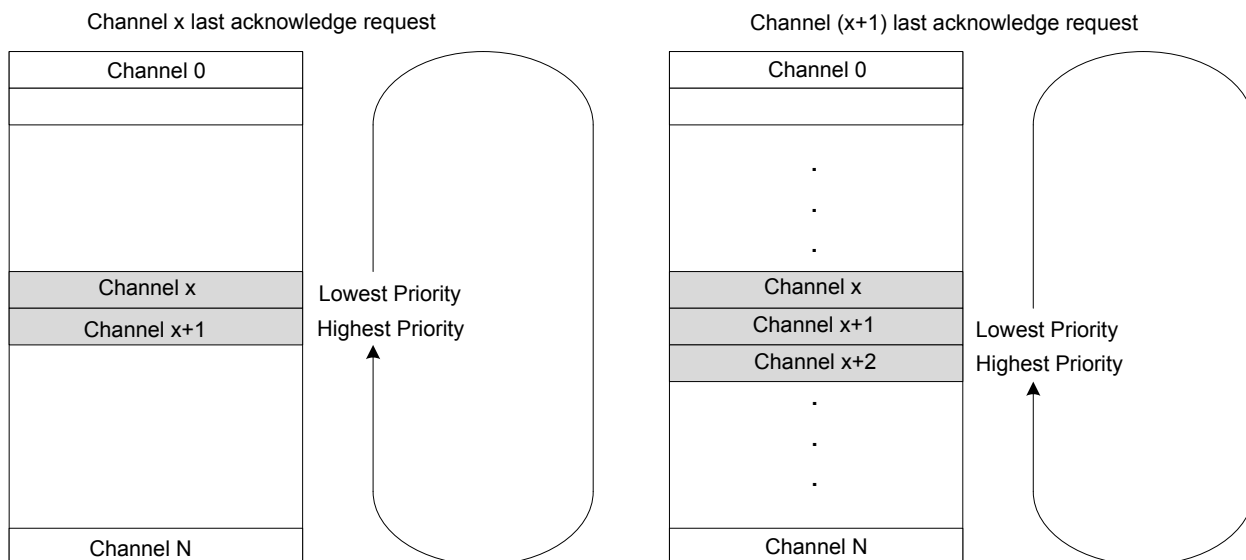
**Figure 26-5. Static Priority Scheduling**



*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 26-6 Dynamic \(Round-Robin\) Priority Scheduling](#) on page 344. 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 26-6. Dynamic (Round-Robin) Priority Scheduling**



### 26.6.2.5. Data Transmission

Before the DMAC can perform a data transmission, a DMA channel has to be configured and enabled, its corresponding transfer descriptor has to be initialized, and the arbiter has to grant the DMA channel access as the active channel.

Once the arbiter has granted a DMA channel access as the active channel (refer to [Figure 26-1 DMAC Block Diagram](#) on page 336) 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, 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.

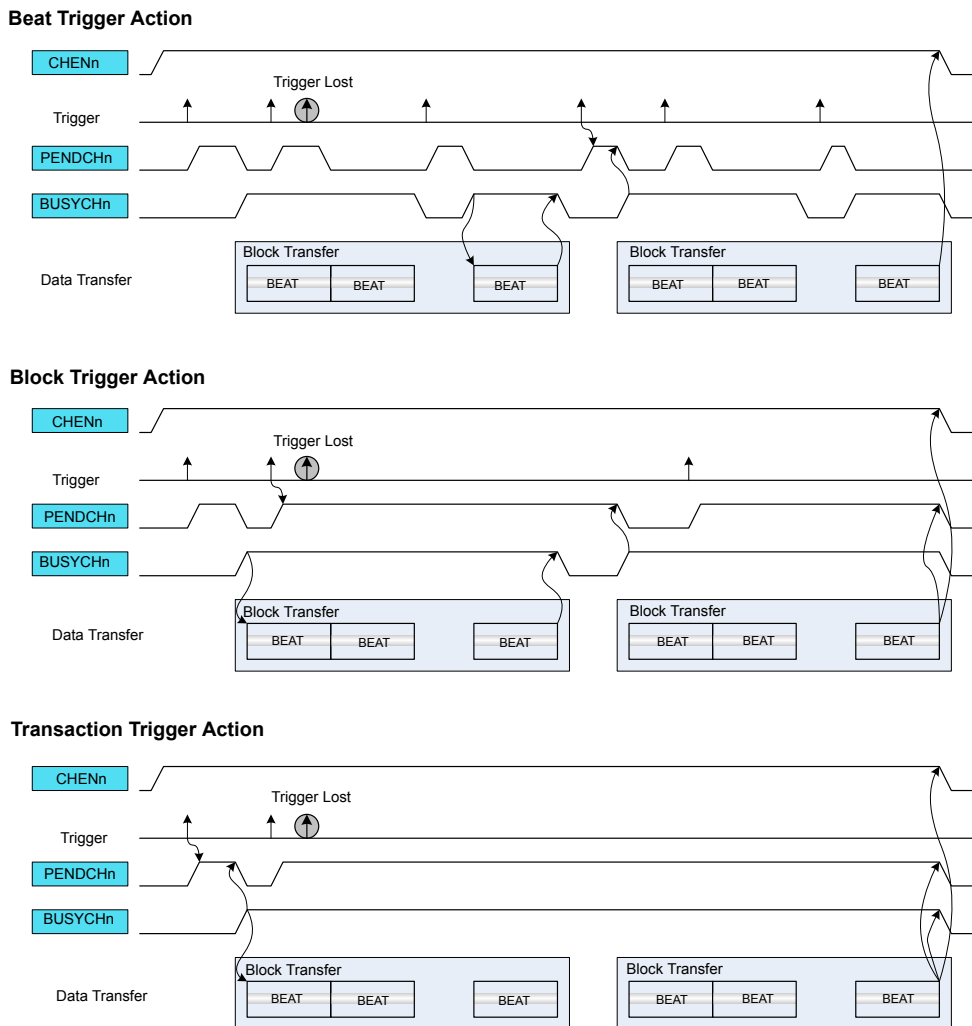
### 26.6.2.6. Transfer Triggers and Actions

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 26-7 Trigger Action and Transfers on page 345 shows an example where triggers are used with two linked block descriptors.

Figure 26-7. 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.

#### 26.6.2.7. 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 ([SRCADDR](#)) register.

The addressing of this DMAC module can be static or incremental, for either source or destination of a block transfer, or both.

Incrementation for the source address of a block transfer is enabled by writing the Source Address Incrementation Enable bit in the Block Transfer Control register ([BTCTRL.SRCINC=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](#):

$$\text{SRCADDR} = \text{SRCADDR}_{\text{START}} + \text{BTCNT} \cdot (\text{BEATSIZE} + 1) \cdot 2^{\text{STEPSIZE}}$$

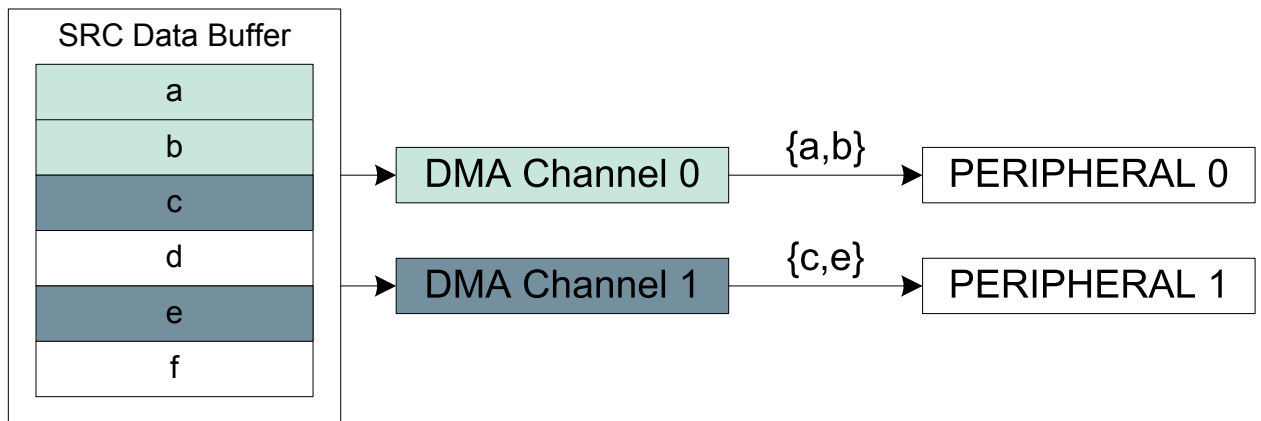
If [BTCTRL.STEPSEL=0](#):

$$\text{SRCADDR} = \text{SRCADDR}_{\text{START}} + \text{BTCNT} \cdot (\text{BEATSIZE} + 1)$$

- $\text{SRCADDR}_{\text{START}}$  is the source address of the first beat transfer in the block transfer
- $\text{BTCNT}$  is the initial number of beats remaining in the block transfer
- $\text{BEATSIZE}$  is the configured number of bytes in a beat
- $\text{STEPSIZE}$  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 26-8. 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.

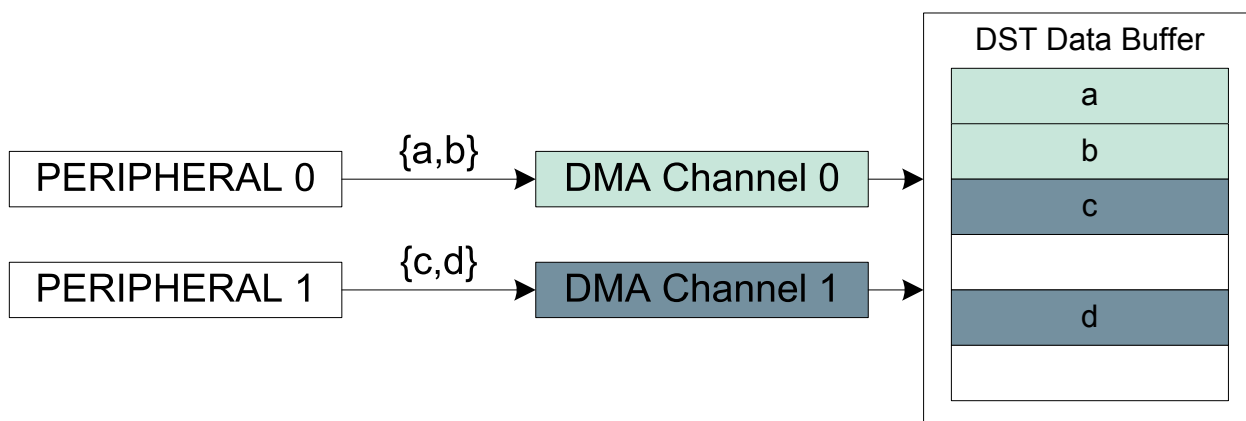
When the destination address incrementation is configured (`BTCTRL.DSTINC=1`), `SRCADDR` must be set and calculated as follows:

$DSTADDR = DSTADDR_{START} + BTCNT \cdot (BEATSIZE + 1) \cdot 2^{STEPSIZE}$	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

Figure 26-9 Destination Address Increment on page 348 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 26-9. Destination Address Increment



### 26.6.2.8. 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.

### 26.6.3. Additional Features

#### 26.6.3.1. Linked Descriptors

A transaction can consist of either a single block transfer or of several block transfers. When a transaction consist of several block transfers it is called linked descriptors.

Figure [Figure 26-3 Memory Sections](#) on page 341 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](#) on page 344.

#### 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.

#### 26.6.3.2. 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](#) on page 340.

#### Related Links

[CHCTRLB](#) on page 384

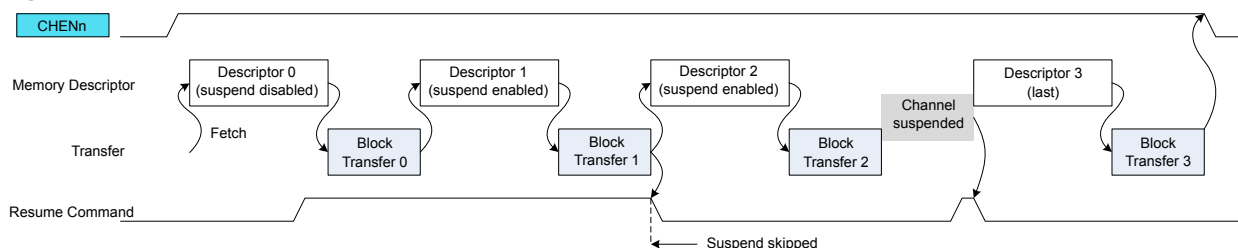
[CHINTFLAG](#) on page 390

[BTCTRL](#) on page 393

### 26.6.3.3. 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 26-10. Channel Suspend/Resume Operation**



#### Related Links

[CHCTRLB](#) on page 384

### 26.6.3.4. 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](#) on page 357.

**Table 26-1. Event Input Action**

Action	CHCTRLB.EVACT	CHCTRLB.TRGSRC
None	NOACT	-
Normal Transfer	TRIG	DISABLE

Action	CHCTRLB.EVACT	CHCTRLB.TRGSRC
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	

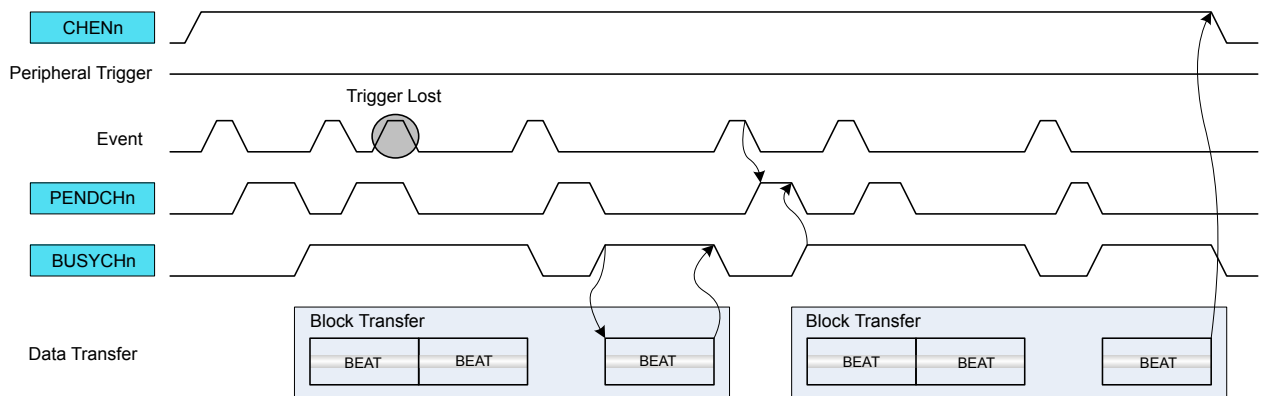
### Normal Transfer

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](#) on page 378.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 26-11. Beat Event Trigger Action**



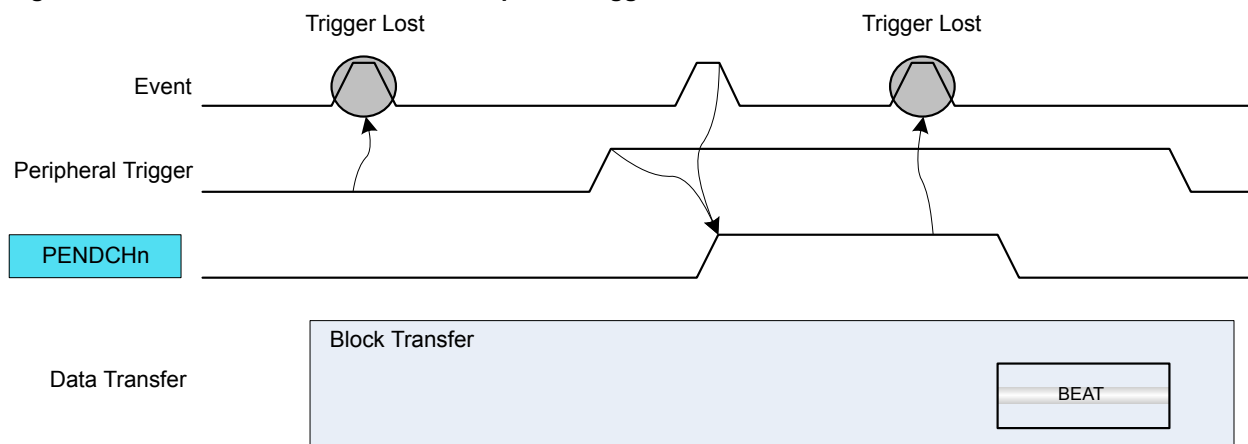
### Conditional Transfer on Strobe

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](#) on page 378.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 26-12. Periodic Event with Beat Peripheral Triggers**



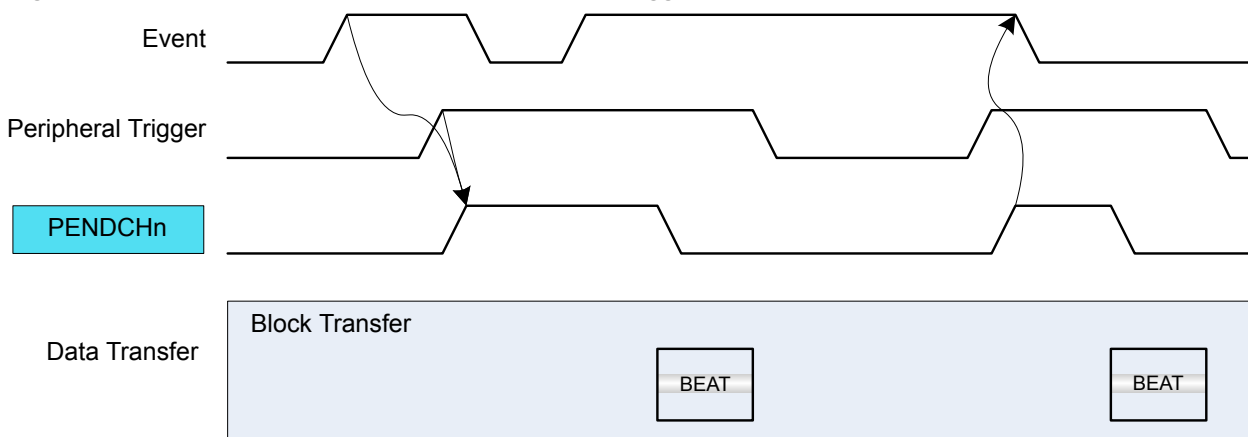
### Conditional Transfer

The event input is used to trigger a conditional transfer on peripherals with pending transfer requests. As an 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](#) on page 378.[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 26-13. Conditional Event with Beat Peripheral Triggers**



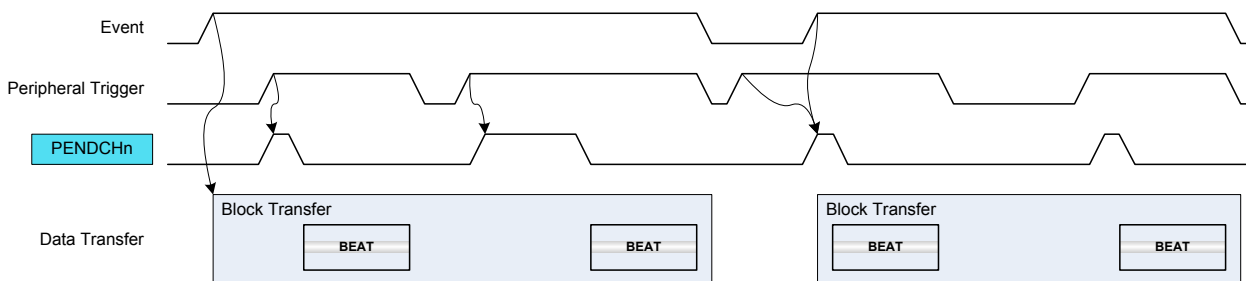
### Conditional Block Transfer

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 26-14. 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](#) on page 349.

### 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](#) on page 349.

### 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.

#### 26.6.3.5. Event Output Selection

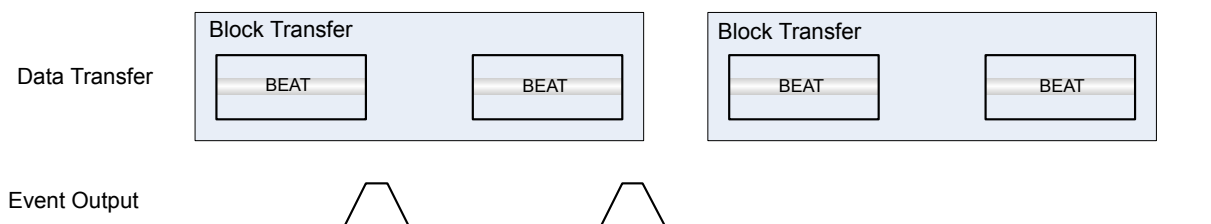
Event output selection is available only for the least significant DMA channels. The pulse width of an event output from a channel is one AHB clock cycle.

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.

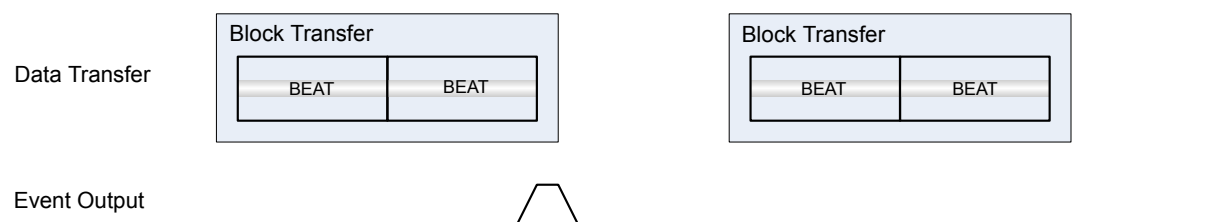
The figure [Figure 26-15 Event Output Generation](#) on page 354 shows an example where the event output generation is enabled in the first block transfer, and disabled in the second block.

**Figure 26-15. Event Output Generation**

**Beat Event Output**



**Block Event Output**



**Related Links**

[CHCTRLB](#) on page 384

[BTCTRL](#) on page 393

**26.6.3.6. 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.

**26.6.3.7. 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.

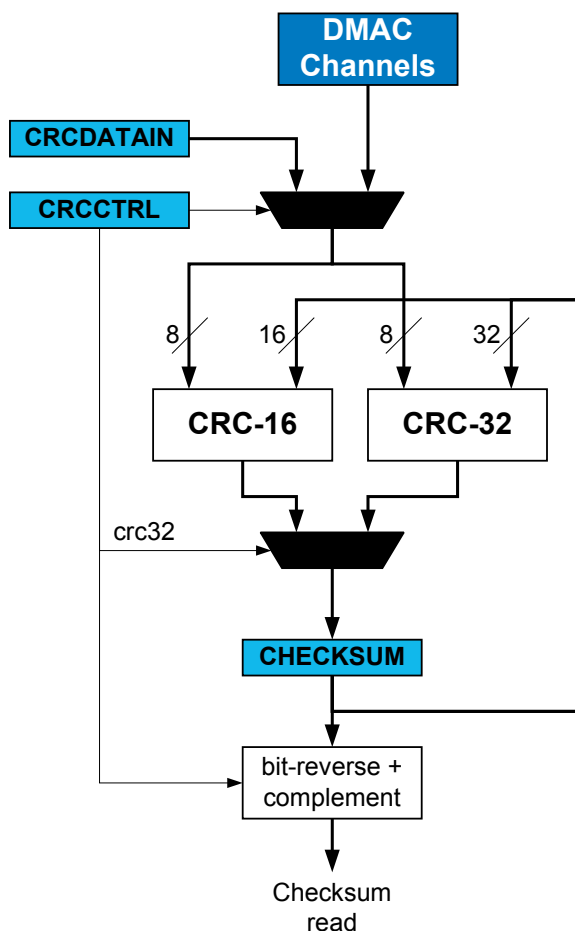
The CRC engine in DMAC supports two commonly used CRC polynomials: CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3). Typically, applying CRC-n (CRC-16 or CRC-32) to a data block of arbitrary length will detect any single alteration that is  $\leq n$  bits in length, and will detect the fraction  $1-2^{-n}$  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 26-16 CRC Generator Block Diagram](#) on page 356.

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 26-16. 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.

#### 26.6.4. DMA Operation

Not applicable.



### 26.6.5. 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](#) on page 344 for details.
- Transfer Error (TERR): Indicates that a bus error has occurred during a burst transfer, or that an invalid descriptor has been fetched. Refer to [Error Handling](#) on page 348 for details.
- Channel Suspend (SUSP): Indicates that the corresponding channel has been suspended. Refer to [Channel Suspend](#) on page 349 and [Data Transmission](#) on page 344 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](#) on page 41

### 26.6.6. 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](#) on page 353 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

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](#) on page 350.

#### Related Links

[EVSYS – Event System](#) on page 474

[CHCTRLB](#) on page 384

[BTCTRL](#) on page 393

#### 26.6.7. 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.

**Note:** In standby sleep mode, the DMAC can only access RAM when it is not back biased (PM.STDBYCFG.BBIASxx=0x0)

#### 26.6.8. Synchronization

Not applicable.

## 26.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRL	7:0							DMAENABLE	SWRST
0x01		15:8					LVLEN3	LVLEN2	LVLEN1	LVLEN0
0x02	CRCCTRL	7:0					CRCPOLY[1:0]		CRCBEATSIZE[1:0]	
0x03		15:8					CRCSRC[5:0]			
0x04	CRCDATAIN	7:0	CRCDATAIN[7:0]							
0x05		15:8	CRCDATAIN[15:8]							
0x06		23:16	CRCDATAIN[23:16]							
0x07		31:24	CRCDATAIN[31:24]							
0x08	CRCCHKSUM	7:0	CRCCHKSUM[7:0]							
0x09		15:8	CRCCHKSUM[15:8]							
0x0A		23:16	CRCCHKSUM[23:16]							
0x0B		31:24	CRCCHKSUM[31:24]							
0x0C	CRCSTATUS	7:0							CRCZERO	CRCBUSY
0x0D	DBGCTRL	7:0								DBGRUN
0x0E	QOSCTRL	7:0			DQOS[1:0]		FQOS[1:0]		WRBQOS[1:0]	
0x0F	Reserved									
0x10	SWTRIGCTRL	7:0	SWTRIG7	SWTRIG6	SWTRIG5	SWTRIG4	SWTRIG3	SWTRIG2	SWTRIG1	SWTRIG0
0x11		15:8					SWTRIG11	SWTRIG10	SWTRIG9	SWTRIG8
0x12		23:16								
0x13		31:24								
0x14	PRICTRL0	7:0	RRLVLEN0				LVLPRIO[3:0]			
0x15		15:8	RRLVLEN1				LVLPRIO[3:0]			
0x16		23:16	RRLVLEN2				LVLPRIO[3:0]			
0x17		31:24	RRLVLEN3				LVLPRIO[3:0]			
0x18 ... 0x1F	Reserved									
0x20	INTPEND	7:0	ID[3:0]							
0x21		15:8	PEND	BUSY	FERR			SUSP	TCMPL	TERR
0x22 ... 0x23	Reserved									
0x24	INTSTATUS	7:0	CHINT7	CHINT6	CHINT5	CHINT4	CHINT3	CHINT2	CHINT1	CHINT0
0x25		15:8					CHINT11	CHINT10	CHINT9	CHINT8
0x26		23:16								
0x27		31:24								
0x28	BUSYCH	7:0	BUSYCH7	BUSYCH6	BUSYCH5	BUSYCH4	BUSYCH3	BUSYCH2	BUSYCH1	BUSYCH0
0x29		15:8					BUSYCH11	BUSYCH10	BUSYCH9	BUSYCH8
0x2A		23:16								
0x2B		31:24								
0x2C	PENDCH	7:0	PENDCH7	PENDCH6	PENDCH5	PENDCH4	PENDCH3	PENDCH2	PENDCH1	PENDCH0
0x2D		15:8					PENDCH11	PENDCH10	PENDCH9	PENDCH8
0x2E		23:16								
0x2F		31:24								

Offset	Name	Bit Pos.									
0x30	ACTIVE	7:0						LVLEX2	LVLEX1	LVLEX0	
0x31		15:8	ABUSY					ID[3:0]			
0x32		23:16	BTCNT[7:0]								
0x33		31:24	BTCNT[15:8]								
0x34	BASEADDR	7:0	BASEADDR[7:0]								
0x35		15:8	BASEADDR[15:8]								
0x36		23:16	BASEADDR[23:16]								
0x37		31:24	BASEADDR[31:24]								
0x38	WRBADDR	7:0	WRBADDR[7:0]								
0x39		15:8	WRBADDR[15:8]								
0x3A		23:16	WRBADDR[23:16]								
0x3B		31:24	WRBADDR[31:24]								
0x3C ... 0x3E	Reserved										
0x3F	CHID	7:0						ID[3:0]			
0x40	CHCTRLA	7:0		RUNSTDBY				ENABLE	SWRST		
0x41 ... 0x43	Reserved										
0x44	CHCTRLB	7:0		LVL[1:0]	EVOE	EVIE	EVACT[2:0]				
0x45		15:8	TRIGSRC[5:0]								
0x46		23:16	TRIGACT[1:0]								
0x47		31:24	CMD[1:0]								
0x48 ... 0x4B	Reserved										
0x4C	CHINTENCLR	7:0					SUSP	TCMPL	TERR		
0x4D	CHINTENSET	7:0					SUSP	TCMPL	TERR		
0x4E	CHINTFLAG	7:0					SUSP	TCMPL	TERR		
0x4F	CHSTATUS	7:0					FERR	BUSY	PEND		

## 26.8. Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#) on page 337.

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.

## 26.8.1. Control

**Name:** CTRL  
**Offset:** 0x00  
**Reset:** 0x00X0  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
					LVLEN3	LVLEN2	LVLEN1	LVLEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
							DMAENABLE	SWRST
Access							R/W	R/W
Reset							0	0

### 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.

### Bits 11,10,9,8 – LVLENx: Priority Level x Enable [x=3..0]

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.

## 26.8.2. 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. 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

Value	Name	Description
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
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 select the CRC polynomial type.

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	HWORD	16-bit bus transfer
0x2	WORD	32-bit bus transfer
0x3		Reserved



### 26.8.3. 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. After the CRCDATAIN register has been written, the number of cycles for the new CRC checksum to be ready is dependent of the configuration of the CRC Beat Size bit group in the CRC Control register([CRCCTRL.CRCBEATSIZE](#)). Each byte needs one clock cycle to be calculated.

## 26.8.4. CRC Checksum

The CRCCHKSUM represents the 16- or 32-bit checksum value and the generated CRC.

**Name:** CRCCHKSUM

**Offset:** 0x08

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Enable-Protected

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

### Bits 31:0 – CRCCHKSUM[31:0]: CRC Checksum

These bits store the generated CRC result. When CRC-16 is enabled, the 16 msb will always read '0'.

These bits should only be read when CRC Module Busy bit in the CRC Status register CRCSTATUS.BUSY=0.

If CRC-16 is selected and [CRCSTATUS.BUSY](#)=0 (CRC generation is completed), this bit group will contain a valid checksum.

If CRC-32 is selected and [CRCSTATUS.BUSY](#)=0 (CRC generation is completed), this bit group will contain a valid *reversed* checksum, i.e.: bit 31 is swapped with bit 0, bit 30 with bit 1, etc.

## 26.8.5. 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.

### Bit 0 – CRCBUSY: CRC Module Busy

When used with an I/O interface ([CRCCTRL.CRCSRC=0x1](#)):

- This bit is cleared by writing a '1' to it
- This bit is set when the CRC Data Input (CRCDATAIN) register is written
- Writing a '1' to this bit will clear the CRC Module Busy bit
- Writing a '0' to this bit has no effect

When used with a DMA channel ([CRCCTRL.CRCSRC=0x20..,0x3F](#)):

- This bit is cleared when the corresponding DMA channel is disabled
- This bit is set when the corresponding DMA channel is enabled
- Writing a '1' to this bit has no effect
- Writing a '0' to this bit has no effect

## 26.8.6. 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.

## 26.8.7. 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

## 26.8.8. Software Trigger Control

**Name:** SWTRIGCTRL  
**Offset:** 0x10  
**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
Access								
Reset								
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	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

### Bits 11:0 – SWTRIGn: 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.

## 26.8.9. 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](#) on page 342.

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 affect the static priority scheme. If the value of this bit group is x, channel x will have the highest priority. The priority will decrease as the channel number increases from x to n, where n is the maximum number of channels. Channel n has higher priority than channel 0, and the priority will continue to decrease from channel 0 to channel (x-1).

This bit group is not reset when round-robin 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](#) on page 342.

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 affect the static priority scheme. If the value of this bit group is x, channel x will have the highest priority. The priority will decrease as the channel number increases from x to n, where n is the maximum number of channels. Channel n has higher priority than channel 0, and the priority will continue to decrease from channel 0 to channel (x-1).

This bit group is not reset when round-robin 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](#) on page 342.

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 affect the static priority scheme. If the value of this bit group is x, channel x will have the highest priority. The priority will decrease as the channel number increases from x to n, where n is the maximum number of channels. Channel n has higher priority than channel 0, and the priority will continue to decrease from channel 0 to channel (x-1).

This bit group is not reset when round-robin 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](#) on page 342.

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 – LVLPRIO[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 affect the static priority scheme. If the value of this bit group is  $x$ , channel  $x$  will have the highest priority. The priority will decrease as the channel number increases from  $x$  to  $n$ , where  $n$  is the maximum number of channels. Channel  $n$  has higher priority than channel 0, and the priority will continue to decrease from channel 0 to channel  $(x-1)$ .

This bit group is not reset when round-robin arbitration gets disabled (PRICTRL0.RRLVLEN0 written to '0').

## 26.8.10. Interrupt Pending

This register allows the user to identify the lowest DMA channel with pending interrupt.

**Name:** INTPEND

**Offset:** 0x20

**Reset:** 0x0000

**Property:** -

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.

## 26.8.11. 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
Access					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

### Bits 11:0 – CHINTn: 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.

## 26.8.12. Busy Channels

**Name:** BUSYCH  
**Offset:** 0x28  
**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					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

### Bits 11:0 – BUSYCHn: 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.

### 26.8.13. Pending Channels

**Name:** PENDCH  
**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
Access					PENDCH11	PENDCH10	PENDCH9	PENDCH8
Reset					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	PENDCH7	PENDCH6	PENDCH5	PENDCH4	PENDCH3	PENDCH2	PENDCH1	PENDCH0
Reset	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 11:0 – PENDCHn: 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.

## 26.8.14. 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[3:0]		
Access	R				R	R	R	R
Reset	0				0	0	0	0
Bit	7	6	5	4	3	2	1	0
						LVLEX2	LVLEX1	LVLEX0
Access						R	R	R
Reset						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 11:8 – ID[3:0]: 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 2,1,0 – LVLEXx: Level x Channel Trigger Request Executing [x=2..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.

## 26.8.15. Descriptor Memory Section Base Address

**Name:** BASEADDR  
**Offset:** 0x34  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

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

### Bits 31:0 – BASEADDR[31:0]: Descriptor Memory Base Address

These bits store the Descriptor memory section base address. The value must be 128-bit aligned.



## 26.8.16. Write-Back Memory Section Base Address

**Name:** WRBADDR  
**Offset:** 0x38  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

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

### Bits 31:0 – WRBADDR[31:0]: Write-Back Memory Base Address

These bits store the Write-Back memory base address. The value must be 128-bit aligned.

## 26.8.17. Channel ID

**Name:** CHID

**Offset:** 0x3F

**Reset:** 0x00

**Property:** -

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

### 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.

### 26.8.18. Channel Control A

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

**Name:** CHCTRLA

**Offset:** 0x40

**Reset:** 0x00

**Property:** PAC Write-Protection, Enable-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 6 – RUNSTDBY: Channel run in standby

This bit is used to keep the DMAC channel running in standby mode.

This bit is not enable-protected.

Value	Description
0	The DMAC channel is halted in standby.
1	The DMAC channel continues to run in standby.

#### 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.

### 26.8.19. Channel Control B

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

**Name:** CHCTRLB

**Offset:** 0x44

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Enable-Protected

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	R/W
Reset		0	0	0	0	0	0	0

#### Bits 25:24 – CMD[1:0]: Software Command

These bits define the software commands. Refer to [Channel Suspend](#) on page 349 and [Channel Resume and Next Suspend Skip](#) on page 350.

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.

**Table 26-2. Peripheral Trigger Source**

Value	Name	Description
0x00	DISABLE	Only software/event triggers
0x01	TSENS	TSENS Result Ready Trigger
0x02	SERCOM0 RX	SERCOM0 RX Trigger
0x03	SERCOM0 TX	SERCOM0TX Trigger
0x04	SERCOM1 RX	SERCOM1 RX Trigger
0x05	SERCOM1 TX	SERCOM1 TX Trigger
0x06	SERCOM2 RX	SERCOM2 RX Trigger
0x07	SERCOM2 TX	SERCOM2 TX Trigger
0x08	SERCOM3 RX	SERCOM3 RX Trigger
0x09	SERCOM3 TX	SERCOM3 TX Trigger
0x0A	SERCOM4 RX	SERCOM4 RX Trigger
0x0B	SERCOM4 TX	SERCOM4 TX Trigger
0x0C	SERCOM5 RX	SERCOM5 RX Trigger
0x0D	SERCOM5 TX	SERCOM5 TX Trigger
0x0E	CAN0 DEBUG	CAN0 Debug Trigger
0x0F	CAN1 DEBUG	CAN1 Debug Trigger
0x10	TCC0 OVF	TCC0 Overflow Trigger
0x11	TCC0 MC0	TCC0 Match/Compare 0 Trigger
0x12	TCC0 MC1	TCC0 Match/Compare 1 Trigger
0x13	TCC0 MC2	TCC0 Match/Compare 2 Trigger
0x14	TCC0 MC3	TCC0 Match/Compare 3 Trigger
0x15	TCC1 OVF	TCC1 Overflow Trigger
0x16	TCC1 MC0	TCC1 Match/Compare 0 Trigger
0x17	TCC1 MC1	TCC1 Match/Compare 1 Trigger
0x18	TCC2 OVF	TCC2 Overflow Trigger
0x19	TCC2 MC0	TCC2 Match/Compare 0 Trigger

Value	Name	Description
0x1A	TCC2 MC1	TCC2 Match/Compare 1 Trigger
0x1B	TC0 OVF	TC0 Overflow Trigger
0x1C	TC0 MC0	TC0 Match/Compare 0 Trigger
0x1D	TC0 MC1	TC0 Match/Compare 1 Trigger
0x1E	TC1 OVF	TC1 Overflow Trigger
0x1F	TC1 MC0	TC1 Match/Compare 0 Trigger
0x20	TC1 MC1	TC1 Match/Compare 1 Trigger
0x21	TC2 OVF	TC2 Overflow Trigger
0x22	TC2 MC0	TC2 Match/Compare 0 Trigger
0x23	TC2 MC1	TC2 Match/Compare 1 Trigger
0x24	TC3 OVF	TC3 Overflow Trigger
0x25	TC3 MC0	TC3 Match/Compare 0 Trigger
0x26	TC3 MC1	TC3 Match/Compare 1 Trigger
0x27	TC4 OVF	TC4 Overflow Trigger
0x28	TC4 MC0	TC4 Match/Compare 0 Trigger
0x29	TC4 MC1	TC4 Match/Compare 1 Trigger
0x2A	ADC0 RESRDY	ADC0 Result Ready Trigger
0x2B	ADC1 RESRDY	ADC1 Result Ready Trigger
0x2C	SDADC RESRDY	SDADC Result Ready Trigger
0x2D	DAC EMPTY	DAC Empty Trigger
0x2E	PTC EOC	PTC End of Conversion Trigger
0x2F	PTC WCOMP	PTC Window Compare Trigger
0x30	PTC SEQ	PTC Sequence 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](#) on page 342.

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 ([BTCCTRL.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.

EVACT[2:0]	Name	Description
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

## 26.8.20. Channel Interrupt Enable Clear

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Channel Interrupt Enable Set (CHINTENSET) register. This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

**Name:** CHINTENCLR  
**Offset:** 0x4C  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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.



### 26.8.21. Channel Interrupt Enable Set

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Channel Interrupt Enable Clear (CHINTENCLR) register. This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

**Name:** CHINTENSET  
**Offset:** 0x4D  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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.

## 26.8.22. Channel Interrupt Flag Status and Clear

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

**Name:** CHINTFLAG

**Offset:** 0x4E

**Reset:** 0x00

**Property:** -

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.

### 26.8.23. Channel Status

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

**Name:** CHSTATUS

**Offset:** 0x4F

**Reset:** 0x00

**Property:** -

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.

## 26.9. Register Summary - SRAM

Offset	Name	Bit Pos.								
0x00	BTCTRL	7:0				BLOCKACT[1:0]		EVOSEL[1:0]		VALID
0x01		15:8	STEPSIZE[2:0]			STEPSEL	DSTINC	SRCINC	BEATSIZE[1:0]	
0x02	BTCNT	7:0	BTCNT[7:0]							
0x03		15:8	BTCNT[15:8]							
0x04	SRCADDR	7:0	SRCADDR[7:0]							
0x05		15:8	SRCADDR[15:8]							
0x06		23:16	SRCADDR[23:16]							
0x07		31:24	SRCADDR[31:24]							
0x08	DSTADDR	7:0	DSTADDR[7:0]							
0x09		15:8	DSTADDR[15:8]							
0x0A		23:16	DSTADDR[23:16]							
0x0B		31:24	DSTADDR[31:24]							
0x0C	DESCADDR	7:0	DESCADDR[7:0]							
0x0D		15:8	DESCADDR[15:8]							
0x0E		23:16	DESCADDR[23:16]							
0x0F		31:24	DESCADDR[31:24]							

## 26.10. 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](#) on page 337.

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.

### 26.10.1. Block Transfer Control

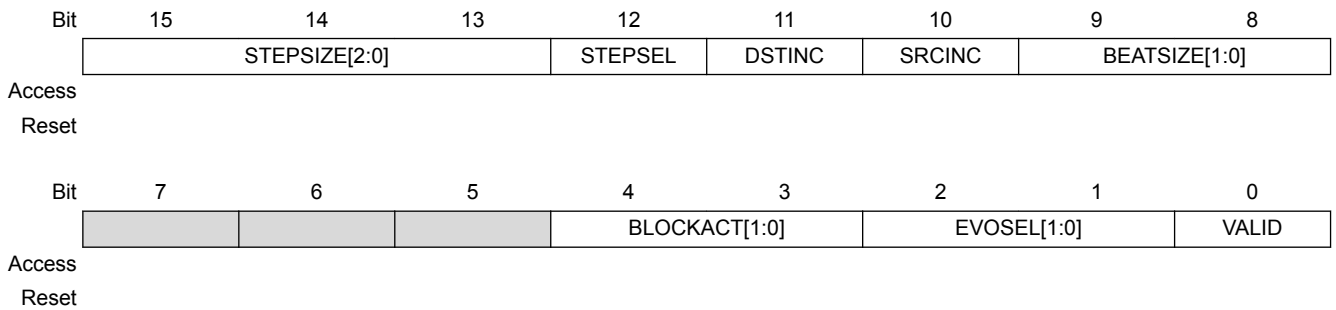
The BTCTRL register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10

**Name:** BTCTRL

**Offset:** 0x00

**Reset:** -

**Property:** -



#### 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.

STEPSIZE[2:0]	Name	Description
0x0	X1	Next ADDR = ADDR + (BEATSIZE+1) * 1
0x1	X2	Next ADDR = ADDR + (BEATSIZE+1) * 2
0x2	X4	Next ADDR = ADDR + (BEATSIZE+1) * 4
0x3	X8	Next ADDR = ADDR + (BEATSIZE+1) * 8
0x4	X16	Next ADDR = ADDR + (BEATSIZE+1) * 16
0x5	X32	Next ADDR = ADDR + (BEATSIZE+1) * 32
0x6	X64	Next ADDR = ADDR + (BEATSIZE+1) * 64
0x7	X128	Next ADDR = ADDR + (BEATSIZE+1) * 128

#### Bit 12 – STEPSEL: Step Selection

This bit selects if source or destination addresses are using the step size settings.

STEPSEL	Name	Description
0x0	DST	Step size settings apply to the destination address
0x1	SRC	Step size settings apply to the source address

#### Bit 11 – DSTINC: Destination Address Increment Enable

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.

BEATSIZE[1:0]	Name	Description
0x0	BYTE	8-bit bus transfer
0x1	HWORDB	16-bit bus transfer
0x2	WORD	32-bit bus transfer
0x3		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.

### 26.10.2. Block Transfer Count

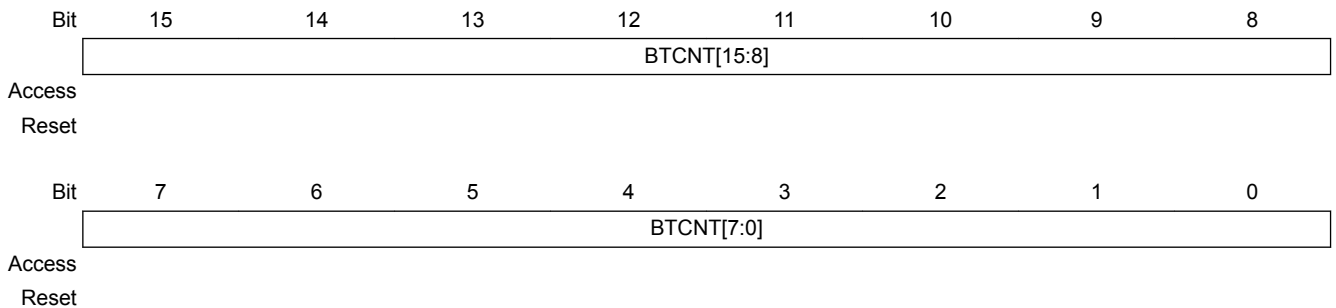
The BTCNT register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10

**Name:** BTCNT

**Offset:** 0x02

**Reset:** -

**Property:** -



#### 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.



### 26.10.3. Block Transfer Source Address

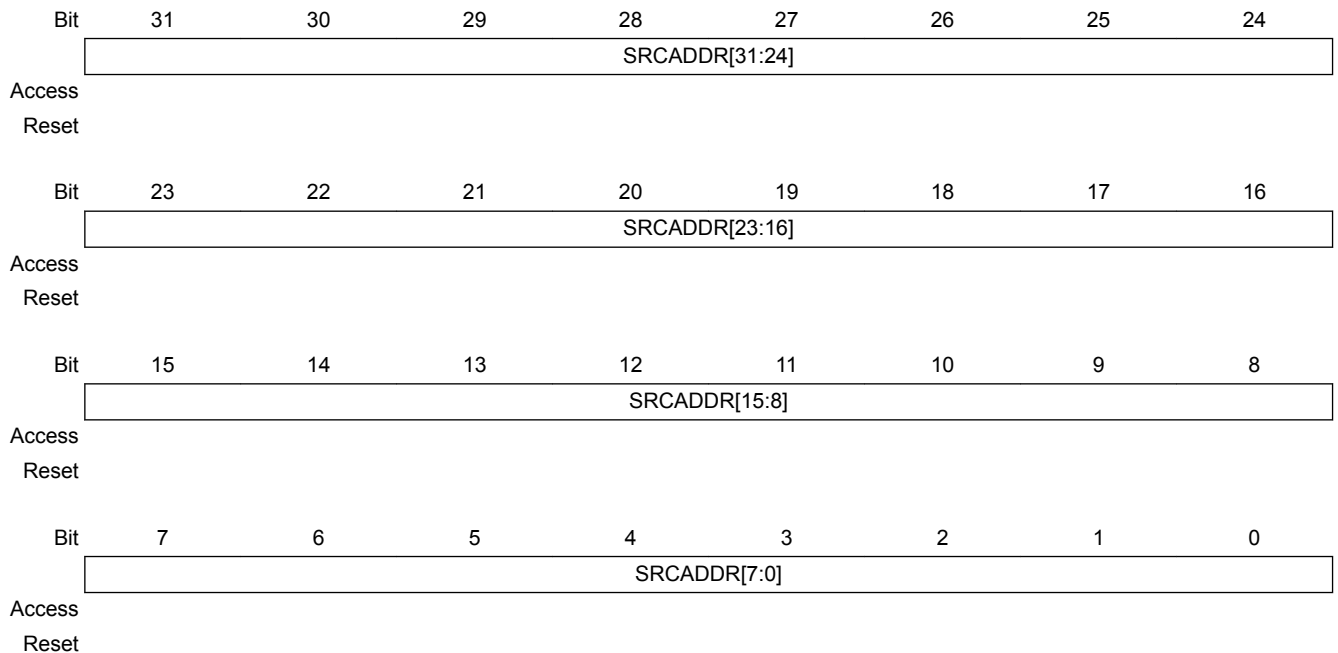
The SRCADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10

**Name:** SRCADDR

**Offset:** 0x04

**Reset:** -

**Property:** -



#### 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.

#### 26.10.4. Block Transfer Destination Address

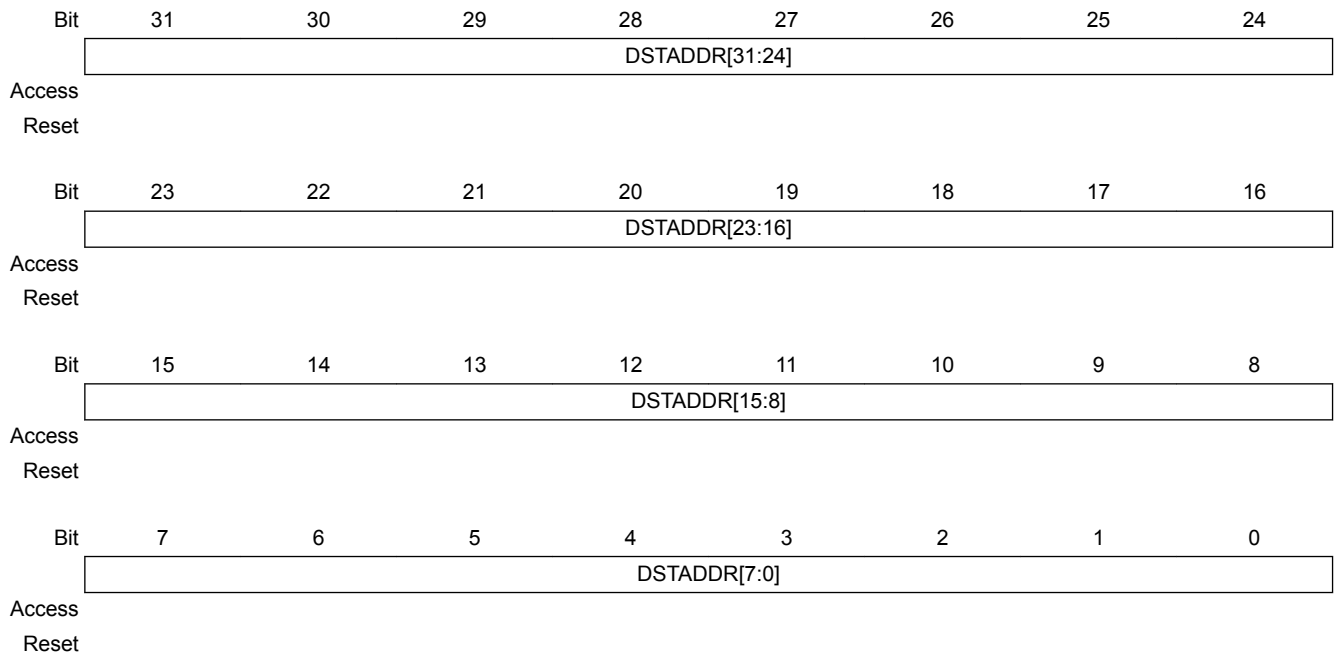
The DSTADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10

**Name:** DSTADDR

**Offset:** 0x08

**Reset:** -

**Property:** -



#### 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.

### 26.10.5. Next Descriptor Address

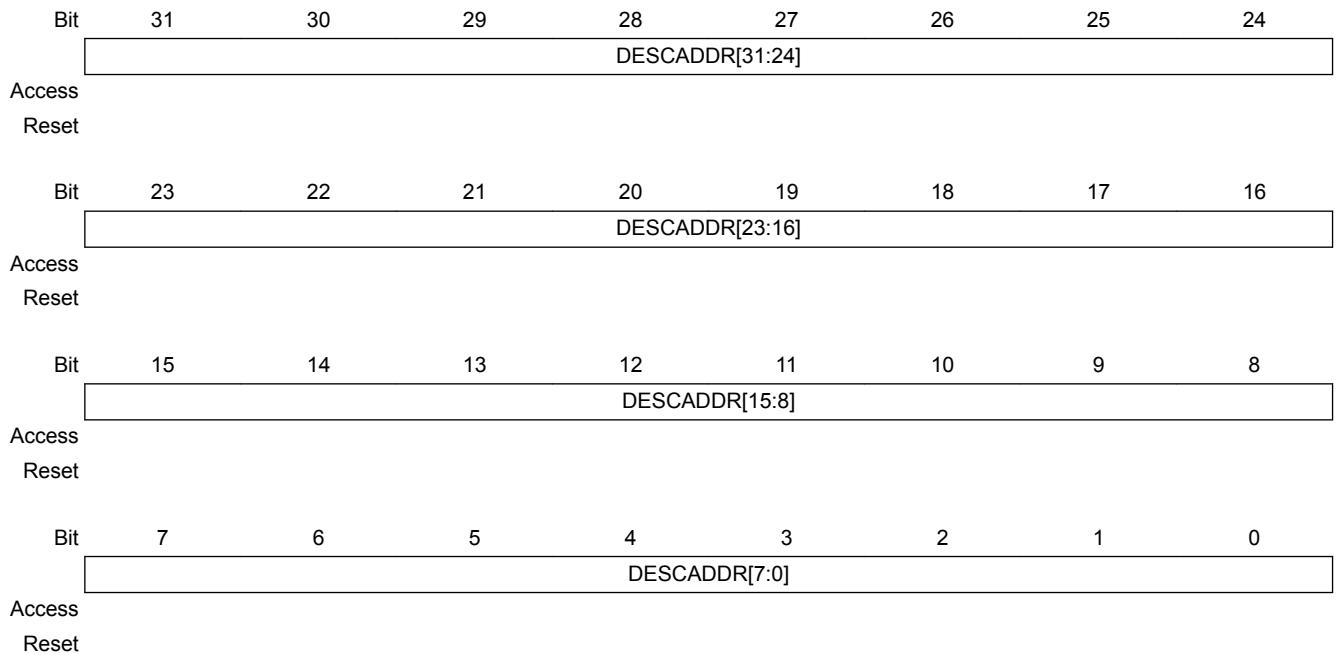
The DESCADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number \* 0x10

**Name:** DESCADDR

**Offset:** 0x0C

**Reset:** -

**Property:** -



#### 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.

## 27. EIC – External Interrupt Controller

### 27.1. 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.

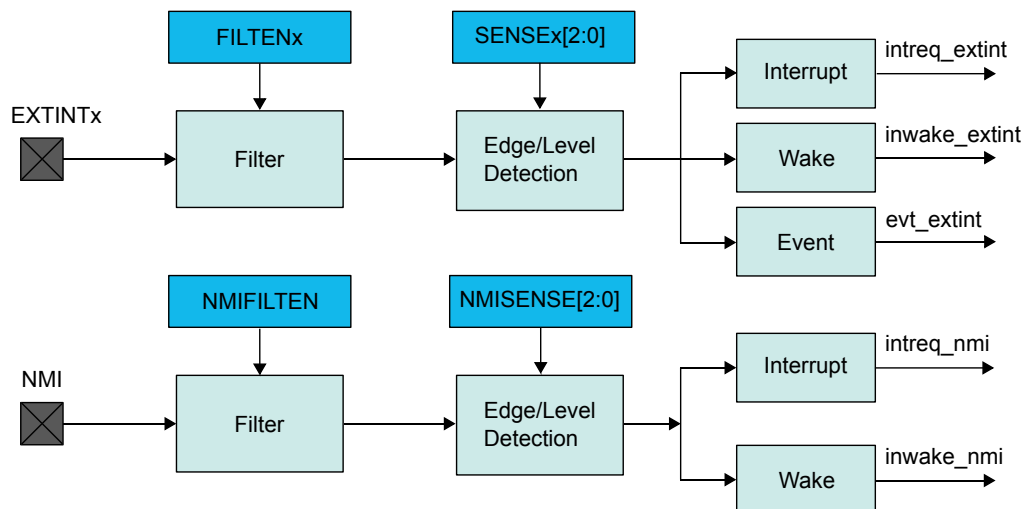
A separate non-maskable interrupt (NMI) is also supported. It has properties similar to the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other interrupt mode.

### 27.2. Features

- Up to 32 external pins, plus one non-maskable pin
- Dedicated, individually maskable interrupt for each pin
- Interrupt on rising, falling, or both edges
- synchronous or asynchronous edge detection mode
- Interrupt on high or low levels
- Asynchronous interrupts for sleep modes without clock
- Filtering of external pins
- Event generation

### 27.3. Block Diagram

Figure 27-1. EIC Block Diagram



## 27.4. Signal Description

Signal Name	Type	Description
EXTINT[31..0]	Digital Input	External interrupt pin
NMI	Digital Input	Non-maskable interrupt pin

One signal can be mapped on several pins.

### Related Links

[I/O Multiplexing and Considerations](#) on page 27

## 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 EIC's I/O lines requires the I/O pins to be configured.

### Related Links

[PORT - I/O Pin Controller](#) on page 444

### 27.5.2. Power Management

All interrupts are available in all sleep modes, but the EIC can be configured to automatically mask some interrupts in order to prevent device wake-up.

The EIC will continue to operate in any sleep mode where the selected source clock is running. The EIC's interrupts can be used to wake up the device from sleep modes. Events connected to the Event System can trigger other operations in the system without exiting sleep modes.

### Related Links

[PM – Power Manager](#) on page 162

### 27.5.3. Clocks

The EIC bus clock (CLK\_EIC\_APB) can be enabled and disabled by the Main Clock Controller, the default state of CLK\_EIC\_APB can be found in the Peripheral Clock Masking section.

Some optional functions need a peripheral clock, which can either be a generic clock (GCLK\_EIC, for wider frequency selection) or a Ultra Low Power 32KHz clock (CLK\_ULP32K, for highest power efficiency). One of the clock sources must be configured and enabled before using the peripheral:

GCLK\_EIC is configured and enabled in the Generic Clock Controller.

CLK\_ULP32K is provided by the internal ultra-low-power (OSCULP32K) oscillator in the OSC32KCTRL module.

Both GCLK\_EIC and CLK\_ULP32K are 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

[MCLK – Main Clock](#) on page 131

[Peripheral Clock Masking](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

#### 27.5.4. DMA

Not applicable.

#### 27.5.5. Interrupts

There are two interrupt request lines, one for the external interrupts (EXTINT) and one for non-maskable interrupt (NMI).

The EXTINT interrupt request line is connected to the interrupt controller. Using the EIC interrupt requires the interrupt controller to be configured first.

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](#) on page 41

#### 27.5.6. 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](#) on page 474

#### 27.5.7. 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.

#### 27.5.8. 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 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](#) on page 46

#### 27.5.9. Analog Connections

Not applicable.

## 27.6. Functional Description

### 27.6.1. 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 or by CLK\_ULP32K.

## 27.6.2. Basic Operation

### 27.6.2.1. Initialization

The EIC must be initialized in the following order:

1. Enable CLK\_EIC\_APB
2. If required, configure the NMI by writing the Non-Maskable Interrupt Control register (**NMICTRL**)
3. When the NMI is used or synchronous edge detection or filtering are required, enable GCLK\_EIC or CLK\_ULP32K.  
GCLK\_EIC is used when a frequency higher than 32KHz is required for filtering, CLK\_ULP32K is recommended when power consumption is the priority. For CLK\_ULP32K write a '1' to the Clock Selection bit in the Control A register (**CTRLA.CKSEL**).
4. Configure the EIC input sense and filtering by writing the Configuration n register (**CONFIGn** on page 418).
5. Enable the EIC.

The following bits are enable-protected, meaning that it can only be written when the EIC is disabled (**CTRLA.ENABLE=0**):

- Clock Selection bit in Control A register (**CTRLA.CKSEL**)

The following registers are enable-protected:

- Event Control register (**EVCTRL**)
- Configuration n register (**CONFIG0, CONFIG1...**)

Enable-protected bits in the **CTRLA** register can be written at the same time when setting **CTRLA.ENABLE** to '1', but not at the same time as **CTRLA.ENABLE** is being cleared.

Enable-protection is denoted by the "Enable-Protected" property in the register description.

### 27.6.2.2. Enabling, Disabling, and Resetting

The EIC is enabled by writing a '1' the Enable bit in the Control A register (**CTRLA.ENABLE**). The EIC is disabled by writing **CTRLA.ENABLE** to '0'.

The EIC is reset by setting the Software Reset bit in the Control register (**CTRLA.SWRST**). All registers in the EIC will be reset to their initial state, and the EIC will be disabled.

Refer to the **CTRLA** register description for details.

### 27.6.3. 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** on page 418.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 or CLK\_ULP32K. Filtering is enabled if bit Filter Enable x in the Configuration n register (**CONFIGn** on page 418.FILTENx) is written to '1'. The majority vote filter samples the external pin three times with GCLK\_EIC or CLK\_ULP32K and outputs the value when two or more samples are equal.

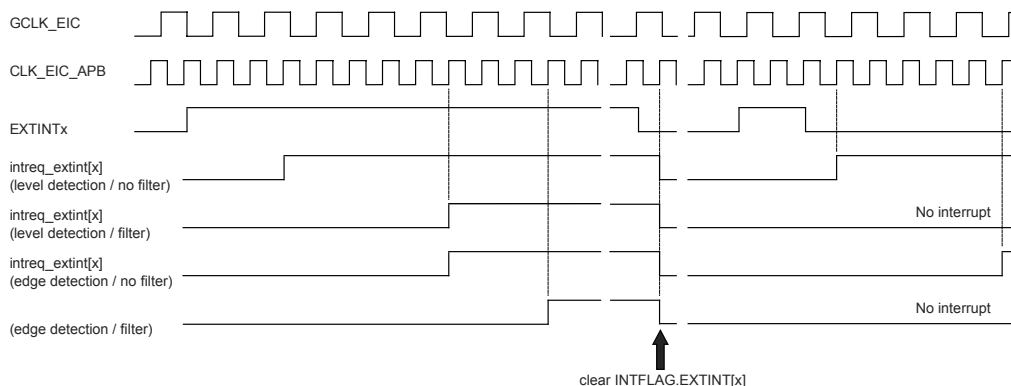
**Table 27-1. Majority Vote Filter**

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 and when filtering is disabled, detection is done asynchronously. Asynchronous detection does not require GCLK\_EIC or CLK\_ULP32K, but interrupt and events can still be generated.

If filtering or edge detection is enabled, the EIC automatically requests GCLK\_EIC or CLK\_ULP32K to operate. The selection between these two clocks is done by writing the Clock Selection bits in the Control A register (CTRLA.CKSEL). GCLK\_EIC must be enabled in the GCLK module.

**Figure 27-2. Interrupt Detections**



The detection delay depends on the detection mode.

**Table 27-2. Interrupt Latency**

Detection mode	Latency (worst case)
Level without filter	Five CLK_EIC_APB periods
Level with filter	Four GCLK_EIC/CLK_ULP32K periods + five CLK_EIC_APB periods
Edge without filter	Four GCLK_EIC/CLK_ULP32K periods + five CLK_EIC_APB periods
Edge with filter	Six GCLK_EIC/CLK_ULP32K periods + five CLK_EIC_APB periods

**Related Links**

[GCLK - Generic Clock Controller](#) on page 114



## 27.6.4. Additional Features

### 27.6.4.1. 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.

### 27.6.4.2. Asynchronous Edge Detection Mode

The EXTINT edge detection can be operated synchronously or asynchronously, selected by the Asynchronous Control Mode bit for external pin x in the External Interrupt Asynchronous Mode register (ASYNCH.ASYNCH[x]). The EIC edge detection is operated synchronously when the Asynchronous Control Mode bit (ASYNCH.ASYNCH[x]) is '0' (default value). It is operated asynchronously when ASYNCH.ASYNCH[x] is written to '1'.

In *Synchronous Edge Detection Mode*, the external interrupt (EXTINT) or the non-maskable interrupt (NMI) pins are sampled using the EIC clock as defined by the Clock Selection bit in the Control A register (CTRLA.CKSEL). The External Interrupt flag (INTFLAG.EXTINT[x]) or Non-Maskable Interrupt flag (NMIFLAG.NMI) is set when the last sampled state of the pin differs from the previously sampled state. In this mode, the EIC clock is required.

The Synchronous Edge Detection Mode can be used in Idle sleep mode.

In *Asynchronous Edge Detection Mode*, the external interrupt (EXTINT) pins or the non-maskable interrupt (NMI) pins set the External Interrupt flag or Non-Maskable Interrupt flag (INTFLAG.EXTINT[x] or NMIFLAG) directly. In this mode, the EIC clock is not requested.

The asynchronous edge detection mode can be used in all sleep modes.

## 27.6.5. DMA Operation

Not applicable.

## 27.6.6. Interrupts

The EIC has the following interrupt sources:

- External interrupt pins (EXTINTx). See [Basic Operation](#) on page 403.
- Non-maskable interrupt pin (NMI). See [Additional Features](#) on page 405.

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 common

interrupt request line for all the interrupt sources, and one interrupt request line for the NMI. The user must read the `INTFLAG` (or `NMIFLAG`) register to determine which interrupt condition is present.

**Note:** Interrupts must be globally enabled for interrupt requests to be generated.

#### Related Links

[Processor and Architecture](#) on page 39

### 27.6.7. Events

The EIC can generate the following output events:

- External event from pin (`EXTINTx`).

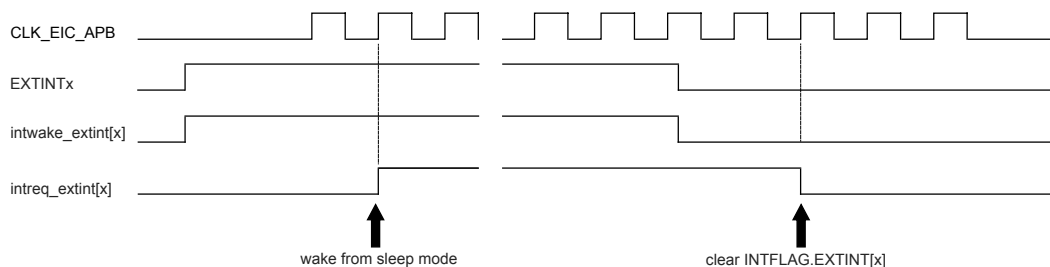
Setting an Event Output Control register (`EVCTRL.EXTINTEO`) enables the corresponding output event. Clearing this bit disables the corresponding output event. Refer to *Event System* for details on configuring the Event System.

When the condition on pin `EXTINTx` matches the configuration in the `CONFIGn` register, the corresponding event is generated, if enabled.

### 27.6.8. Sleep Mode Operation

In sleep modes, an `EXTINTx` pin can wake up the device if the corresponding condition matches the configuration in `CONFIGn` on page 418 register, and the corresponding bit in the Interrupt Enable Set register (`INTENSET`) is written to '1'.

**Figure 27-3. Wake-up Operation Example (High-Level Detection, No Filter, Interrupt Enable Set)**



### 27.6.9. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (`PCHCTRLm.CHEN`). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

The following bits are synchronized when written:

- Software Reset bit in control register (`CTRLA.SWRST`)
- Enable bit in control register (`CTRLA.ENABLE`)

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

## 27.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0				CKSEL			ENABLE	SWRST
0x01	NMICTRL	7:0				ASYNCH	NMIFILTEN		NMISENSE[2:0]	
0x02	NMIFLAG	7:0								NMI
0x03		15:8								
0x04	SYNCBUSY	7:0							ENABLE	SWRST
0x05		15:8								
0x06		23:16								
0x07		31:24								
0x08	EVCTRL	7:0				EXTINTEO[7:0]				
0x09		15:8				EXTINTEO[15:8]				
0x0A		23:16				EXTINTEO[23:16]				
0x0B		31:24				EXTINTEO[31:24]				
0x0C	INTENCLR	7:0				EXTINT[7:0]				
0x0D		15:8				EXTINT[15:8]				
0x0E		23:16				EXTINT[23:16]				
0x0F		31:24				EXTINT[31:24]				
0x10	INTENSET	7:0				EXTINT[7:0]				
0x11		15:8				EXTINT[15:8]				
0x12		23:16				EXTINT[23:16]				
0x13		31:24				EXTINT[31:24]				
0x14	INTFLAG	7:0				EXTINT[7:0]				
0x15		15:8				EXTINT[15:8]				
0x16		23:16				EXTINT[23:16]				
0x17		31:24				EXTINT[31:24]				
0x18	ASYNCH	7:0				ASYNCH[7:0]				
0x19		15:8				ASYNCH[15:8]				
0x1A		23:16				ASYNCH[23:16]				
0x1B		31:24				ASYNCH[31:24]				
0x1C	CONFIG0	7:0	FILTEN1		SENSE1[2:0]	FILTEN0		SENSE0[2:0]		
0x1D		15:8	FILTEN3		SENSE3[2:0]	FILTEN2		SENSE2[2:0]		
0x1E		23:16	FILTEN5		SENSE5[2:0]	FILTEN4		SENSE4[2:0]		
0x1F		31:24	FILTEN7		SENSE7[2:0]	FILTEN6		SENSE6[2:0]		
0x20	CONFIG1	7:0	FILTEN9		SENSE9[2:0]	FILTEN8		SENSE8[2:0]		
0x21		15:8	FILTEN11		SENSE11[2:0]	FILTEN10		SENSE10[2:0]		
0x22		23:16	FILTEN13		SENSE13[2:0]	FILTEN12		SENSE12[2:0]		
0x23		31:24	FILTEN15		SENSE15[2:0]	FILTEN14		SENSE14[2:0]		
0x24	CONFIG2	7:0	FILTEN17		SENSE17[2:0]	FILTEN16		SENSE16[2:0]		
0x25		15:8	FILTEN19		SENSE19[2:0]	FILTEN18		SENSE18[2:0]		
0x26		23:16	FILTEN21		SENSE21[2:0]	FILTEN20		SENSE20[2:0]		
0x27		31:24	FILTEN23		SENSE23[2:0]	FILTEN22		SENSE22[2:0]		

Offset	Name	Bit Pos.					
0x28	CONFIG3	7:0	FILTEN25		SENSE25[2:0]	FILTEN24	SENSE24[2:0]
0x29		15:8	FILTEN26		SENSE26[2:0]	FILTEN26	SENSE26[2:0]
0x2A		23:16	FILTEN27		SENSE27[2:0]	FILTEN28	SENSE28[2:0]
0x2B		31:24	FILTEN31		SENSE31[2:0]	FILTEN30	SENSE30[2:0]

## 27.8. Register Description

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

Some registers 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.

## 27.8.1. Control A

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized, Enable-Protected

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

### Bit 4 – CKSEL: Clock Selection

The EIC can be clocked either by GCLK\_EIC (when a frequency higher than 32KHz is required for filtering) or by CLK\_ULP32K (when power consumption is the priority).

This bit is not Write-Synchronized.

Value	Description
0	The EIC is clocked by GCLK_EIC.
1	The EIC is clocked by CLK_ULP32K.

### Bit 1 – ENABLE: Enable

Due to synchronization there is a delay between writing to CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable bit in the Synchronization Busy register will be set (SYNCBUSY.ENABLE=1). SYNCBUSY.ENABLE will be cleared when the operation is complete.

This bit is not Enable-Protected.

Value	Description
0	The EIC is disabled.
1	The EIC 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 EIC to their initial state, and the EIC 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.

This bit is not Enable-Protected.

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

## 27.8.2. Non-Maskable Interrupt Control

**Name:** NMICTRL  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

### Bit 4 – ASYNCH: Asynchronous Edge Detection Mode

The NMI edge detection can be operated synchronously or asynchronously.

In Synchronous Edge Detection Mode, the non-maskable interrupt (NMI) pin is sampled using the EIC clock as defined by the bit [CTRLA.CKSEL](#). The non-maskable interrupt flag ([NMIFLAG](#)) is set when the pin and the pin sampler have a different value. In this mode, the EIC clock is required. The Synchronous Edge Detection mode can be used in all sleep modes, except STANDBY.

In Asynchronous Edge Detection Mode, the non-maskable interrupt (NMI) pins directly drives the set of the non-maskable interrupt flag ([NMIFLAG](#)). In this mode, the EIC clock is not requested. The Asynchronous Edge Detection Mode can be used in all sleep modes.

Value	Description
0	The NMI edge detection is synchronously operated.
1	The NMI edge detection is asynchronously operated.

### Bit 3 – NMIFILTEN: Non-Maskable Interrupt Filter Enable

Value	Description
0	NMI filter is disabled.
1	NMI filter is enabled.

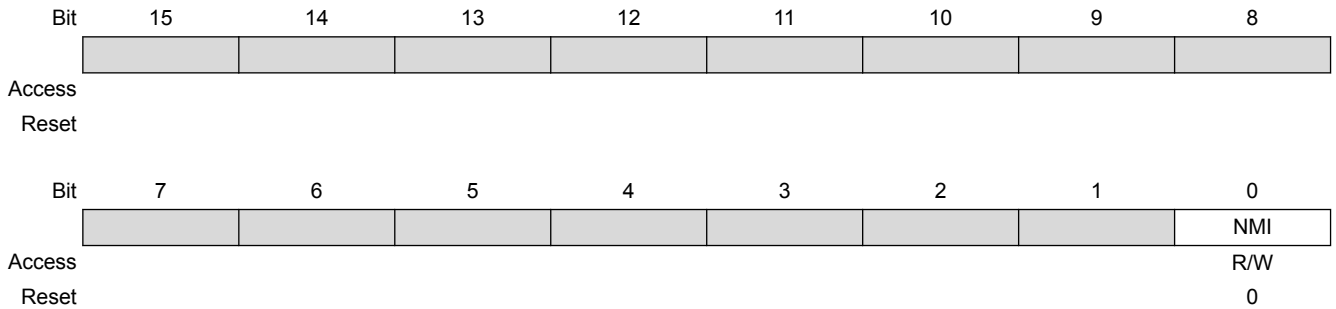
### Bits 2:0 – NMISENSE[2:0]: Non-Maskable Interrupt Sense

These bits define on which edge or level the NMI triggers.

Value	Name	Description
0x0	NONE	No detection
0x1	RISE	Rising-edge detection
0x2	FALL	Falling-edge detection
0x3	BOTH	Both-edge detection
0x4	HIGH	High-level detection
0x5	LOW	Low-level detection
0x6 - 0x7	-	Reserved

### 27.8.3. Non-Maskable Interrupt Flag Status and Clear

**Name:** NMIFLAG  
**Offset:** 0x02  
**Reset:** 0x0000  
**Property:** -



#### Bit 0 – NMI: Non-Maskable Interrupt

This flag is cleared by writing a '1' to it.

This flag is set when the NMI pin matches the NMI sense configuration, and will generate an interrupt request.

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

Writing a '1' to this bit clears the non-maskable interrupt flag.

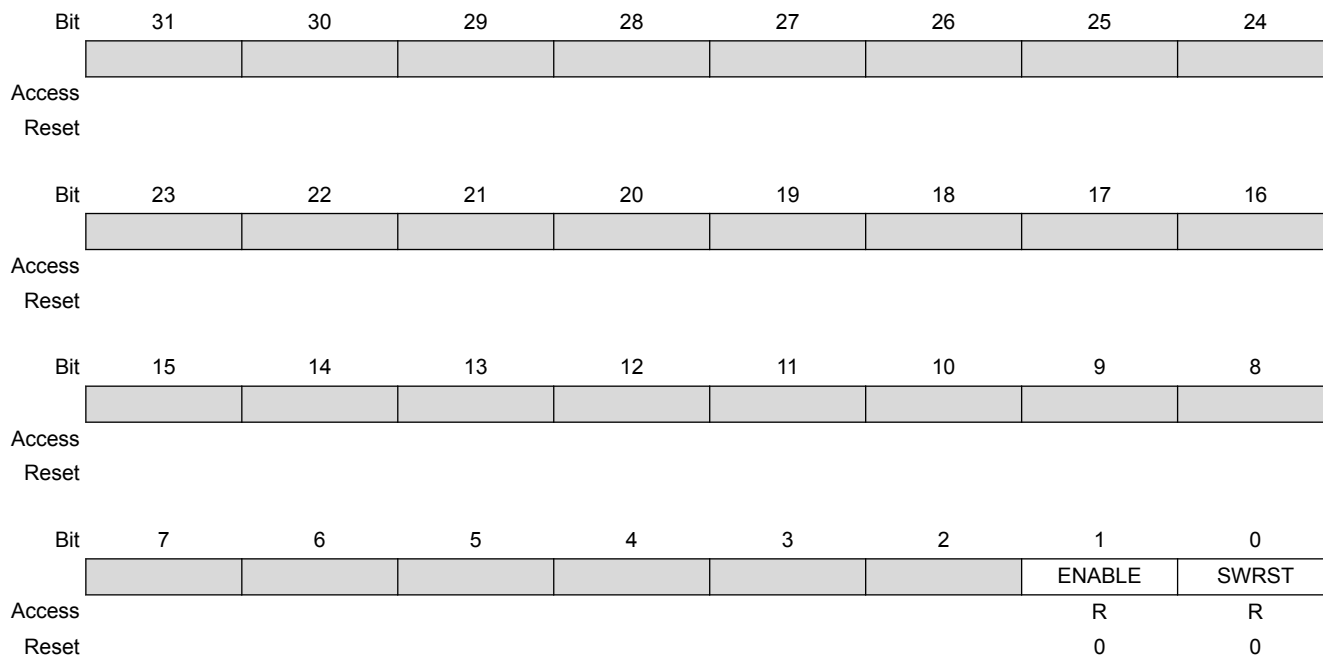
## 27.8.4. Synchronization Busy

**Name:** SYNCBUSY

**Offset:** 0x04

**Reset:** 0x00000000

**Property:** –



### Bit 1 – ENABLE: Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.ENABLE bit is complete.
1	Write synchronization for CTRLA.ENABLE bit is ongoing.

### Bit 0 – SWRST: Software Reset Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.SWRST bit is complete.
1	Write synchronization for CTRLA.SWRST bit is ongoing.



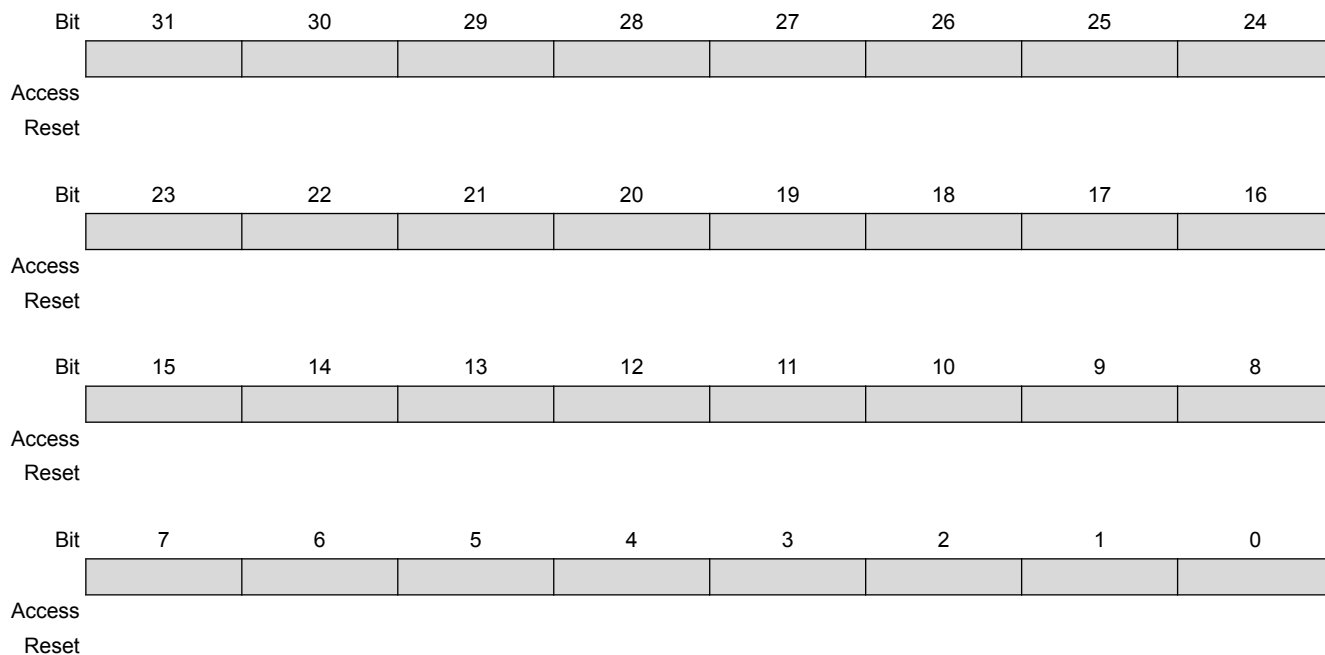
## 27.8.5. Event Control

**Name:** EVCTRL

**Offset:** 0x08

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Enable-Protected



### 27.8.6. Interrupt Enable Clear

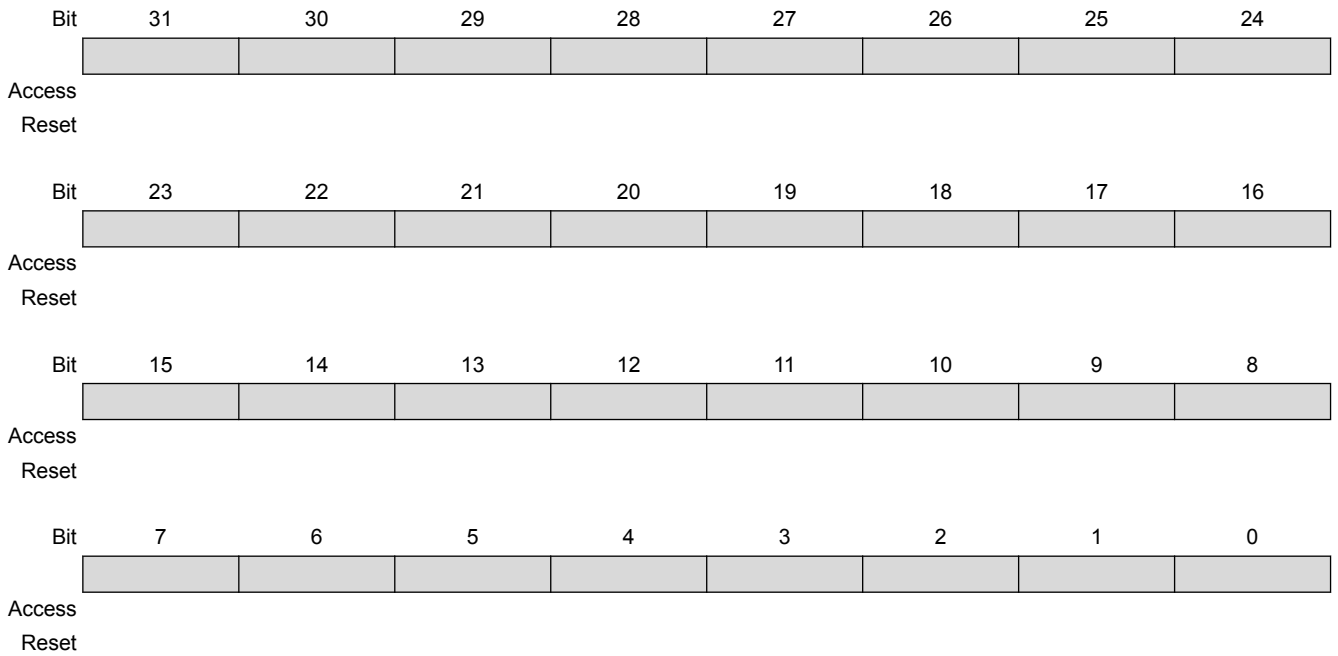
This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

**Name:** INTENCLR

**Offset:** 0x0C

**Reset:** 0x00000000

**Property:** PAC Write-Protection



### 27.8.7. Interrupt Enable Set

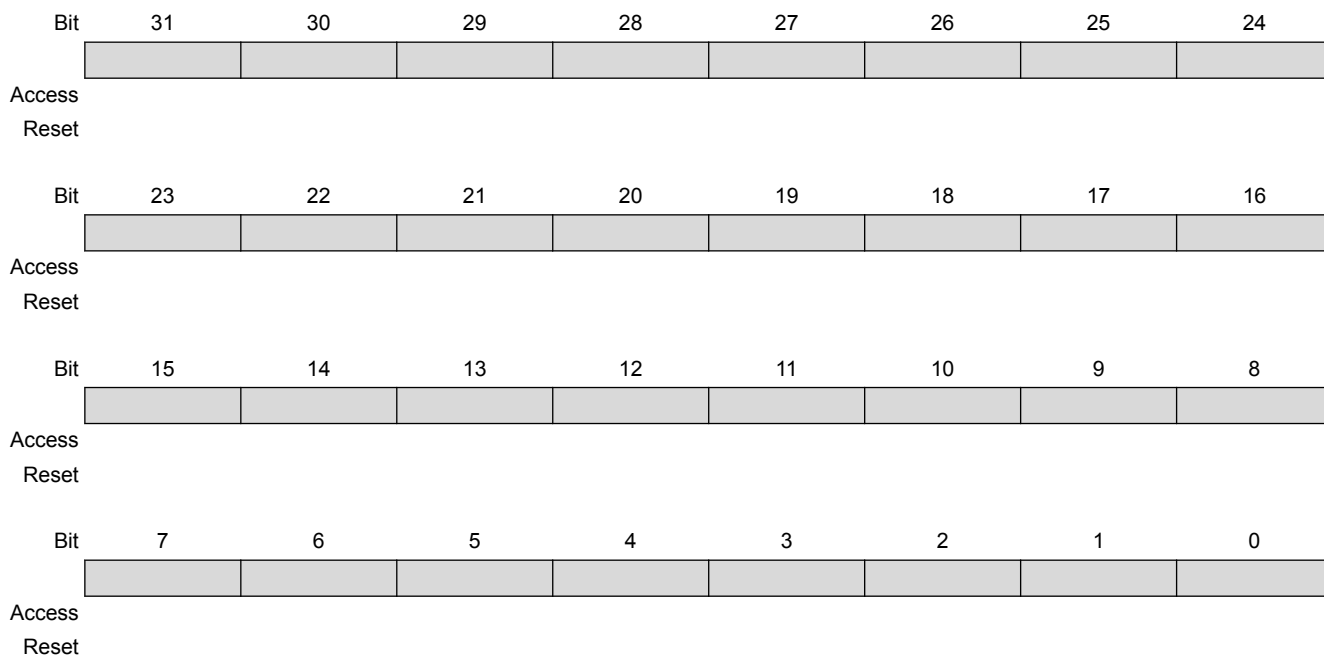
This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

**Name:** INTENSET

**Offset:** 0x10

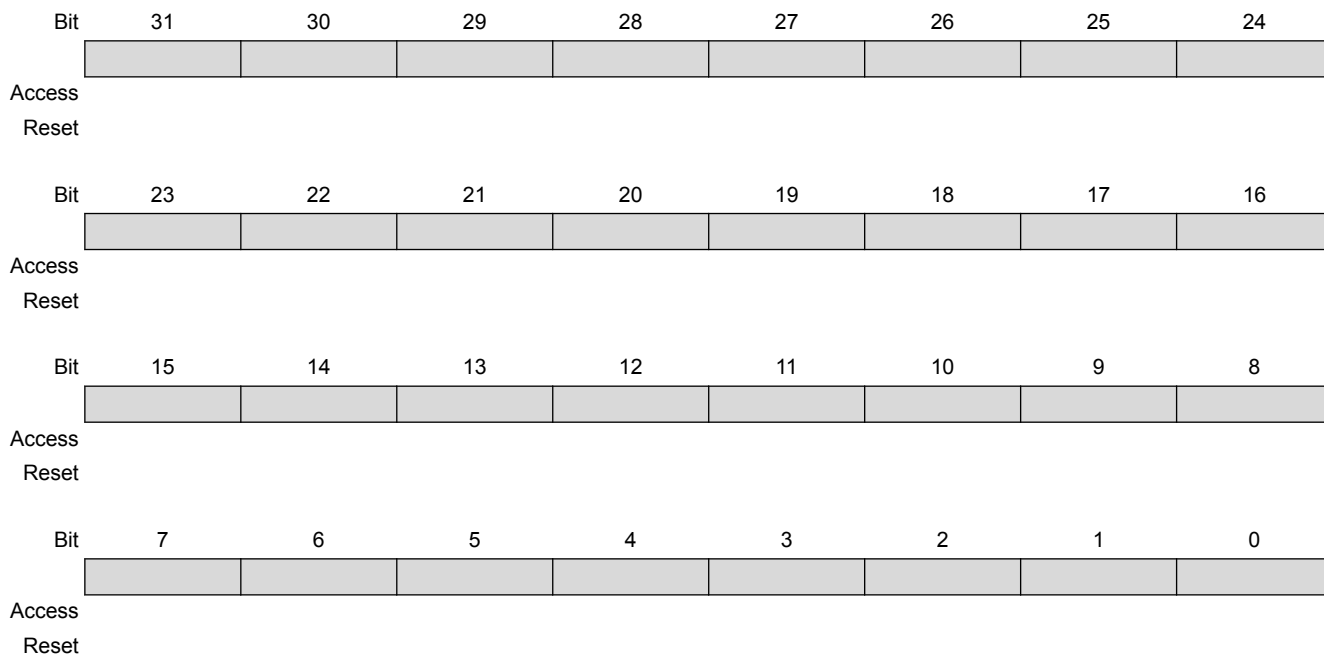
**Reset:** 0x00000000

**Property:** PAC Write-Protection



## 27.8.8. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** -



## 27.8.9. External Interrupt Asynchronous Mode

**Name:** ASYNCH  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	ASYNCH[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
	ASYNCH[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
	ASYNCH[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
	ASYNCH[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 – ASYNCH[31:0]: Asynchronous Edge Detection Mode

Value	Description
0	The EXTINT edge detection is synchronously operated.
1	The EXTINT edge detection is asynchronously operated.

## 27.8.10. Configuration n

**Name:** CONFIGn  
**Offset:** 0x1C+n\*0x4 [n = 0..3]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24	
	FILTEN7		SENSE7[2:0]			FILTEN6		SENSE6[2:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	
Bit	23	22	21	20	19	18	17	16	
	FILTEN5		SENSE5[2:0]			FILTEN4		SENSE4[2:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	
	FILTEN3		SENSE3[2:0]			FILTEN2		SENSE2[2:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	FILTEN1		SENSE1[2:0]			FILTEN0		SENSE0[2:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	

### Bits 3,7,11,15,19,23,27,31 – FILTENx: Filter x Enable [x = 7..0]

Value	Description
0	Filter is disabled for EXTINT[n*8+1] input.
1	Filter is enabled for EXTINT[n*8+1] input.

### Bits 0:2,4:6,8:10,12:14,16:18,20:22,24:26,28:30 – SENSEx: Input Sense x Configuration

These bits define on which edge or level the interrupt or event for EXTINT[n\*8+x] will be generated.

Value	Name	Description
0x0	NONE	No detection
0x1	RISE	Rising-edge detection
0x2	FALL	Falling-edge detection
0x3	BOTH	Both-edge detection
0x4	HIGH	High-level detection
0x5	LOW	Low-level detection
0x6 - 0x7	-	Reserved

## 28. NVMCTRL – Non-Volatile Memory Controller

### 28.1. Overview

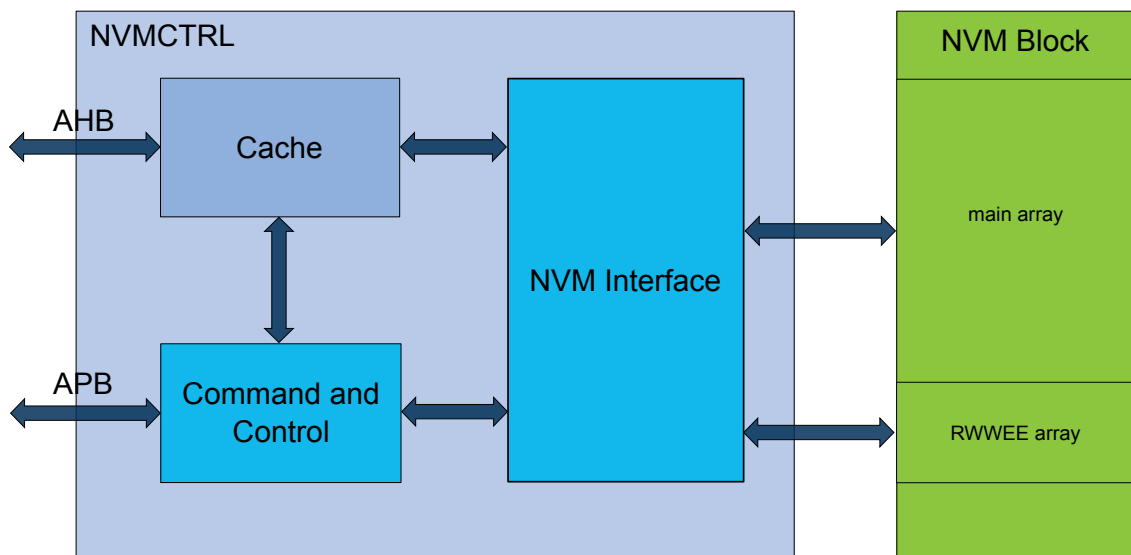
Non-Volatile Memory (NVM) is a reprogrammable Flash memory that retains program and data storage even with power off. It embeds a main array and a separate smaller array intended for EEPROM emulation (RWWEE) that can be programmed while reading the main array. The NVM Controller (NVMCTRL) connects to 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.

### 28.2. Features

- 32-bit AHB interface for reads and writes
- Read While Write EEPROM emulation area
- All NVM sections are memory mapped to the AHB, including calibration and system configuration
- 32-bit APB interface for commands and control
- Programmable wait states for read optimization
- 16 regions can be individually protected or unprotected
- Additional protection for boot loader
- Supports device protection through a security bit
- Interface to Power Manager for power-down of Flash blocks in sleep modes
- Can optionally wake up on exit from sleep or on first access
- Direct-mapped cache

### 28.3. Block Diagram

Figure 28-1. Block Diagram



## 28.4. Signal Description

Not applicable.

## 28.5. Product Dependencies

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

### 28.5.1. 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) SLEEPPRM bit setting. Refer to the [CTRLB.SLEEPPRM](#) register description for more details.

#### Related Links

[PM – Power Manager](#) on page 162

### 28.5.2. 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](#) on page 1105

### 28.5.3. 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.

### 28.5.4. 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](#) on page 427 for details.

### 28.5.5. 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](#) on page 46



### 28.5.6. Analog Connections

Not applicable.

## 28.6. Functional Description

### 28.6.1. 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.

#### 28.6.1.1. 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.

### 28.6.2. 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 28-2. NVM Row Organization**

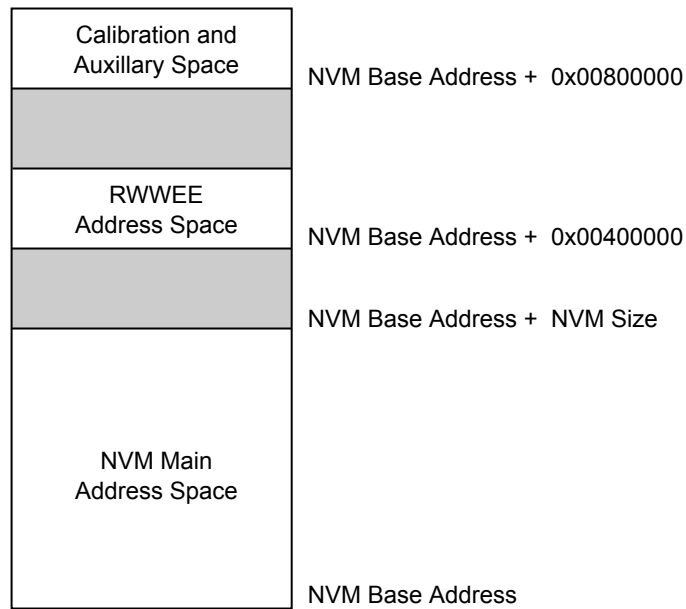
Row n	Page (n*4) + 3	Page (n*4) + 2	Page (n*4) + 1	Page (n*4) + 0
-------	----------------	----------------	----------------	----------------

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 28-3. NVM Memory Organization**

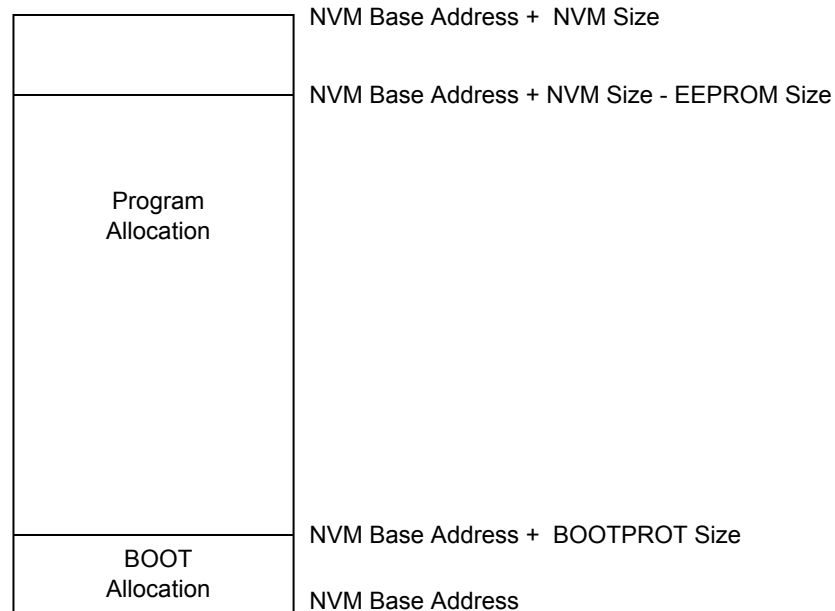


The lower rows in the NVM main address space can be allocated as a boot loader section by using the BOOTPROT fuses, and the upper rows can be allocated to EEPROM, as shown in the figure below.

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 28-4. EEPROM and Boot Loader Allocation**



**Related Links**

[Physical Memory Map](#) on page 35

### 28.6.3. 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 28-1. 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.

#### Related Links

[Physical Memory Map](#) on page 35

### 28.6.4. 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.

Read the [CTRLA](#) register description for more details.

The [CTRLB](#) register must be used to control the power reduction mode, read wait states, and the write mode.

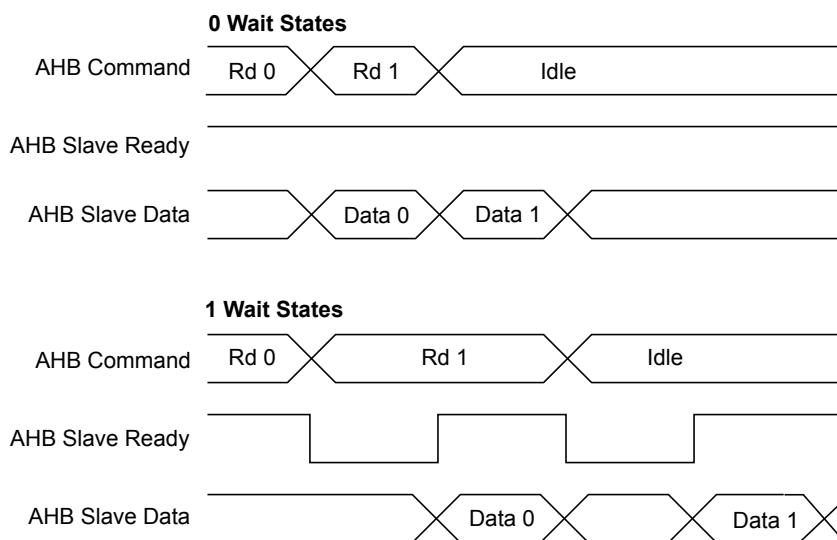
#### 28.6.4.1. NVM Read

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.

**Figure 28-5. Read Wait State Examples**



#### 28.6.4.2. RWWEE Read

Reading from the RWW EEPROM address space is performed via the AHB bus by addressing the RWWEE address space directly. Refer to the figures in [Memory Organization](#) on page 421 for details.

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.

#### 28.6.4.3. 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.

Internally, writes to the page buffer are on a 64-bit basis through the page buffer load data register (PBLDATA1 and PBLDATA0). The PBLDATA register is a holding register for writes to the same 64-bit page buffer section. Data within a 64-bit section can be written in any order. Crossing a 64-bit boundary will reset the PBLDATA register to all ones. The following example assumes startup from reset where the current address is 0 and PBLDATA is all ones. Only 64 bits of the page buffer are written at a time, but 128 bits are shown for reference.

##### Sequential 32-bit Write Example:

- 32-bit 0x1 written to address 0
  - Page buffer[127:0] = {0xFFFFFFFF\_FFFFFFFF, PBLDATA[63:32], 0x00000001}

- PBLDATA[63:0] = {PBLDATA[63:32], 0x00000001}
- 32-bit 0x2 written to address 1
  - Page buffer[127:0] = {0xFFFFFFFF\_FFFFFFFF, 0x00000002, PBLDATA[31:0]}
  - PBLDATA[63:0] = 0x00000002, PBLDATA[31:0]}
- 32-bit 0x3 written to address 2 (crosses 64-bit boundary)
  - Page buffer[127:0] = 0xFFFFFFFF\_00000003\_00000002\_00000001
  - PBLDATA[63:0] = 0xFFFFFFFF\_00000003

Random access writes to 32-bit words within the page buffer will overwrite the opposite word within the same 64-bit section with ones. In the following example, notice that 0x00000001 is overwritten with 0xFFFFFFFF from the third write due to the 64-bit boundary crossing. Only 64 bits of the page buffer are written at a time, but 128 bits are shown for reference.

#### Random Access 32-bit Write Example:

- 32-bit 0x1 written to address 2
  - Page buffer[127:0] = 0xFFFFFFFF\_00000001\_FFFFFFFF\_FFFFFFFF
  - PBLDATA[63:0] = 0xFFFFFFFF\_00000001
- 32-bit 0x2 written to address 1
  - Page buffer[127:0] = 0xFFFFFFFF\_00000001\_00000002\_FFFFFFFF
  - PBLDATA[63:0] = 0x00000002\_FFFFFFFF
- 32-bit 0x3 written to address 3
  - Page buffer[127:0] = 0x00000003\_FFFFFFFF\_00000002\_FFFFFFFF
  - PBLDATA[63:0] = 0x00000003\_0xFFFFFFFF

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.

#### 28.6.4.4. 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.

#### 28.6.4.5. 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.

#### 28.6.4.6. Lock and Unlock Region

These commands are used to lock and unlock regions as detailed in section [Region Lock Bits](#) on page 423.

#### 28.6.4.7. 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.

### 28.6.5. 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 28-2. Boot Loader Size**

BOOTPROT [2:0]	Rows Protected by BOOTPROT	Boot Loader Size in Bytes
7	None	0
6	2	512
5	4	1024
4	8	2048
3	16	4096
2	32	8192
1	64	16384
0	128	32768

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 28-3. 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

**Related Links**

[Physical Memory Map](#) on page 35

**28.6.6. 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 BODVDD when the security bit is set.

**Related Links**

[DSU - Device Service Unit](#) on page 59

**28.6.7. 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.

## 28.7. Register Summary

Offset	Name	Bit Pos.									
0x00	CTRLA	7:0							CMD[6:0]		
0x01		15:8							CMDEX[7:0]		
0x02 ... 0x03	Reserved										
0x04	CTRLB	7:0	MANW						RWS[3:0]		
0x05		15:8							SLEPPRM[1:0]		
0x06		23:16						CACHEDIS	READMODE[1:0]		
0x07		31:24									
0x08	PARAM	7:0							NVMP[7:0]		
0x09		15:8							NVMP[15:8]		
0x0A		23:16							RWWECP[3:0]	PSZ[2:0]	
0x0B		31:24							RWWECP[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
0x19		15:8									SB
0x1A ... 0x1B	Reserved										
0x1C	ADDR	7:0							ADDR[7:0]		
0x1D		15:8							ADDR[15:8]		
0x1E		23:16							ADDR[20:16]		
0x1F		31:24									
0x20	LOCK	7:0							LOCK[7:0]		
0x21		15:8							LOCK[15:8]		
0x22 ... 0x27	Reserved										
0x28	PBLDATA0	7:0							PBLDATA[7:0]		
0x29		15:8							PBLDATA[15:8]		
0x2A		23:16							PBLDATA[23:16]		
0x2B		31:24							PBLDATA[31:24]		



Offset	Name	Bit Pos.							
0x2C	PBLDATA1	7:0	PBLDATA[7:0]						
0x2D		15:8	PBLDATA[15:8]						
0x2E		23:16	PBLDATA[23:16]						
0x2F		31:24	PBLDATA[31:24]						

## 28.8. Register Description

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

Some registers 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.

## 28.8.1. 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 PROGE status bit will 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.

### 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.
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

CMD[6:0]	Group Configuration	Description
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.
0x47-0x7F	-	Reserved

## 28.8.2. 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.

Value	Name	Description
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.

### 28.8.3. 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.

#### 28.8.4. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x0C

**Reset:** 0x00

**Property:** PAC Write-Protection

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.

### 28.8.5. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x10

**Reset:** 0x00

**Property:** PAC Write-Protection

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.



## 28.8.6. 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.

## 28.8.7. 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 SLEEP<sub>PRM</sub> 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 SLEEP<sub>PRM</sub> set accordingly.

Value	Description
0	NVM is not in power reduction mode.
1	NVM is in power reduction mode.

## 28.8.8. 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	
Access				ADDR[20:16]					
Reset				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	ADDR[15:8]								
Reset	R/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	ADDR[7:0]								
Reset	R/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 20:0 – ADDR[20:0]: 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.

## 28.8.9. Lock Section

**Name:** LOCK  
**Offset:** 0x20  
**Reset:** 0XXXXX  
**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

In order 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.

## 28.8.10. Page Buffer Load Data 0

**Name:** PBLDATA0  
**Offset:** 0x28  
**Reset:** 0xFFFFFFFF  
**Property:** -

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

### Bits 31:0 – PBLDATA[31:0]: Page Buffer Load Data

The PBLDATA register is a holding register for partial AHB writes to the same 64-bit page buffer section. Page buffer loads are performed on a 64-bit basis.

This is a read only register.

## 28.8.11. Page Buffer Load Data 1

**Name:** PBLDATA1  
**Offset:** 0x2C  
**Reset:** 0xFFFFFFFF  
**Property:** -

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

**Bits 31:0 – PBLDATA[31:0]: Page Buffer Load Data (Bits 63:32)** Once the dimension element functions are supported the bit description must be updated to bit names becomes **PBLDATA[63:32]** in the register table.

The PBLDATA register is a holding register for partial AHB writes to the same 64-bit page buffer section. Page buffer loads are performed on a 64-bit basis.

This is a read only register.

## 29. PORT - I/O Pin Controller

### 29.1. 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 group can have up to 32 pins that can be configured and controlled individually or as a group. Each pin may either be used for general-purpose I/O under direct application control or 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).

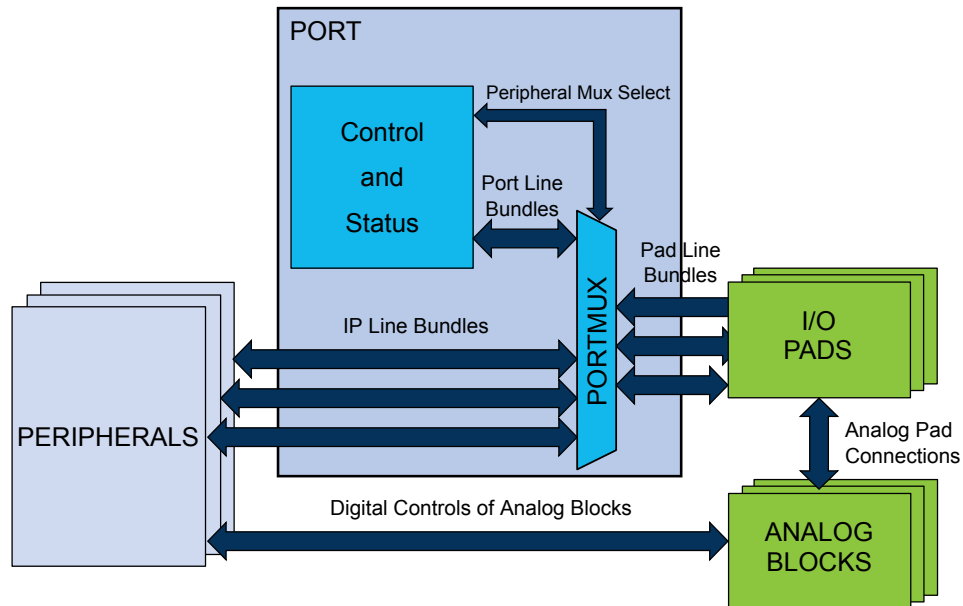
### 29.2. Features

- 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
- Input event:
  - Up to four input event pins for each PORT group
  - SET/CLEAR/TOGGLE event actions for each event input on output value of a pin
  - Can be output to pin
- Power saving using STANDBY mode
  - No access to configuration registers
  - Possible access to data registers (DIR, OUT or IN)



## 29.3. Block Diagram

Figure 29-1. PORT Block Diagram



## 29.4. Signal Description

Table 29-1. 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](#) on page 27

## 29.5. Product Dependencies

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

### 29.5.1. 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](#) on page 27

### 29.5.2. Power Management

During reset, all PORT lines are configured as inputs with input buffers, output buffers and pull disabled.

If the PORT peripheral is shut down, the latches in the pad will keep their current configuration in any sleep mode, such as the output value and pull settings. However, the PORT configuration registers and input synchronizers will lose their contents, and these will not be restored when PORT is powered up again. Therefore, user must reconfigure the PORT peripheral at power-up to ensure it is in a well-defined state before use.

The PORT will continue operating in any sleep mode where the selected module source clock is running because the selected module source clock is still running.

### 29.5.3. Clocks

The PORT bus clock (CLK\_PORT\_APB) can be enabled and disabled in the Main Clock module, and the default state of CLK\_PORT\_APB can be found in the *Peripheral Clock Masking* section in *MCLK – Main Clock*.

The PORT is fed by two different clocks: a CPU main clock, which allows the CPU to access the PORT through the low latency CPU local bus (IOBUS); an APB clock, which is a divided clock of 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 priority of IOBUS accesses is higher than event accesses and APB accesses. The EVSYS and APB will insert wait states in the event of concurrent PORT accesses.

The PORT input synchronizers use the CPU main clock so that the resynchronization delay is minimized with respect to the APB clock.

#### Related Links

[MCLK – Main Clock](#) on page 131

### 29.5.4. DMA

Not applicable.

### 29.5.5. Interrupts

Not applicable.

### 29.5.6. Events

The events of this peripheral are connected to the Event System.

#### Related Links

[EVSYS – Event System](#) on page 474

### 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

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.

When the CPU is halted in debug mode, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger.

#### Related Links

[PAC - Peripheral Access Controller](#) on page 46

### 29.5.9. Analog Connections

Analog functions are connected directly between the analog blocks and the I/O pads using analog buses. However, selecting an analog peripheral function for a given pin will disable the corresponding digital features of the pad.

### 29.5.10. CPU Local Bus

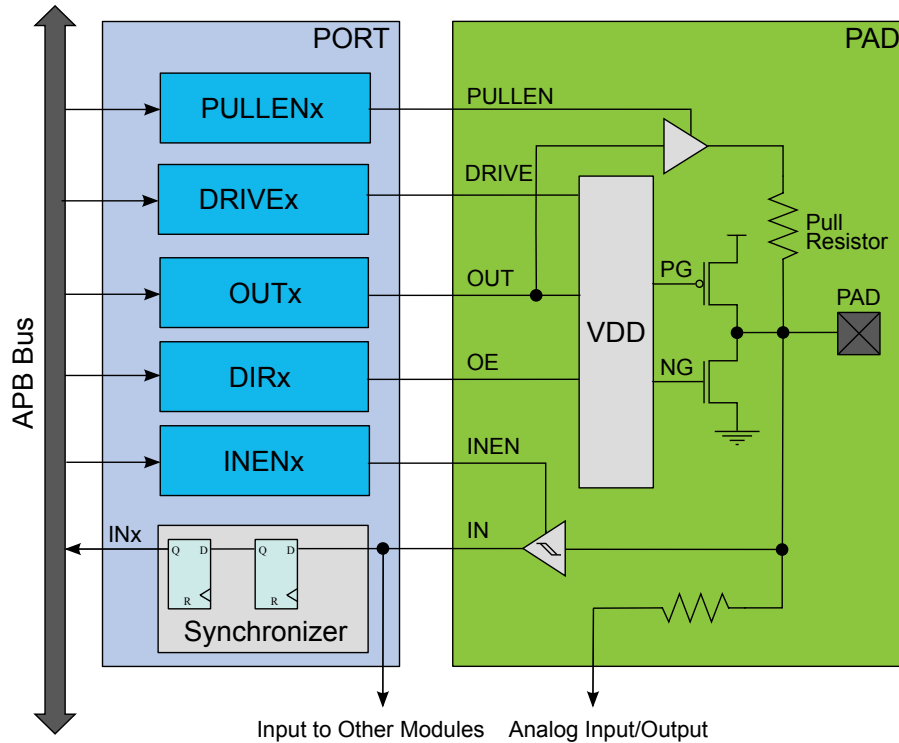
The CPU local bus (IOBUS) is an interface that connects the CPU directly to the PORT. It is a single-cycle bus interface, which does not support wait states. It supports 8-bit, 16-bit and 32-bit sizes.

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.

## 29.6. Functional Description

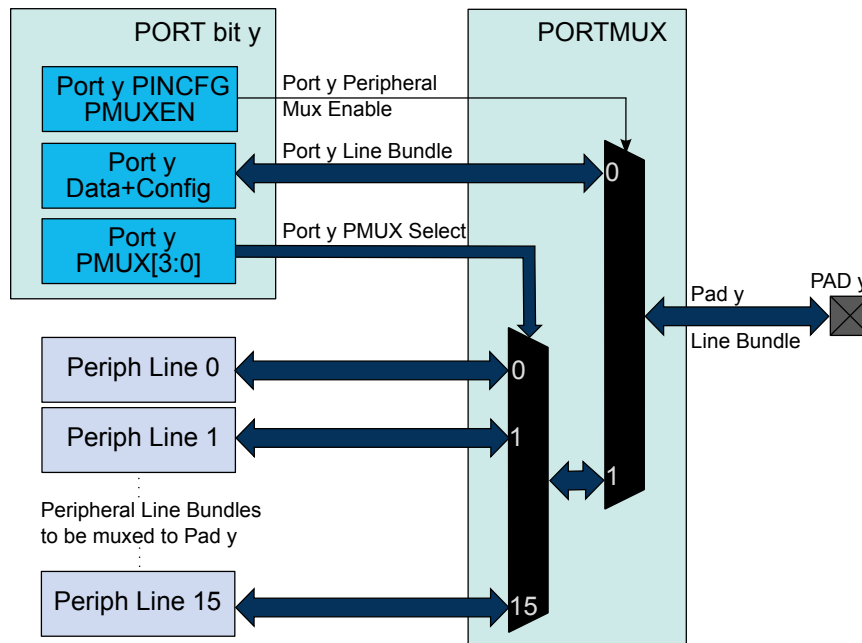
Figure 29-2. Overview of the PORT



### 29.6.1. Principle of Operation

Each pin 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 pin group, with increasing base addresses.

Figure 29-3. 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 pin 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 (PINCFG<sub>y</sub>) 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 are clocked only when system requires reading the input value. 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 (PINCFG<sub>y</sub>.INEN) is '0', the input value will not be sampled.

In PORT, the Peripheral Multiplexer Enable bit in the PINCFG<sub>y</sub> register (PINCFG<sub>y</sub>.PMUXEN) can be written to '1' to enable the connection between peripheral functions and individual I/O pins. The Peripheral Multiplexing (PMUX<sub>n</sub>) 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.

## 29.6.2. Basic Operation

### 29.6.2.1. 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.

### 29.6.2.2. Operation

Each I/O pin *y* can be controlled by the registers in PORT. Each pin group has its own set of PORT registers, the base address of the register set for pin *y* is at byte address PORT + ([*y*] \* 0x4). The index within that register set is [*y*].

To use pin number *y* 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.

To use pin *y* 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 (PINCFG<sub>y</sub>.INEN) is written to '1'. Refer to *I/O Multiplexing and Considerations* for details on pin configuration.

By default, the input synchronizer is clocked only when an input read is requested. This will delay the read operation by two CLK\_PORT cycles. To remove the delay, the input synchronizers for each pin 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 SAMPLING<sub>n</sub> bit field of the CTRL register, see CTRL.SAMPLING for details.

To use pin  $y$  as one of the available peripheral functions, the corresponding PMUXEN bit of the PINCFGy register must be '1'. The PINCFGy register for pin  $y$  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](#) on page 27

### 29.6.3. 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 29-2 Pin Configurations Summary](#) on page 450.

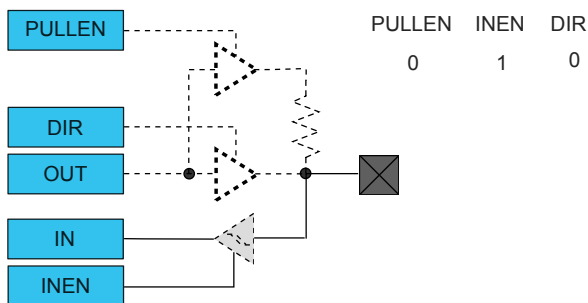
#### 29.6.3.1. Pin Configurations Summary

**Table 29-2. Pin Configurations Summary**

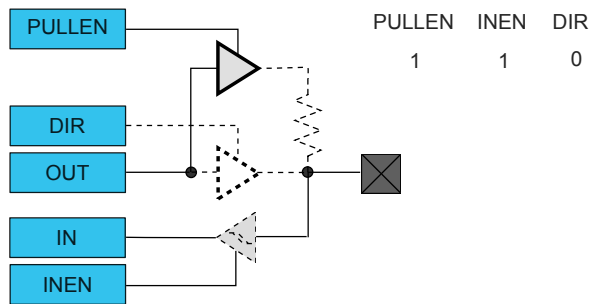
DIR	INEN	PULLEN	OUT	Configuration
0	0	0	X	Reset or analog I/O: all digital disabled
0	0	1	0	Pull-down; input disabled
0	0	1	1	Pull-up; input disabled
0	1	0	X	Input
0	1	1	0	Input with pull-down
0	1	1	1	Input with pull-up
1	0	X	X	Output; input disabled
1	1	X	X	Output; input enabled

#### 29.6.3.2. Input Configuration

**Figure 29-4. I/O configuration - Standard Input**



**Figure 29-5. I/O Configuration - Input with Pull**



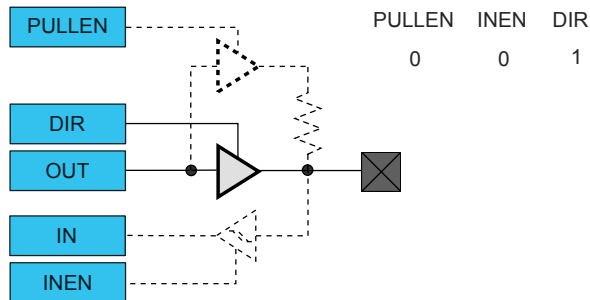
**Note:** When pull is enabled, the pull value is defined by the OUT value.

### 29.6.3.3. Totem-Pole Output

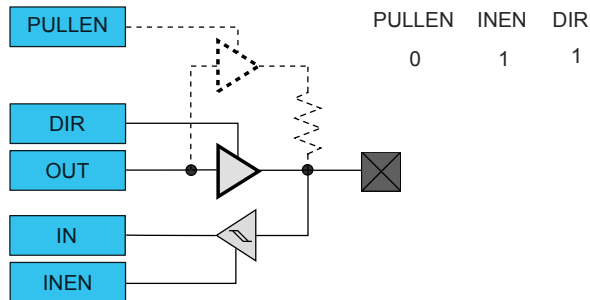
When configured for totem-pole (push-pull) output, the pin is driven low or high according to the corresponding bit setting in the OUT register. In this configuration there is no current limitation for sink or source other than what the pin is capable of. If the pin is configured for input, the pin will float if no external pull is connected.

**Note:** Enabling the output driver will automatically disable pull.

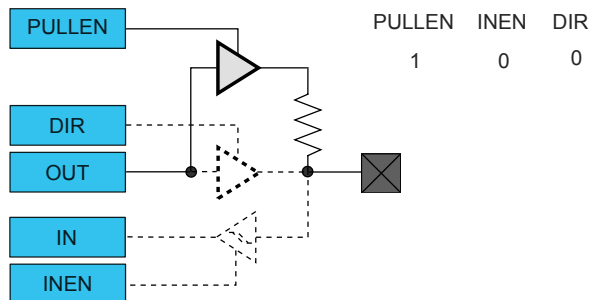
**Figure 29-6. I/O Configuration - Totem-Pole Output with Disabled Input**



**Figure 29-7. I/O Configuration - Totem-Pole Output with Enabled Input**



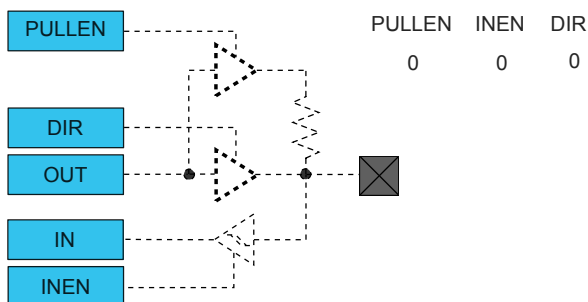
**Figure 29-8. I/O Configuration - Output with Pull**



### 29.6.3.4. Digital Functionality Disabled

Neither Input nor Output functionality are enabled.

**Figure 29-9. I/O Configuration - Reset or Analog I/O: Digital Output, Input and Pull Disabled**



### 29.6.4. Events

The PORT allows input events to control individual I/O pins. These input events are generated by the EVSYS module and can originate from a different clock domain than the PORT module.

The PORT can perform the following actions:

- Output (OUT): I/O pin will be set when the incoming event has a high level ('1') and cleared when the incoming event has a low-level ('0').
- Set (SET): I/O pin will be set when an incoming event is detected.
- Clear (CLR): I/O pin will be cleared when an incoming event is detected.
- Toggle (TGL): I/O pin will toggle when an incoming event is detected.

The event is output to pin without any internal latency. For SET, CLEAR and TOGGLE event actions, the action will be executed up to three clock cycles after a rising edge.

The event actions can be configured with the Event Action m bit group in the Event Input Control register (EVCTRL.EVACTm). Writing a '1' to a PORT Event Enable Input m of the Event Control register (EVCTRL.PORTEIm) enables the corresponding action on input event. Writing '0' to this bit disables the corresponding action on input event. Note that several actions can be enabled for incoming events. If several events are connected to the peripheral, any enabled action will be taken for any of the incoming events. Refer to *EVSYS – Event System*. for details on configuring the Event System.

Each event input can address one and only one I/O pin at a time. The selection of the pin is indicated by the PORT Event Pin Identifier of the Event Input Control register (EVCTR.PIDn). On the other hand, one I/O pin can be addressed by up to four different input events. To avoid action conflict on the output value of the register (OUT) of this particular I/O pin, only one action is performed according to the table below.

Note that this truth table can be applied to any SET/CLR/TGL configuration from two to four active input events.

**Table 29-3. Priority on Simultaneous SET/CLR/TGL Event Actions**

EVACT0	EVACT1	EVACT2	EVACT3	Executed Event Action
SET	SET	SET	SET	SET
CLR	CLR	CLR	CLR	CLR
All Other Combinations				TGL

Be careful when the event is output to pin. Due to the fact the events are received asynchronously, the I/O pin may have unpredictable levels, depending on the timing of when the events are received. When



several events are output to the same pin, the lowest event line will get the access. All other events will be ignored.

#### Related Links

[EVSYS – Event System](#) on page 474

### 29.6.5. 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)
- EVSYS through four asynchronous input events

The following priority is adopted:

1. ARM® CPU IOBUS (No wait tolerated)
2. APB
3. EVSYS input events

For input events that require different actions on the same I/O pin, refer to [Events](#) on page 452.

## 29.7. Register Summary

The I/O pins are assembled in pin groups with up to 32 pins. Group 0 consists of the PA pins, and group 1 is for the PB pins, etc. Each pin group has its own PORT registers. 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.

Offset	Name	Bitpos.							
0x00	DIR	7:0	DIR[7:0]						
0x01		15:8	DIR[15:8]						
0x02		23:16	DIR[23:16]						
0x03		31:24	DIR[31:24]						
0x04	DIRCLR	7:0	DIRCLR[7:0]						
0x05		15:8	DIRCLR[15:8]						
0x06		23:16	DIRCLR[23:16]						
0x07		31:24	DIRCLR[31:24]						
0x08	DIRSET	7:0	DIRSET[7:0]						
0x09		15:8	DIRSET[15:8]						
0x0A		23:16	DIRSET[23:16]						
0x0B		31:24	DIRSET[31:24]						
0x0C	DIRTGL	7:0	DIRTGL[7:0]						
0x0D		15:8	DIRTGL[15:8]						
0x0E		23:16	DIRTGL[23:16]						
0x0F		31:24	DIRTGL[31:24]						
0x10	OUT	7:0	OUT[7:0]						
0x11		15:8	OUT[15:8]						
0x12		23:16	OUT[23:16]						
0x13		31:24	OUT[31:24]						

Offset	Name	Bitpos.							
0x14	OUTCLR	7:0	OUTCLR[7:0]						
0x15		15:8	OUTCLR[15:8]						
0x16		23:16	OUTCLR[23:16]						
0x17		31:24	OUTCLR[31:24]						
0x18	OUTSET	7:0	OUTSET[7:0]						
0x19		15:8	OUTSET[15:8]						
0x1A		23:16	OUTSET[23:16]						
0x1B		31:24	OUTSET[31:24]						
0x1C	OUTTGL	7:0	OUTTGL[7:0]						
0x1D		15:8	OUTTGL[15:8]						
0x1E		23:16	OUTTGL[23:16]						
0x1F		31:24	OUTTGL[31:24]						
0x20	IN	7:0	IN[7:0]						
0x21		15:8	IN[15:8]						
0x22		23:16	IN[23:16]						
0x23		31:24	IN[31:24]						
0x24	CTRL	7:0	SAMPLING[7:0]						
0x25		15:8	SAMPLING[15:8]						
0x26		23:16	SAMPLING[23:16]						
0x27		31:24	SAMPLING[31:24]						
0x28	WRCONFIG	7:0	PINMASK[7:0]						
0x29		15:8	PINMASK[15:8]						
0x2A		23:16		DRVSTR			PULLEN	INEN	PMUXEN
0x2B		31:24	HWSEL	WRPINCFG		WRPMUX	PMUX[3:0]		
0x2C	EVCTRL	7:0	PORTEI0	EVACT0[1:0]		PID0[4:0]			
0x2D		15:8	PORTEI1	EVACT1[1:0]		PID1[4:0]			
0x2E		23:16	PORTEI2	EVACT2[1:0]		PID2[4:0]			
0x2F		31:24	PORTEI3	EVACT3[1:0]		PID3[4:0]			
0x30	PMUX0	7:0	PMUXO[3:0]			PMUXE[3:0]			
0x31	PMUX1	7:0	PMUXO[3:0]			PMUXE[3:0]			
...	...	...							
0x3E	PMUX14	7:0	PMUXO[3:0]			PMUXE[3:0]			
0x3F	PMUX15	7:0	PMUXO[3:0]			PMUXE[3:0]			
0x40	PINCFG0	7:0		DRVSTR			PULLEN	INEN	PMUXEN
0x41	PINCFG1	7:0		DRVSTR			PULLEN	INEN	PMUXEN
...	...	...							
0x5E	PINCFG30	7:0		DRVSTR			PULLEN	INEN	PMUXEN
0x5F	PINCFG31	7:0		DRVSTR			PULLEN	INEN	PMUXEN

## 29.8. Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write-protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to [Register Access Protection](#) on page 447.

### 29.8.1. Data Direction

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.

**Name:** DIR

**Offset:** 0x00 + n\*0x80 [n=0,1]

**Reset:** 0x00000000

**Property:** PAC Write-Protection

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

#### Bits 31:0 – DIR[31:0]: Port Data Direction

These bits set the data direction for the individual I/O pins in the pin group.

Value	Description
0	The corresponding I/O pin in the pin group is configured as an input.
1	The corresponding I/O pin in the pin group is configured as an output.

## 29.8.2. Data Direction Clear

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.

**Name:** DIRCLR

**Offset:** 0x04 + n\*0x80 [n=0..2]

**Reset:** 0x00000000

**Property:** PAC Write-Protection

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

### 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 group will keep its configuration.
1	The corresponding I/O pin in the pin group is configured as input.

### 29.8.3. Data Direction Set

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.

**Name:** DIRSET  
**Offset:** 0x08 + n\*0x80 [n=0..2]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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

#### 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 group will keep its configuration.
1	The corresponding I/O pin in the pin group is configured as an output.

### 29.8.4. Data Direction Toggle

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.

**Name:** DIRTGL  
**Offset:** 0x0C + n\*0x80 [n=0..2]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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

#### 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 group will keep its configuration.
1	The direction of the corresponding I/O pin is toggled.

### 29.8.5. Data Output Value

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.

**Name:** OUT

**Offset:** 0x10 + n\*0x80 [n=0..2]

**Reset:** 0x00000000

**Property:** PAC Write-Protection

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

#### 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.

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.

### 29.8.6. Data Output Value Clear

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.

**Name:** OUTCLR  
**Offset:** 0x14 + n\*0x80[n=0..2]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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

#### 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 set the input pull direction to an internal pull-down.

Value	Description
0	The corresponding I/O pin in the 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.



### 29.8.7. Data Output Value Set

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.

**Name:** OUTSET  
**Offset:** 0x18 + n\*0x80 [n=0..2]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

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

#### 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.

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.

### 29.8.8. Data Output Value Toggle

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.

**Name:** OUTTGL

**Offset:** 0x1C + n\*0x80 [n=0..2]

**Reset:** 0x00000000

**Property:** PAC Write-Protection

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

#### 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 group will keep its configuration.
1	The corresponding OUT bit value is toggled.

## 29.8.9. Data Input Value

**Name:** IN  
**Offset:**  $0x20 + n \cdot 0x80$  [ $n=0..2$ ]  
**Reset:** 0x00000000  
**Property:** -

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

### Bits 31:0 – IN[31:0]: PORT Data Input Value

These bits are cleared when the corresponding I/O pin input sampler detects a logical low level on the input pin.

These bits are set when the corresponding I/O pin input sampler detects a logical high level on the input pin.

## 29.8.10. Control

**Name:** CTRL  
**Offset:** 0x24 + n\*0x80 [n=0..2]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	SAMPLING[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
	SAMPLING[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
	SAMPLING[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
	SAMPLING[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 – 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	The I/O pin input synchronizer is disabled.
1	The I/O pin input synchronizer is enabled.

### 29.8.11. Write Configuration

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.

**Name:** WRCONFIG

**Offset:** 0x28 + n\*0x80 [n=0..2]

**Reset:** 0x00000000

**Property:** PAC Write-Protection

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-pin 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 pin group will be updated.

### 29.8.12. Event Input Control

There is one 32-bit Event Input Control register for each PORT group. Each byte of this register addresses a group of 32-bit I/O lines. The x denotes the number of the PORT group.

**Name:** EVCTRL  
**Offset:** 0x2C + n\*0x80 [n=0..2]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	PORTEI3		EVACT3[1:0]		PID3[4:0]			
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
	PORTEI2		EVACT2[1:0]		PID2[4:0]			
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
	PORTEI1		EVACT1[1:0]		PID1[4:0]			
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
	PORTEI0		EVACT0[1:0]		PID0[4:0]			
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 31,23,15,7 – PORTEIx: PORT Event Input x Enable [x = 3..0]

Value	Description
0	The event action x (EVACTx) will not be triggered on any incoming event.
1	The event action x (EVACTx) will be triggered on any incoming event.

#### Bits 30:29, 22:21,14:13,6:5 – EVACTx: PORT Event Action x [x = 3..0]

These bits define the event action the PORT will perform on event input x. See also [Table 29-4 PORT Event x Action \( x = \[3..0\] \)](#) on page 469.

#### Bits 28:24,20:16,12:8,4:0 – PIDx: PORT Event Pin Identifier x [x = 3..0]

These bits define the I/O pin on which the event action will be performed, according to [Table 29-5 PORT Event x Pin Identifier \( x = \[3..0\] \)](#) on page 469.



**Table 29-4. PORT Event x Action ( x = [3..0] )**

Value	Name	Description
0x0	OUT	Output register of pin will be set to level of event.
0x1	SET	Set output register of pin on event.
0x2	CLR	Clear output register of pin on event.
0x3	TGL	Toggle output register of pin on event.

**Table 29-5. PORT Event x Pin Identifier ( x = [3..0] )**

Value	Name	Description
0x0	PIN0	Event action to be executed on PIN 0.
0x1	PIN1	Event action to be executed on PIN 1.
...	...	...
0x31	PIN31	Event action to be executed on PIN 31.

### 29.8.13. Peripheral Multiplexing n

There are up to 16 Peripheral Multiplexing registers in each group, one for every set of two subsequent I/O lines. The n denotes the number of the set of I/O lines, while the x denotes the number of the group.

**Name:** PMUXn

**Offset:** 0x30 + n [n=0..15] + m\*0x80 [m=0..2]

**Reset:** 0x00000000

**Property:** PAC Write-Protection

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

#### Bits 7:4 – PMUXO[3:0]: Peripheral Multiplexing Odd

These bits select the peripheral function for odd-numbered pins ( $2*n + 1$ ) of a PORT group, if the corresponding PINCFGy.PMUXEN bit is '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 Even

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

PMUXE[3:0]	Name	Description
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

### 29.8.14. Pin Configuration y

There are up to 32 Pin Configuration registers in each group, one for each I/O line. The n denotes the number of the I/O line, while the m denotes the number of the group.

**Name:** PINCFGy

**Offset:** 0x40 + n\*0x1 [n=0..31] + m\*0x80 [m=0..2]

**Reset:** 0x00

**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
		DRVSTR				PULLEN	INEN	PMUXEN
Access		R/W				R/W	R/W	R/W
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 PINCFGy.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.

## 30. EVSYS – Event System

### 30.1. 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.

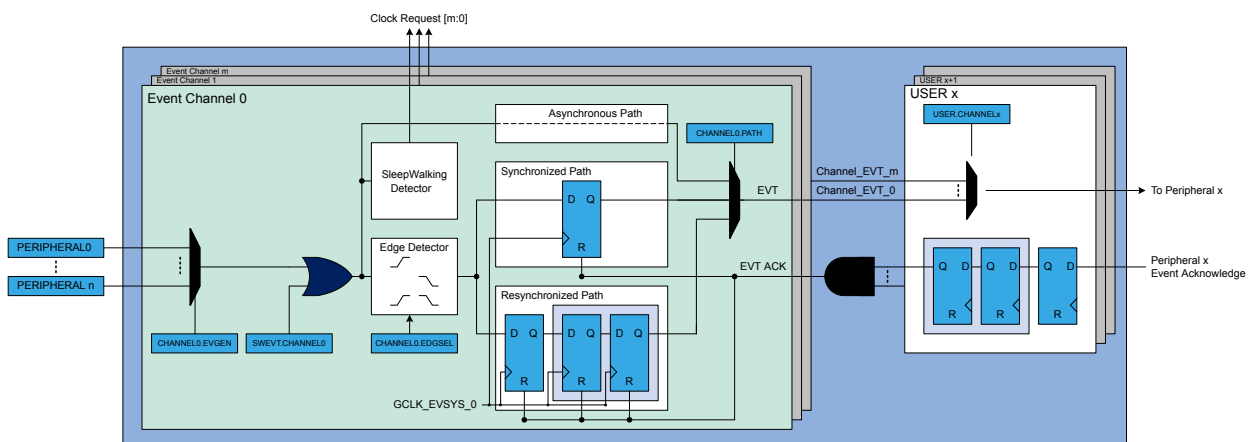
Communication is made without CPU intervention and without consuming system resources such as bus or RAM bandwidth. This reduces the load on the CPU and other system resources, compared to a traditional interrupt-based system.

### 30.2. Features

- 12 configurable event channels, where each channel can:
  - Be connected to any event generator.
  - Provide a pure asynchronous, resynchronized or synchronous path
- 87 event generators.
- 47 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.

### 30.3. Block Diagram

Figure 30-1. Event System Block Diagram



## 30.4. Signal Description

Not applicable.

## 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

Not applicable.

### 30.5.2. 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. The generic clock for the channel (GCLK\_EVSYS\_CHANNEL\_n) will be restarted if that channel uses a synchronized path or a resynchronized path. It does not need to wake the system from sleep.

#### Related Links

[PM – Power Manager](#) on page 162

### 30.5.3. 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](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

### 30.5.4. DMA

Not applicable.

### 30.5.5. 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](#) on page 41

### 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 optionally write-protected by the Peripheral Access Controller (PAC), except the following:

- Channel Status (CHSTATUS)
- Interrupt Flag Status and Clear register (INTFLAG)

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

When the CPU is halted in debug mode, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger.

### 30.5.9. Analog Connections

Not applicable.

## 30.6. Functional Description

### 30.6.1. 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.

A channel path can be configured in asynchronous, synchronous or re-synchronized mode of operation. The mode of operation must be selected based on the requirements of the application.

When using synchronous or resynchronized path, the Event System includes options to transfer events to users when rising, falling or both edges are detected on on event generators.

For further details, refer to "[Channel Path](#) on page 477" of this chapter.

### 30.6.2. Basic Operation

#### 30.6.2.1. Initialization

Before enabling events routing within the system, the Event Users Multiplexer and Event Channels must be configured. The Event Users Multiplexer must be configured first.

For further details about the event user multiplexer configuration, refer to "[User Multiplexer Setup](#) on page 476".

For further details about the event channels configuration, refer to "[Event System Channel](#) on page 477".

#### 30.6.2.2. 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 A register (CTRLA.SWRST). All registers in the EVSYS will be reset to their initial state and all ongoing events will be canceled.

Refer to [CTRLA.SWRST](#) register for details.

#### 30.6.2.3. 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 [Figure 30-1 Event System Block Diagram](#) on page 474. The channel is selected with the Channel bit group in the User register (USERm.CHANNEL).

The user multiplexer must always be configured before the channel. A list of all user multiplexers is found in the User (USERm) register description.

#### Related Links

[USERm](#) on page 493

#### 30.6.2.4. Event System Channel

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. A channel block diagram is shown in [Figure 30-1 Event System Block Diagram](#) on page 474.

#### 30.6.2.5. Event Generators

Each event channel can receive the events from all event generators. All event generators are listed in the Event Generator bit field in the Channel n register (CHANNELn.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 (CHANNELn.EVGEN). By default, the channels are not connected to any event generators (ie, CHANNELn.EVGEN = 0)

#### 30.6.2.6. Channel Path

There are three different ways to propagate the event from an event generator:

- Asynchronous path
- Synchronous path
- Resynchronized path

The path is decided by writing to the Path Selection bit group of the Channel register (CHANNELn.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. The edge detection is not required and must be disabled by software. Each peripheral event user has to select which event edge must trigger internal actions. For further details, refer to each peripheral chapter description.

##### Synchronous Path

The synchronous path should be used when the event generator and the event channel share the same generator for the generic clock. If they do not share the same clock, a logic change from the event generator to the event channel might not be detected in the channel, which means that the event will not be propagated to the event user. 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 busy n bit in the Channel Status register (CHSTATUS.CHBUSYn) are also updated and available for use.

## Resynchronized Path

The resynchronized path are used when the event generator and the event channel do not share the same generator for the generic clock. 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 busy n bits in the Channel Status register (CHSTATUS.CHBUSYn) are also updated and available for use.

### Related Links

[GCLK - Generic Clock Controller](#) on page 114

#### 30.6.2.7. Edge Detection

When synchronous or resynchronized paths are used, edge detection must be enabled. The event system can execute 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 of the Channel register (CHANNELn.EDGSEL).

#### 30.6.2.8. Event Latency

An event from an event generator is propagated to an event user with different latency, depending on event channel configuration.

- Asynchronous Path: The maximum routing latency of an external event is related to the internal signal routing and it is device dependent.
- Synchronous Path: The maximum routing latency of an external event is one GCLK\_EVSYS\_CHANNEL\_n clock cycle.
- Resynchronized Path: The maximum routing latency of an external event is three GCLK\_EVSYS\_CHANNEL\_n clock cycles.

The maximum propagation latency of a user event to the peripheral clock core domain is three peripheral clock cycles.

The event generators, event channel and event user clocks ratio must be selected in relation with the internal event latency constraints. Events propagation or event actions in peripherals may be lost if the clock setup violates the internal latencies.

#### 30.6.2.9. The Overrun Channel n Interrupt

The Overrun Channel n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVRn) 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 synchronous or resynchronized paths. In the case of asynchronous path, the INTFLAG.OVRn is always read as zero.

#### 30.6.2.10. The Event Detected Channel n Interrupt

The Event Detected Channel n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.EVDn) 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 synchronous or resynchronized paths. In the case of asynchronous path, the INTFLAG.EVDn is always zero.

#### 30.6.2.11. 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.

#### 30.6.2.12. Software Event

A software event can be initiated on a channel by setting the Channel n bit in the Software Event register (SWEVT.CHANNELn) to '1'. Then the software event can be serviced as any event generator; i.e., when the bit is set to '1', an event will be generated on the respective channel.

### 30.6.3. Interrupts

The EVSYS has the following interrupt sources:

- Overrun Channel n interrupt (OVRn): for details, refer to [The Overrun Channel n Interrupt](#) on page 478.
- Event Detected Channel n interrupt (EVDn): for details, refer to [The Event Detected Channel n Interrupt](#) on page 478.

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](#) on page 486 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

[Sleep Mode Controller](#) on page 164

#### 30.6.4. Sleep Mode Operation

The EVSYS can generate interrupts to wake up the device from any sleep mode.

To be able to run in standby, the Run in Standby bit in the Channel register (CHANNELn.RUNSTDBY) must be set to '1'. When the Generic Clock On Demand bit in Channel register (CHANNELn.ONDEMAND) is set to '1' and the event generator is detected, the event channel will request its clock (GCLK\_EVSYS\_CHANNEL\_n). The event latency for a resynchronized channel path will increase by two GCLK\_EVSYS\_CHANNEL\_n clock (i.e., up to five GCLK\_EVSYS\_CHANNEL\_n clock cycles).

A channel will behave differently in different sleep modes regarding to CHANNELn.RUNSTDBY and CHANNELn.ONDEMAND, as shown in the table below:

**Table 30-1. Event Channel Sleep Behavior**

CHANNELn.ONDEMAND	CHANNELn.RUNSTDBY	Sleep Behavior
0	0	Only run in IDLE sleep modes if an event must be propagated. Disabled in STANDBY sleep mode.
0	1	Always run in IDLE and STANDBY sleep modes.
1	0	Only run in IDLE sleep modes if an event must be propagated. Disabled in STANDBY sleep mode. Two GCLK_EVSYN_n latency added in RESYNC path before the event is propagated internally.
1	1	Always run in IDLE and STANDBY sleep modes. Two GCLK_EVSYN_n latency added in RESYNC path before the event is propagated internally.

## 30.7. Register Summary

### 30.7.1. Common Registers

Offset	Name	Bit Pos.									
0x00	CTRLA	7:0								SWRST	
0x01..0x0B	Reserved										
0x0C	CHSTATUS	7:0	USRRDY7	USRRDY6	USRRDY5	USRRDY4	USRRDY3	USRRDY2	USRRDY1	USRRDY0	
0x0D		15:8					USRRDY11	USRRDY10	USRRDY9	USRRDY8	
0x0E		23:16	CHBUSY7	CHBUSY6	CHBUSY5	CHBUSY4	CHBUSY3	CHBUSY2	CHBUSY1	CHBUSY0	
0x0F		31:24					CHBUSY11	CHBUSY10	CHBUSY9	CHBUSY8	
0x10	INTENCLR	7:0	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0	
0x11		15:8					OVR11	OVR10	OVR9	OVR8	
0x12		23:16	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0	
0x13		31:24					EVD11	EVD10	EVD9	EVD8	
0x14	INTENSET	7:0	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0	
0x15		15:8					OVR11	OVR10	OVR9	OVR8	
0x16		23:16	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0	
0x17		31:24					EVD11	EVD10	EVD9	EVD9	
0x18	INTFLAG	7:0	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0	
0x19		15:8					OVR11	OVR10	OVR9	OVR8	
0x1A		23:16	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0	
0x1B		31:24					EVD11	EVD10	EVD9	EVD9	
0x1C	SWEVT	7:0	CHANNEL[7:0]								
0x1D		15:8						CHANNEL[11:8]			
0x1E		23:16									
0x1F		31:24									

### 30.7.2. CHANNELn

Offset	Name	Bit Pos.							
0x20 + 0x4*n	CHANNELn on page 488	7:0	EVGEN[7:0]						
0x21 + 0x4*n		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]	PATH[1:0]	
0x22 + 0x4*n		23:16							
0x23 + 0x4*n		31:24							

### 30.7.3. USERm

Offset	Name	Bit Pos.							
0x80 + 0x4*m	USERm on page 493	7:0	CHANNEL[7:0]						
0x81 + 0x4*m		15:8							
0x82 + 0x4*m		23:16							
0x83 + 0x4*m		31:24							

## 30.8. 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.

Refer to [Register Access Protection](#) on page 476 and *PAC - Peripheral Access Controller*.

#### Related Links

[PAC - Peripheral Access Controller](#) on page 46

### 30.8.1. Control

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

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

#### **Bit 0 – SWRST: Software Reset**

Writing '0' to this bit has no effect.

Writing '1' to this bit resets all registers in the EVSYS to their initial state.

## 30.8.2. Channel Status

**Name:** CHSTATUS  
**Offset:** 0x0C  
**Reset:** 0x000000FF  
**Property:** –

Bit	31	30	29	28	27	26	25	24
					CHBUSY11	CHBUSY10	CHBUSY9	CHBUSY8
Access					R	R	R	R
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
	CHBUSY7	CHBUSY6	CHBUSY5	CHBUSY4	CHBUSY3	CHBUSY2	CHBUSY1	CHBUSY0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
					USRRDY11	USRRDY10	USRRDY9	USRRDY8
Access					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	USRRDY7	USRRDY6	USRRDY5	USRRDY4	USRRDY3	USRRDY2	USRRDY1	USRRDY0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

### Bits 27:16 – CHBUSYn: Channel Busy n [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 11:0 – USRRDYn: User Ready for Channel n [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.

### 30.8.3. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x10

**Reset:** 0x00000000

**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
					EVD11	EVD10	EVD9	EVD8
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
Access	R/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
					OVR11	OVR10	OVR9	OVR8
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
Access	R/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 27:16 – EVDn: Event Detected Channel n Interrupt Enable [n = 11..0]

Writing '0' to this bit has no effect.

Writing '1' 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 11:0 – OVRn: Overrun Channel n Interrupt Enable[n = 11..0]

Writing '0' to this bit has no effect.

Writing '1' 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.



### 30.8.4. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x14

**Reset:** 0x00000000

**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
					EVD11	EVD10	EVD9	EVD8
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
Access	R/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
					OVR11	OVR10	OVR9	OVR8
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
Access	R/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 27:16 – EVDn: Event Detected Channel n Interrupt Enable [n = 11..0]

Writing '0' to this bit has no effect.

Writing '1' 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 11:0 – OVRn: Overrun Channel n Interrupt Enable [n = 11..0]

Writing '0' to this bit has no effect.

Writing '1' to this bit will set 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.

### 30.8.5. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** –

Bit	31	30	29	28	27	26	25	24
					EVD11	EVD10	EVD9	EVD8
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
	EVD7	EVD6	EVD5	EVD4	EVD3	EVD2	EVD1	EVD0
Access	R/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
					OVR11	OVR10	OVR9	OVR8
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	OVR7	OVR6	OVR5	OVR4	OVR3	OVR2	OVR1	OVR0
Access	R/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 27:16 – EVDn: Event Detected Channel n Interrupt Enable [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 '1'.

When the event channel path is asynchronous, the EVDn interrupt flag will not be set.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Event Detected Channel n interrupt flag.

#### Bits 11:0 – OVRn: Overrun Channel n Interrupt Enable [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.OVRn is '1'.

When the event channel path is asynchronous, the OVRn interrupt flag will not be set.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Overrun Detected Channel n interrupt flag.

### 30.8.6. Software Event

**Name:** SWEVT  
**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
Access								
Reset								
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	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

#### Bits 11:0 – CHANNELn: Channel n Software [n=11..0] Selection

Writing '0' to this bit has no effect.

Writing '1' to this bit will trigger a software event for the channel n.

These bits will always return zero when read.

### 30.8.7. Channel

This register allows the user to configure channel n. To write to this register, do a single, 32-bit write of all the configuration data.

**Name:** CHANNELn  
**Offset:** 0x20+n\*0x4 [n=0..11]  
**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
Access								
Reset								
Bit	15	14	13	12	11	10	9	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
Access	R/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 – ONDEMAND: Generic Clock On Demand

Value	Description
0	Generic clock for a channel is always on, if the channel is configured and generic clock source is enabled.
1	Generic clock is requested on demand while an event is handled

#### Bit 14 – RUNSTDBY: Run in Standby

This bit is used to define the behavior during standby sleep mode.

Value	Description
0	The channel is disabled in standby sleep mode.
1	The channel is not stopped in standby sleep mode and depends on the CHANNEL.ONDEMAND

#### Bits 11:10 – 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.

Value	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
0x2	FALLING_EDGE	Event detection only on the falling edge of the signal from the event generator
0x3	BOTH_EDGES	Event detection on rising and falling edges of the signal from the event generator

#### Bits 9:8 – PATH[1:0]: Path Selection

These bits are used to choose which path will be used by the selected channel.

The path choice can be limited by the channel source, see the table in [USERm](#) on page 493.

Value	Name	Description
0x0	SYNCHRONOUS	Synchronous path
0x1	RESYNCHRONIZED	Resynchronized path
0x2	ASYNCHRONOUS	Asynchronous path
0x3	-	Reserved

#### Bits 7:0 – EVGEN[7:0]: Event Generator

These bits are used to choose the event generator to connect to the selected channel.

Value	Event Generator	Description
0x00	NONE	No event generator selected
0x01	OSCCTRL FAIL	XOSC Clock Failure
0x02	OSC32KCTRL FAIL	XOSC32K Clock Failure
0x03	RTC CMP0	Compare 0 (mode 0 and 1) or Alarm 0 (mode 2)
0x04	RTC CMP1	Compare 1
0x05	RTC OVF	Overflow
0x06	RTC PER0	Period 0
0x07	RTC PER1	Period 1
0x08	RTC PER2	Period 2
0x09	RTC PER3	Period 3
0x0A	RTC PER4	Period 4
0x0B	RTC PER5	Period 5
0x0C	RTC PER6	Period 6
0x0D	RTC PER7	Period 7
0x0E	EIC EXTINT0	External Interrupt 0

Value	Event Generator	Description
0x0F	EIC EXTINT1	External Interrupt 1
0x10	EIC EXTINT2	External Interrupt 2
0x11	EIC EXTINT3	External Interrupt 3
0x12	EIC EXTINT4	External Interrupt 4
0x13	EIC EXTINT5	External Interrupt 5
0x14	EIC EXTINT6	External Interrupt 6
0x15	EIC EXTINT7	External Interrupt 7
0x16	EIC EXTINT8	External Interrupt 8
0x17	EIC EXTINT9	External Interrupt 9
0x18	EIC EXTINT10	External Interrupt 10
0x19	EIC EXTINT11	External Interrupt 11
0x1A	EIC EXTINT12	External Interrupt 12
0x1B	EIC EXTINT13	External Interrupt 13
0x1C	EIC EXTINT14	External Interrupt 14
0x1D	EIC EXTINT15	External Interrupt 15
0x1E	TSENS WINMON	Window Monitor
0x1F	DMAC CH0	Channel 0
0x20	DMAC CH1	Channel 1
0x21	DMAC CH2	Channel 2
0x22	DMAC CH3	Channel 3
0x23	TCC0 OVF	Overflow
0x24	TCC0 TRG	Trig
0x25	TCC0 CNT	Counter
0x26	TCC0 MC0	Match/Capture 1
0x27	TCC0 MC1	Match/Capture 1
0x28	TCC0 MC2	Match/Capture 2
0x29	TCC0 MC3	Match/Capture 3
0x2A	TCC1 OVF	Overflow
0x2B	TCC1 TRG	Trig
0x2C	TCC1 CNT	Counter
0x2D	TCC1 MC0	Match/Capture 0
0x2E	TCC1 MC1	Match/Capture 1
0x2F	TCC2 OVF	Overflow

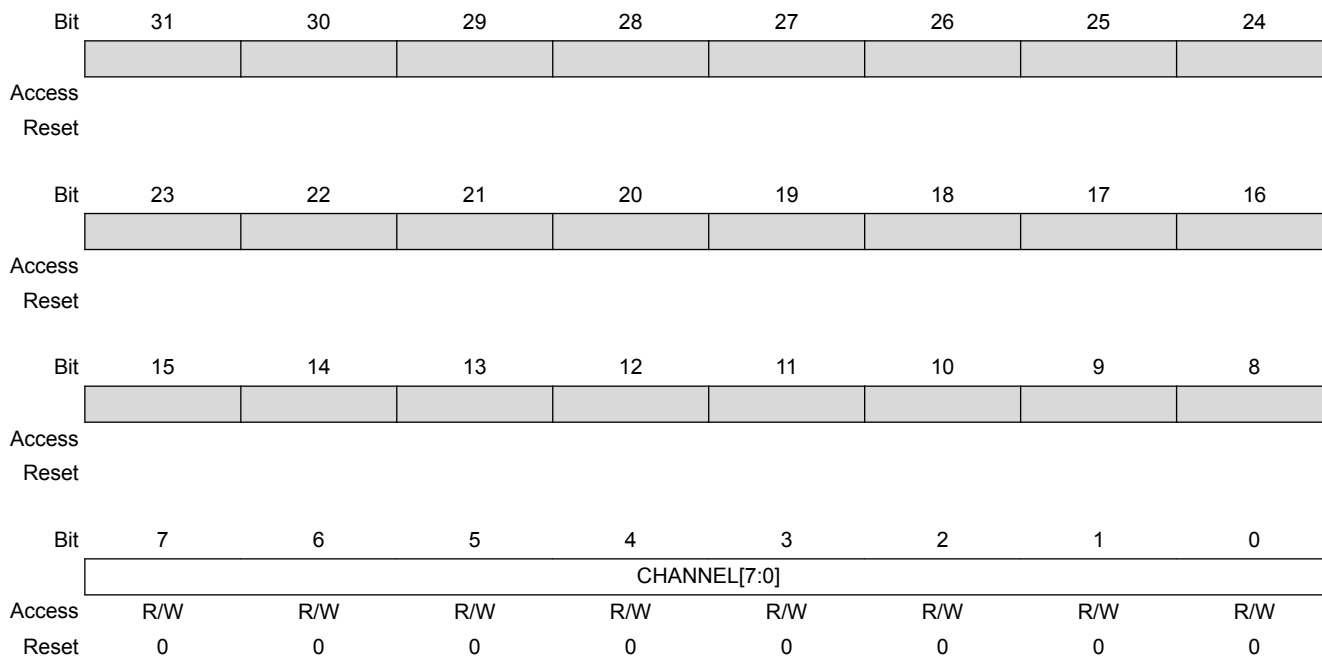
Value	Event Generator	Description
0x30	TCC2 TRG	Trig
0x31	TCC2 CNT	Counter
0x32	TCC2 MC0	Match/Capture 0
0x33	TCC2 MC1	Match/Capture 1
0x34	TC0 OVF	Overflow/Underflow
0x35	TC0 MC0	Match/Capture 0
0x36	TC0 MC1	Match/Capture 1
0x37	TC1 OVF	Overflow/Underflow
0x38	TC1 MC0	Match/Capture 0
0x39	TC1 MC1	Match/Capture 1
0x3A	TC2 OVF	Overflow/Underflow
0x3B	TC2 MC1	Match/Capture 0
0x3C	TC2 MC0	Match/Capture 1
0x3D	TC3 OVF	Overflow/Underflow
0x3E	TC3 MC0	Match/Capture 0
0x3F	TC3 MC1	Match/Capture 1
0x40	TC4 OVF	Overflow/Underflow
0x41	TC4 MC0	Match/Capture 0
0x42	TC4 MC1	Match/Capture 1
0x43	ADC0 RESRDY	Result Ready
0x44	ADC0 WINMON	Window Monitor
0x45	ADC1 RESRDY	Result Ready
0x46	ADC1 WINMON	Window Monitor
0x47	SDADC RESRDY	Result Ready
0x48	SDADC WINMON	Window Monitor
0x49	AC COMP0	Comparator 0
0x4A	AC COMP1	Comparator 1
0x4B	AC COMP2	Comparator 2
0x4C	AC COMP3	Comparator 3
0x4D	AC WIN0	Window 0
0x4E	AC WIN1	Window 1
0x4F	DAC EMPTY	Data Buffer Empty
0x50	PTC EOC	End of Conversion

Value	Event Generator	Description
0x51	PTC WCOMP	Window Comparator
0x52	CCL LUTOUT0	CCL output
0x53	CCL LUTOUT1	CCL output
0x54	CCL LUTOUT2	CCL output
0x55	CCL LUT3	CCL output
0x56	PAC ACCERR	Access Error
0x57-0xFF	-	Reserved



### 30.8.8. Event User m

**Name:** USERm  
**Offset:** 0x80+m\*0x4 [m=0..46]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection



#### Bits 7:0 – CHANNEL[7:0]: Channel Event Selection

These bits are used to select the channel to connect to the event user.

Note that to select channel m, the value (m+1) must be written to the USER.CHANNEL bit group.

Value	Channel Number
0x00	No channel output selected
0x01	0
0x02	1
0x03	2
0x04	3
0x05	4
0x06	5
0x07	6
0x08	7
0x09	8
0x0A	9

Value	Channel Number
0x0B	10
0x0C	11
0x0D-0xFF	Reserved

**Table 30-2. User Multiplexer Number**

USERm	User Multiplexer	Description	Path Type
m = 0	TSENS START	Start measurement	Asynchronous, synchronous, and resynchronized paths
m = 1	PORT EV0	Event 0	Asynchronous, synchronous, and resynchronized paths
m = 2	PORT EV1	Event 1	Asynchronous, synchronous, and resynchronized paths
m = 3	PORT EV2	Event 2	Asynchronous, synchronous, and resynchronized paths
m = 4	PORT EV3	Event 3	Asynchronous, synchronous, and resynchronized paths
m = 5	DMAC CH0	Channel 0	Asynchronous, synchronous, and resynchronized paths
m = 6	DMAC CH1	Channel 1	Asynchronous, synchronous, and resynchronized paths
m = 7	DMAC CH2	Channel 2	Asynchronous, synchronous, and resynchronized paths
m = 8	DMAC CH3	Channel 3	Asynchronous, synchronous, and resynchronized paths
m = 9	TCC0 EV0	-	Asynchronous, synchronous, and resynchronized paths
m = 10	TCC0 EV1	-	Asynchronous, synchronous, and resynchronized paths

USERm	User Multiplexer	Description	Path Type
m = 11	TCC0 MC0	Match/Capture 0	Asynchronous, synchronous, and resynchronized paths
m = 12	TCC0 MC1	Match/Capture 1	Asynchronous, synchronous, and resynchronized paths
m = 13	TCC0 MC2	Match/Capture 2	Asynchronous, synchronous, and resynchronized paths
m = 14	TCC0 MC3	Match/Capture 3	Asynchronous, synchronous, and resynchronized paths
m = 15	TCC1 EV0	-	Asynchronous, synchronous, and resynchronized paths
m = 16	TCC1 EV1	-	Asynchronous, synchronous, and resynchronized paths
m = 17	TCC1 MC0	Match/Capture 0	Asynchronous, synchronous, and resynchronized paths
m = 18	TCC1 MC1	Match/Capture 1	Asynchronous, synchronous, and resynchronized paths
m = 19	TCC2 EV0	-	Asynchronous, synchronous, and resynchronized paths
m = 20	TCC2 EV1	-	Asynchronous, synchronous, and resynchronized paths
m = 21	TCC2 MC0	Match/Capture 0	Asynchronous, synchronous, and resynchronized paths
m = 22	TCC2 MC1	Match/Capture 1	Asynchronous, synchronous, and resynchronized paths
m = 23	TC0	-	Asynchronous, synchronous, and resynchronized paths
m = 24	TC1	-	Asynchronous, synchronous, and resynchronized paths

USERm	User Multiplexer	Description	Path Type
m = 25	TC2	-	Asynchronous, synchronous, and resynchronized paths
m = 26	TC3	-	Asynchronous, synchronous, and resynchronized paths
m = 27	TC4	-	Asynchronous, synchronous, and resynchronized paths
m = 28	ADC0 START	ADC start conversion	Asynchronous path only
m = 29	ADC0 SYNC	Flush ADC	Asynchronous path only
m = 30	ADC1 START	ADC start conversion	Asynchronous path only
m = 31	ADC1 SYNC	Flush ADC	Asynchronous path only
m = 32	SDADC START	Flush ADC	Asynchronous path only
m = 33	SDADC FLUSH	Flush ADC	Asynchronous path only
m = 34	AC COMP0	Start comparator 0	Asynchronous path only
m = 35	AC COMP1	Start comparator 1	Asynchronous path only
m = 36	AC COMP2	Start comparator 2	Asynchronous path only
m = 37	AC COMP3	Start comparator 3	Asynchronous path only
m = 38	DAC START	DAC start conversion	Asynchronous path only
m = 39	PTC STCONC	PTC start conversion	Asynchronous path only
m = 40	CCL LUTIN 0	CCL input	Asynchronous path only
m = 41	CCL LUTIN 1	CCL input	Asynchronous path only
m = 42	CCL LUTIN 2	CCL input	Asynchronous path only
m = 43	CCL LUTIN 3	CCL input	Asynchronous path only
others	Reserved	-	Reserved

## 31. SERCOM – Serial Communication Interface

### 31.1. 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 SERCOM is configured and enabled, all SERCOM resources will be dedicated to the selected mode.

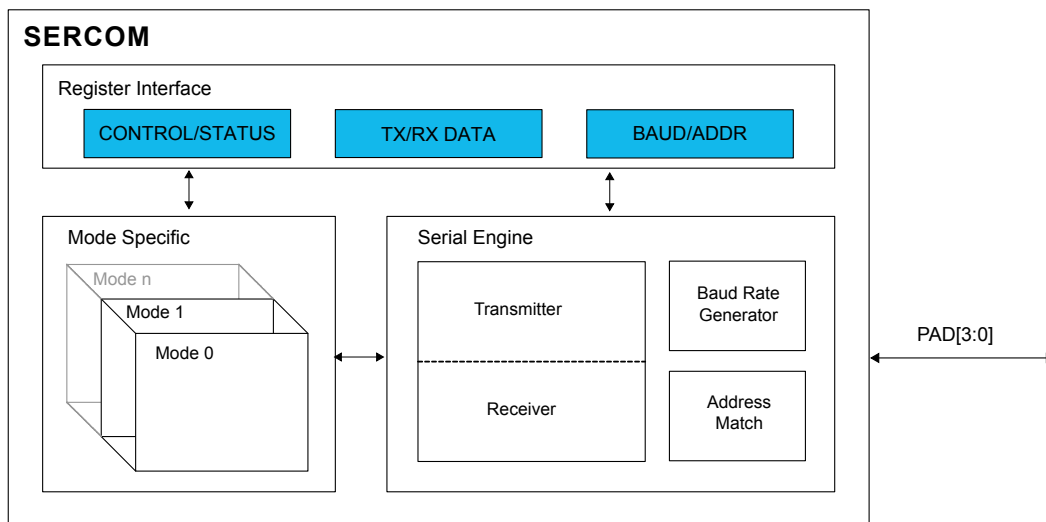
The SERCOM serial engine consists of a transmitter and receiver, baud-rate generator and address matching functionality. It can use the internal generic clock or an external clock to operate in all sleep modes.

### 31.2. Features

- Interface for configuring into one of the following:
  - I<sup>2</sup>C – Two-wire serial interface  
SMBus™ compatible
  - SPI – Serial peripheral interface
  - USART – Universal synchronous and asynchronous serial receiver and transmitter
- Single transmit buffer and double receive buffer
- Baud-rate generator
- Address match/mask logic
- Operational in all sleep modes
- Can be used with DMA

### 31.3. Block Diagram

Figure 31-1. SERCOM Block Diagram



### 31.4. Signal Description

See the respective SERCOM mode chapters for details.

### Related Links

[SERCOM USART – SERCOM Universal Synchronous and Asynchronous Receiver and Transmitter](#) on page 505

[SERCOM SPI – SERCOM Serial Peripheral Interface](#) on page 545

[SERCOM I2C – SERCOM Inter-Integrated Circuit](#) on page 579

## 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

Using the SERCOM I/O lines requires the I/O pins to be configured using port configuration (PORT).

From *USART Block Diagram* one can see that the SERCOM has four internal pads, PAD[3:0]. 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 Universal Synchronous and Asynchronous Receiver and Transmitter](#) on page 505

[SERCOM SPI – SERCOM Serial Peripheral Interface](#) on page 545

[SERCOM I2C – SERCOM Inter-Integrated Circuit](#) on page 579

[PORT: IO Pin Controller](#) on page 444

[Block Diagram](#) on page 506

### 31.5.2. Power Management

The SERCOM can operate in any sleep mode where the selected clock source is running. SERCOM interrupts can be used to wake up the device from sleep modes.

### Related Links

[PM – Power Manager](#) on page 162

### 31.5.3. Clocks

The SERCOM bus clock (CLK\_SERCOMx\_APB) is enabled by default, and can be enabled and disabled in the Main 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](#) on page 504 for details.

### Related Links

[GCLK - Generic Clock Controller](#) on page 114

[MCLK – Main Clock](#) on page 131

### 31.5.4. 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](#) on page 334

### 31.5.5. 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](#) on page 41

### 31.5.6. Events

Not applicable.

### 31.5.7. 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.

### 31.5.8. 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 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](#) on page 46

### 31.5.9. Analog Connections

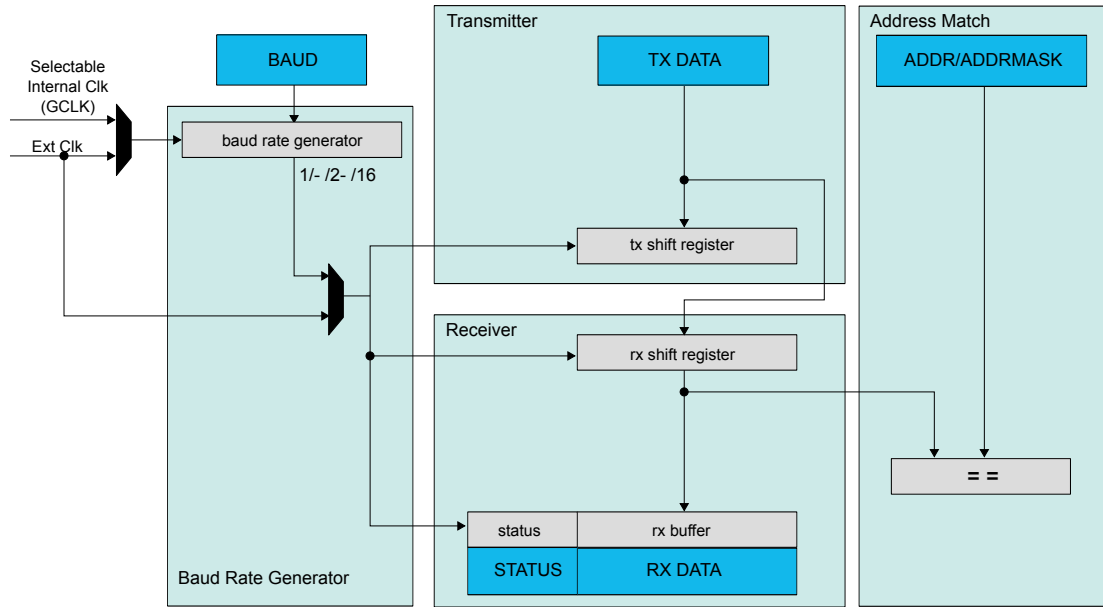
Not applicable.

## 31.6. Functional Description

### 31.6.1. Principle of Operation

The basic structure of the SERCOM serial engine is shown in [Figure 31-2 SERCOM Serial Engine](#) on page 500. 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 31-2. SERCOM Serial Engine**



The transmitter consists of a single write buffer and a shift register.

The receiver consists of a two-level receive buffer and a shift register.

The baud-rate generator is capable of running on the GCLK\_SERCOMx\_CORE clock or an external clock.

Address matching logic is included for SPI and I<sup>2</sup>C operation.

## 31.6.2. Basic Operation

### 31.6.2.1. 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 31-1. 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 Universal Synchronous and Asynchronous Receiver and Transmitter](#) on page 505

[SERCOM SPI – SERCOM Serial Peripheral Interface](#) on page 545



### 31.6.2.2. 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.

### 31.6.2.3. Clock Generation – Baud-Rate Generator

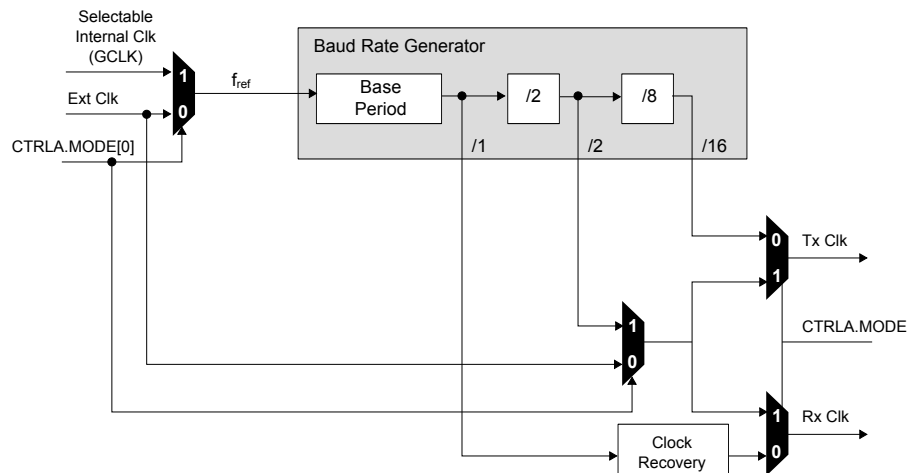
The baud-rate generator, as shown in [Figure 31-3 Baud Rate Generator](#) on page 501, generates internal clocks for asynchronous and synchronous communication. The output frequency ( $f_{\text{BAUD}}$ ) is determined by the Baud register (BAUD) setting and the baud reference frequency ( $f_{\text{ref}}$ ). 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 31-3. Baud Rate Generator**



[Table 31-2 Baud Rate Equations](#) on page 502 contains equations for the baud rate (in bits per second) and the BAUD register value for each operating mode.

For asynchronous operation, there are two different modes: In *arithmetic mode*, the BAUD register value is 16 bits (0 to 65,535). In *fractional mode*, the BAUD register is 13 bits, while the fractional adjustment is 3 bits. In this mode the BAUD setting must be greater than or equal to 1.

For synchronous operation, the BAUD register value is 8 bits (0 to 255).

**Table 31-2. Baud Rate Equations**

Operating Mode	Condition	Baud Rate (Bits Per Second)	BAUD Register Value Calculation
Asynchronous Arithmetic	$f_{BAUD} \leq \frac{f_{ref}}{S}$	$f_{BAUD} = \frac{f_{ref}}{S} \left(1 - \frac{BAUD}{65536}\right)$	$BAUD = 65536 \cdot \left(1 - S \cdot \frac{f_{BAUD}}{f_{ref}}\right)$
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. 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 31-3 BAUD Register Value vs. Baud Frequency on page 502 shows the BAUD register value versus baud frequency  $f_{BAUD}$  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 31-3. BAUD Register Value vs. Baud Frequency**

BAUD Register Value	Serial Engine CPF	$f_{BAUD}$ at 48MHz Serial Engine Frequency ( $f_{REF}$ )
0 – 406	160	3MHz
407 – 808	161	2.981MHz
809 – 1205	162	2.963MHz
...	...	...
65206	31775	15.11kHz
65207	31871	15.06kHz
65208	31969	15.01kHz

### 31.6.3. Additional Features

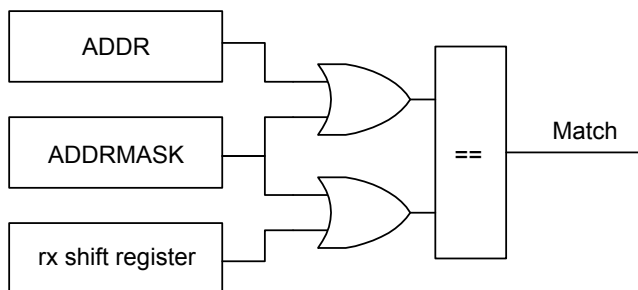
#### 31.6.3.1. Address Match and Mask

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.

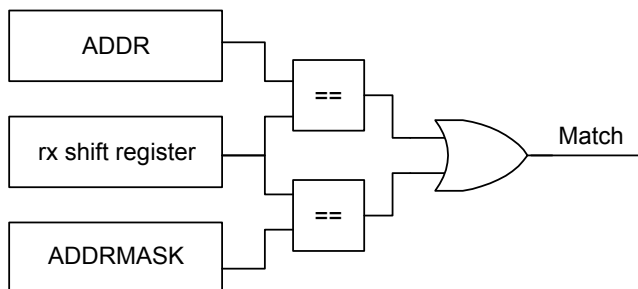
Figure 31-4. Address With Mask



##### Two Unique Addresses

The two addresses written to ADDR and ADDRMASK will cause a match.

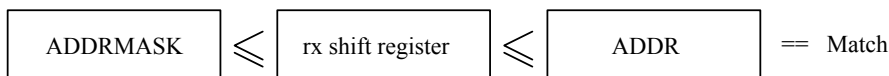
Figure 31-5. Two Unique Addresses



##### 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 31-6. Address Range



#### 31.6.4. DMA Operation

Not applicable.

#### 31.6.5. 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 SERCOM has one common interrupt request line for all the interrupt sources. The value of INTFLAG indicates which interrupt condition occurred. The user must read the INTFLAG register to determine which interrupt condition is present.

**Note:**

Note that interrupts must be globally enabled for interrupt requests.

**Related Links**

[Nested Vector Interrupt Controller](#) on page 41

**31.6.6. Events**

Not applicable.

**31.6.7. 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.

**31.6.8. Synchronization**

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

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](#) on page 110

## 32. SERCOM USART – SERCOM Universal Synchronous and Asynchronous Receiver and Transmitter

### 32.1. 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](#) on page 506. 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](#) on page 497

### 32.2. USART Features

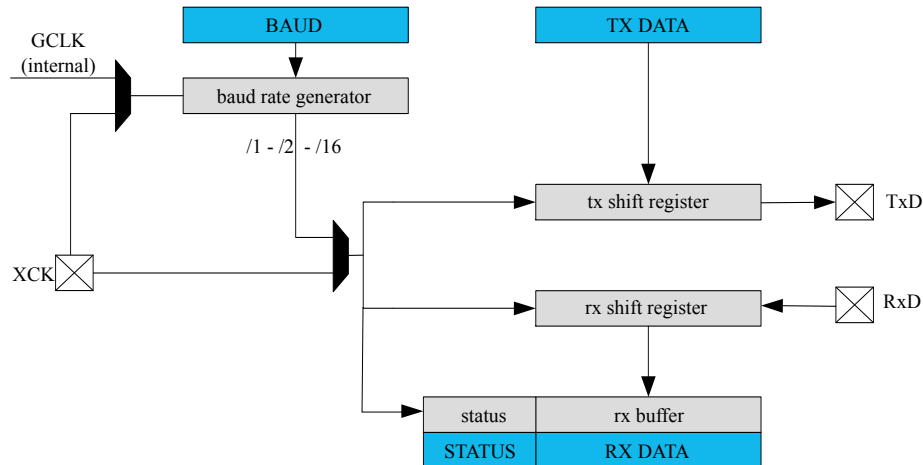
- Full-duplex operation
- Asynchronous (with clock reconstruction) or synchronous operation
- Internal or external clock source for asynchronous and synchronous operation
- Baud-rate generator
- Supports serial frames with 5, 6, 7, 8 or 9 data bits and 1 or 2 stop bits
- Odd or even parity generation 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 master support
- LIN slave support
  - Auto-baud and break character detection
- RS485 Support
- Start-of-frame detection
- Can work with DMA

#### Related Links

[Features](#) on page 497

## 32.3. Block Diagram

Figure 32-1. USART Block Diagram



## 32.4. Signal Description

Table 32-1. 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](#) on page 27

## 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

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 32-2. 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 32-2 USART Pin Configuration](#) on page 506.

### Related Links

[PORT: IO Pin Controller](#) on page 444

#### 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. Refer to *PM – Power Manager* for details on the different sleep modes.

### Related Links

[PM – Power Manager](#) on page 162

#### 32.5.3. Clocks

The SERCOM bus clock (CLK\_SERCOMx\_APB) is enabled by default, and can be disabled and enabled in the Main Clock Controller. Refer to *Peripheral Clock Masking* for details.

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](#) on page 519 for further details.

### Related Links

[Peripheral Clock Masking](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

#### 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](#) on page 334

#### 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](#) on page 41

#### 32.5.6. Events

Not applicable.

#### 32.5.7. 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.

#### 32.5.8. 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](#) on page 46

### 32.5.9. Analog Connections

Not applicable.

## 32.6. Functional Description

### 32.6.1. Principle of Operation

The USART uses the following lines for data transfer:

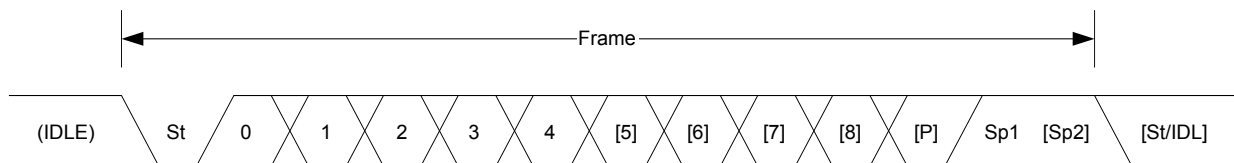
- RXD for receiving
- TXD for transmitting
- XCK for the transmission clock in synchronous operation

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 32-2. 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.



## 32.6.2. Basic Operation

### 32.6.2.1. 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)

Any writes to these registers when the USART is enabled or is being enabled (CTRL.ENABLE is one) will be discarded. Writes to these registers while the peripheral is being disabled, will be completed after the disabling is complete.

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).

### 32.6.2.2. 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.

### 32.6.2.3. 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.

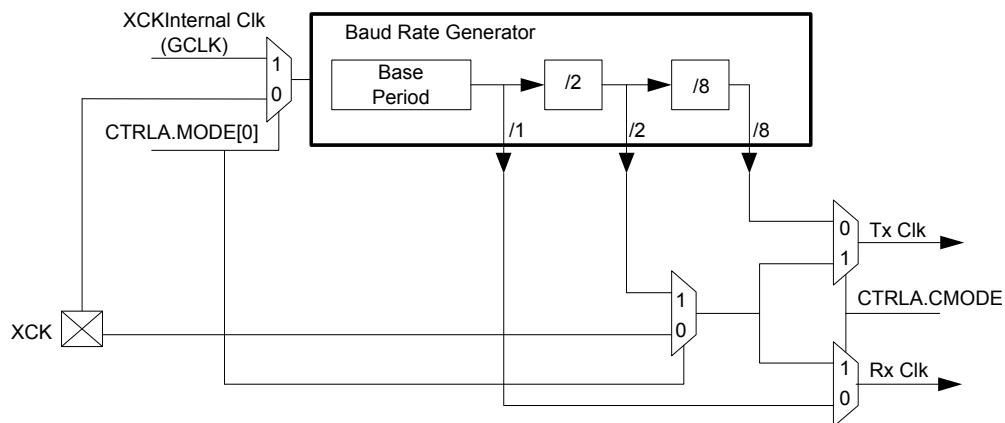
The internal clock source is selected by writing 0x1 to the Operation Mode bit field in the Control A register (CTRLA.MODE), the external clock source is selected by writing 0x0 to CTRLA.MODE.

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 32-3. Clock Generation



#### Related Links

[Clock Generation – Baud-Rate Generator](#) on page 501

[Asynchronous Arithmetic Mode BAUD Value Selection](#) on page 502

#### Synchronous Clock Operation

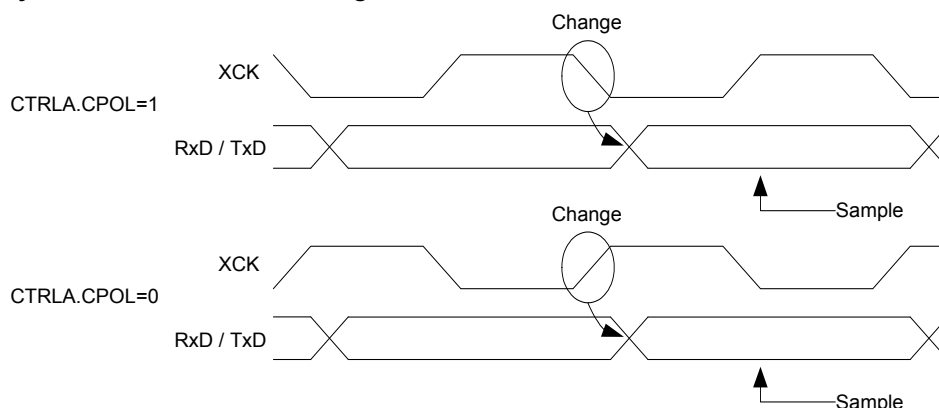
In synchronous mode, the CTRLA.MODE bit field determines whether the transmission clock line (XCK) serves either as input or output. The dependency between clock edges, data sampling, and data change is the same for internal and external clocks. Data input on the Rx pin is sampled at the opposite XCK clock edge when data is driven on the Tx pin.

The Clock Polarity bit in the Control A register (CTRLA.CPOL) selects which XCK clock edge is used for Rx sampling, and which is used for Tx 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 32-4. 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.

#### 32.6.2.4. 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.

#### 32.6.2.5. 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.

#### 32.6.2.6. 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 32-3. 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
9	96.39	104.76	+3.61/-4.76	±1.5
10	96.70	104.35	+3.30/-4.35	±1.5

The recommended maximum receiver baud-rate error assumes that the receiver and transmitter equally divide the maximum total error.

The following equations can calculate the ratio of the incoming data rate and internal receiver baud rate:

$$R_{\text{SLOW}} = \frac{(D + 1)S}{S - 1 + D \cdot S + S_F} \quad , \quad R_{\text{FAST}} = \frac{(D + 2)S}{(D + 1)S + S_M}$$

- $R_{\text{SLOW}}$  is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- $R_{\text{FAST}}$  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.

#### Related Links

[Clock Generation – Baud-Rate Generator](#) on page 501

[Asynchronous Arithmetic Mode BAUD Value Selection](#) on page 502

### 32.6.3. Additional Features

#### 32.6.3.1. 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.

#### 32.6.3.2. 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 32-5. Connection with a Remote Device for Hardware Handshaking**

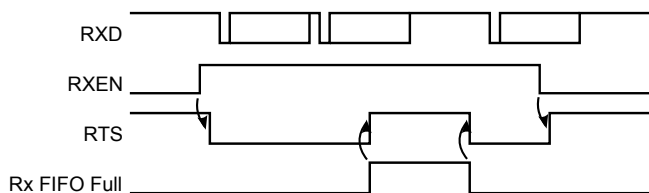


Hardware handshaking is only available in the following configuration:

- USART with internal clock (CTRLA.MODE=1),
- Asynchronous mode (CTRLA.CMODE=0),
- and Flow control pinout (CTRLA.TXPO=2).

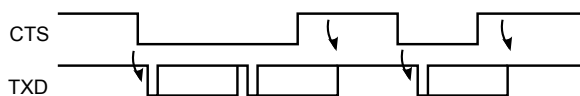
When the receiver is disabled or the receive FIFO is full, the receiver will drive the RTS pin high. This notifies the remote device to stop transfer after the ongoing transmission. Enabling and disabling the receiver by writing to CTRLB.RXEN will set/clear the RTS pin after a synchronization delay. 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 32-6. 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 32-7. Transmitter Behavior when Operating with Hardware Handshaking**



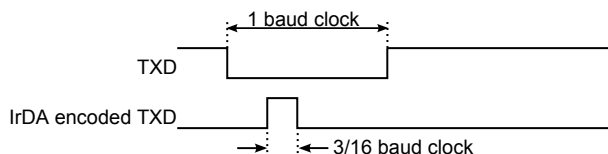
### 32.6.3.3. IrDA Modulation and Demodulation

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 32-8. IrDA Transmit Encoding**



The reception decoder has two main functions.

The first is to synchronize the incoming data to the IrDA baud rate counter. Synchronization is performed at the start of each zero pulse.

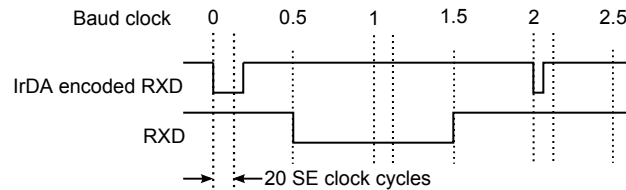
The second main function is to decode incoming Rx data. If a pulse width meets the minimum length set by configuration (RXPL.RXPL), it is accepted. When the baud rate counter reaches its middle value (1/2 bit length), it is transferred to the receiver.

**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 32-9. IrDA Receive Decoding**



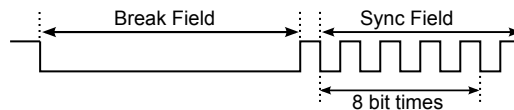
#### 32.6.3.4. Break Character Detection and Auto-Baud

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 auto-baud follows the LIN format. All LIN Frames start with a Break Field followed by a Sync Field. The USART uses a break detection threshold of greater than 11 nominal bit times at the configured baud rate. At any time, if more than 11 consecutive dominant bits are detected on the bus, the USART detects a Break Field. When a Break Field has been detected, the Receive Break interrupt flag (INTFLAG.RXBRK) is set and the USART expects the Sync Field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized. If the received Sync character is not 0x55, then the Inconsistent Sync Field error flag (STATUS.ISF) is set along with the Error interrupt flag (INTFLAG.ERROR), and the baud rate is unchanged.

**Figure 32-10. LIN Break and Sync Fields**



After a break field is detected and the start bit of the Sync Field is detected, a counter is started. The counter is then incremented for the next 8 bit times of the Sync Field. At the end of these 8 bit times, the counter is stopped. At this moment, the 13 most significant bits of the counter (value divided by 8) 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.

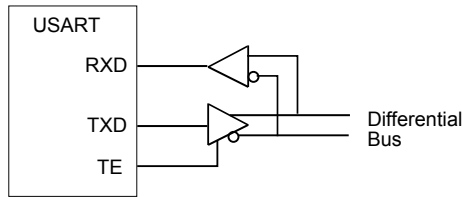
#### 32.6.3.5. RS485

RS485 is available with the following configuration:

- USART frame format (CTRLA.FORM = 0x00 or 0x01)
- RS485 pinout (CTRLA.TXPO=0x3).

The RS485 feature enables control of an external line driver as shown in the figure below. While operating in RS485 mode, the transmit enable pin (TE) is driven high when the transmitter is active.

**Figure 32-11. RS485 Bus Connection**



The TE pin will remain high for the complete frame including stop bit(s). If a Guard Time is programmed in the Control C register (CTRLC.GTIME), the line will remain driven after the last character completion. The following figure shows a transfer with one stop bit and CTRLC.GTIME=3.

**Figure 32-12. Example of TE Drive with Guard Time**



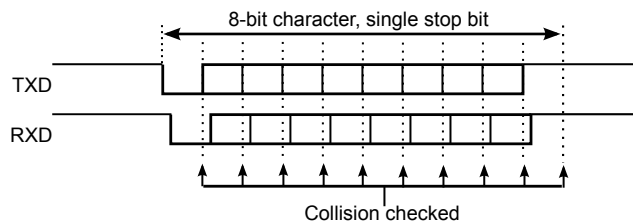
The Transmit Complete interrupt flag (INTFLAG.TXC) will be raised after the guard time is complete and TE goes low.

### 32.6.3.6. 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).

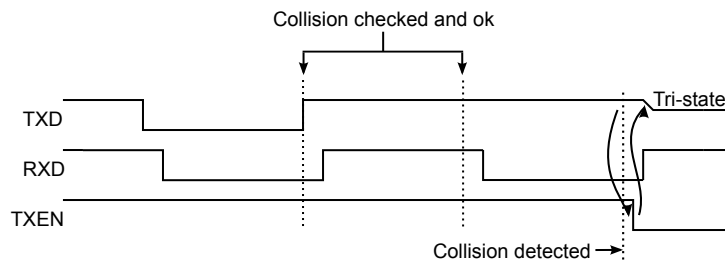
Collision detection is performed for each bit transmitted by comparing the received value with the transmit value, as shown in the figure below. While the transmitter is idle (no transmission in progress), characters can be received on RxD without triggering a collision.

**Figure 32-13. 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 32-14. 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.

#### 32.6.3.7. 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.

#### 32.6.3.8. 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](#) on page 1105

#### 32.6.3.9. 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.

## 32.6.4. DMA, Interrupts and Events

Table 32-4. Module Request for SERCOM USART

Condition	Request		
	DMA	Interrupt	Event
Data Register Empty (DRE)	Yes (request cleared when data is written)	Yes	NA
Receive Complete (RXC)	Yes (request cleared when data is read)	Yes	
Transmit Complete (TXC)	NA	Yes	
Receive Start (RXS)	NA	Yes	
Clear to Send Input Change (CTSIC)	NA	Yes	
Receive Break (RXBRK)	NA	Yes	
Error (ERROR)	NA	Yes	

### 32.6.4.1. DMA Operation

The USART generates the following DMA requests:

- Data received (RX): The request is set when data is available in the receive FIFO. The request is cleared when DATA is read.
- Data transmit (TX): The request is set when the transmit buffer (TX DATA) is empty. The request is cleared when DATA is written.

### 32.6.4.2. 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 USART has one common interrupt request line for all the interrupt sources. 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](#) on page 41

### 32.6.4.3. Events

Not applicable.

### 32.6.5. 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.

### 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. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

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](#) on page 526 for details.

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

## Related Links

[Register Synchronization](#) on page 110

## 32.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY				MODE[2:0]		ENABLE	SWRST
0x01		15:8	SAMPR[2:0]							IBON
0x02		23:16	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
0x03		31:24		DORD	CPOL	CMODE		FORM[3:0]		
0x04	CTRLB	7:0		SBMODE				CHSIZE[2:0]		
0x05		15:8			PMODE			ENC	SFDE	COLDEN
0x06		23:16							RXEN	TXEN
0x07		31:24								
0x08	CTRLC	7:0						GTIME[2:0]		
0x09		15:8					HDRDLY[1:0]		BRKLEN[1:0]	
0x0A		23:16								
0x0B		31:24								
0x0C	BAUD	7:0	BAUD[7:0]							
0x0D		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
0x1B		15:8								
0x1C	SYNDBUSY	7:0						CTRLB	ENABLE	SWRST
0x1D		15:8								
0x1E		23:16								
0x1F		31:24								
0x20 ... 0x27	Reserved									
0x28	DATA	7:0	DATA[7:0]							
0x29		15:8								DATA[8:8]
0x2A ... 0x2F	Reserved									
0x30	DBGCTRL	7:0								DBGSTOP

## 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

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 - 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	SERCOM_PAD[0]	SERCOM_PAD[1]	SERCOM_PAD[2]	N/A

#### 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.

## 32.8.2. 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	R/W
Reset							0	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 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 receiver 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

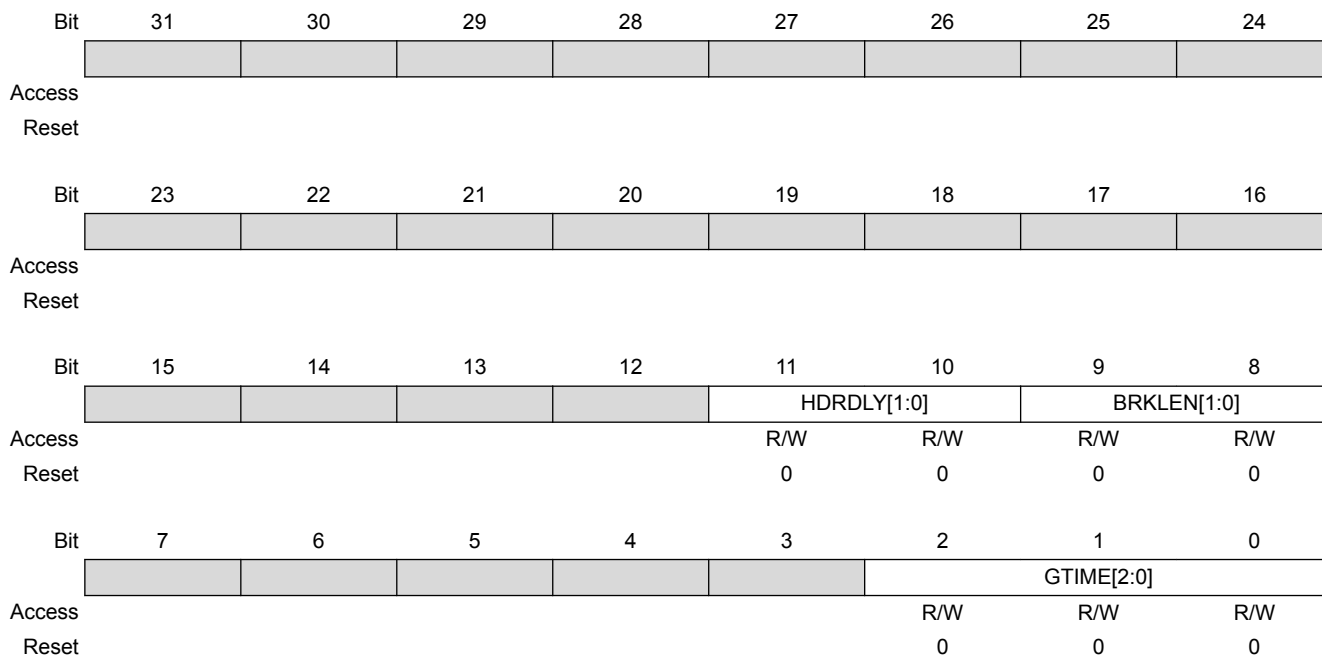
These bits select the number of bits in a character.

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

### 32.8.3. Control C

**Name:** CTRLC  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



#### Bits 11:10 – HDRDLY[1:0]: LIN Master Header Delay

These bits define the delay between break and sync transmission in addition to the delay between the sync and identifier (ID) fields when in LIN master mode (CTRLA.FORM=0x2).

This field is only valid when using the LIN header command (CTRLB.LINCMD=0x2).

Value	Description
0x0	Delay between break and sync transmission is 1 bit time. Delay between sync and ID transmission is 1 bit time.
0x1	Delay between break and sync transmission is 4 bit time. Delay between sync and ID transmission is 4 bit time.
0x2	Delay between break and sync transmission is 8 bit time. Delay between sync and ID transmission is 4 bit time.
0x3	Delay between break and sync transmission is 14 bit time. Delay between sync and ID transmission is 4 bit time.

#### Bits 9:8 – BRKLEN[1:0]: LIN Master Break Length

These bits define the length of the break field transmitted when in LIN master mode (CTRLA.FORM=0x2).

Value	Description
0x0	Break field transmission is 13 bit times
0x1	Break field transmission is 17 bit times
0x2	Break field transmission is 21 bit times
0x3	Break field transmission is 26 bit times

**Bits 2:0 – GTIME[2:0]: Guard Time**

These bits define the guard time when using RS485 mode (CTRLA.TXPO=0x3).

For RS485 mode, the guard time is programmable from 0-7 bit times and defines the time that the transmit enable pin (TE) remains high after the last stop bit is transmitted and there is no remaining data to be transmitted.

Value	Description
0	The receiver is disabled or being enabled.
1	The receiver is enabled or will be enabled when the USART is enabled.

### 32.8.4. 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[21:0]: Baud Value**

These bits control the clock generation, as described in the *SERCOM Clock Generation – Baud-Rate Generator* section.

### 32.8.5. Receive Pulse Length Register

**Name:** RXPL  
**Offset:** 0x0E  
**Reset:** 0x00  
**Property:** 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  $SE_{per}$ .

$$PULSE \geq (RXPL + 2) \cdot SE_{per}$$



### 32.8.6. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x14

**Reset:** 0x00

**Property:** PAC Write-Protection

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.

### 32.8.7. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x16

**Reset:** 0x00

**Property:** PAC Write-Protection

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.

## 32.8.8. 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.

### 32.8.9. 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
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

When CTRLA.FORM is set to LIN master mode, this bit is set when any ongoing transmission is complete and TxDATA is empty. When CTRLA.FORM is not set to LIN master mode, 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.

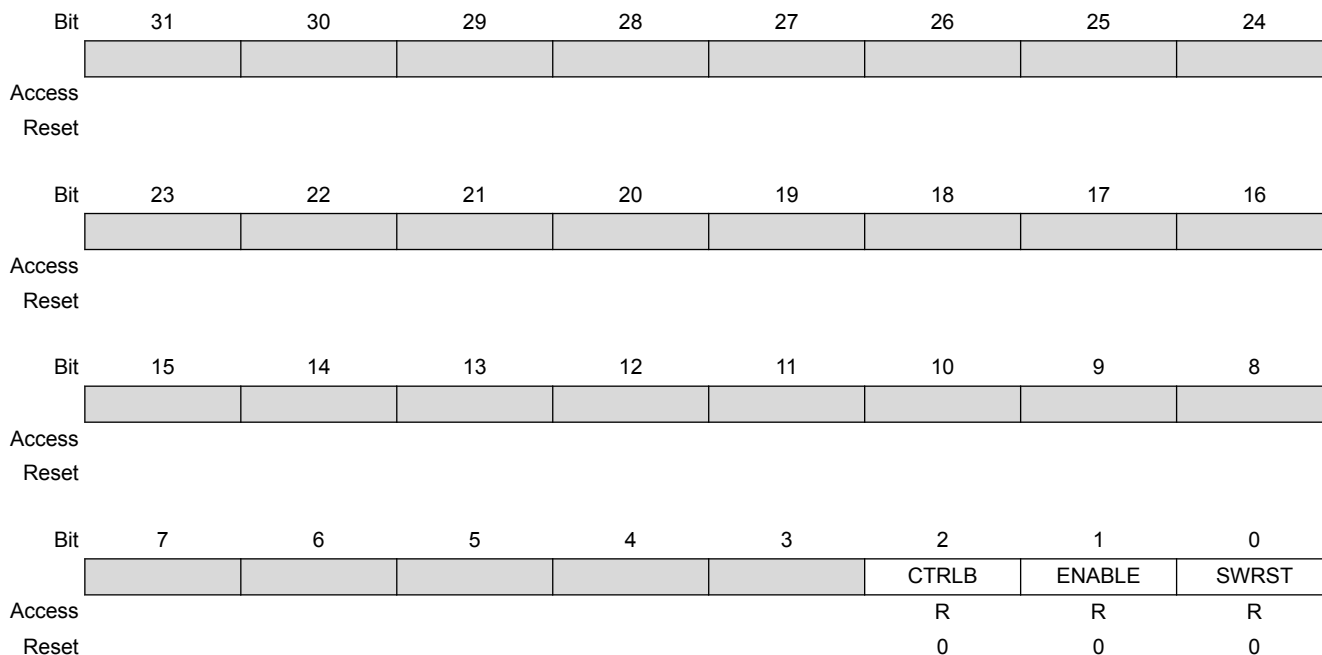
Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.



### 32.8.10. 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.

Writes to any register (except for CTRLA.SWRST) while enable synchronization is on-going will be discarded and an APB error will be generated.

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.

Writes to any register while synchronization is on-going will be discarded and an APB error will be generated.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

### 32.8.11. 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.

### 32.8.12. 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.

## 33. SERCOM SPI – SERCOM Serial Peripheral Interface

### 33.1. 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](#) on page 546. Each side, master and slave, depicts a separate SPI containing a shift register, a transmit buffer and two receive buffers. In addition, the SPI master uses the SERCOM baud-rate generator, while the SPI slave can use the SERCOM address match logic. 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](#) on page 497

### 33.2. Features

SERCOM SPI includes the following features:

- Full-duplex, four-wire interface (MISO, MOSI, SCK,  $\overline{SS}$ )
- Single-buffered transmitter, double-buffered receiver
- Supports all four SPI modes of operation
- Single data direction operation allows alternate function on MISO or MOSI pin
- Selectable LSB- or MSB-first data transfer
- Can be used with DMA
- Master operation:
  - Serial clock speed up to 12MHz
  - 8-bit clock generator
  - Hardware controlled  $\overline{SS}$
- Slave operation:
  - Serial clock speed up to the system clock
  - Optional 8-bit address match operation
  - Operation in all sleep modes
  - Wake on  $\overline{SS}$  transition

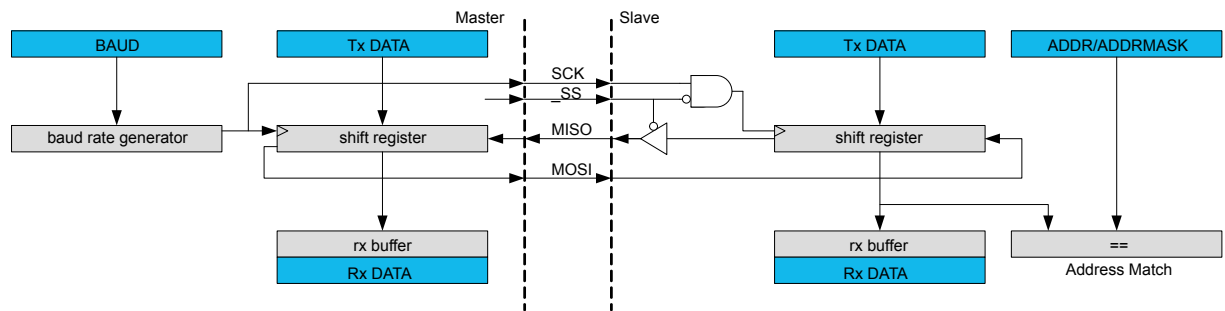
#### Related Links

[SERCOM – Serial Communication Interface](#) on page 497

[Features](#) on page 497

### 33.3. Block Diagram

Figure 33-1. Full-Duplex SPI Master Slave Interconnection



### 33.4. Signal Description

Table 33-1. 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](#) on page 27

### 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

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.MSSEN) is '1'.

Table 33-2. SPI Pin Configuration

Pin	Master SPI	Slave SPI
MOSI	Output	Input
MISO	Input	Output
SCK	Output	Input
$\overline{SS}$	Output (CTRLB.MSSEN=1)	Input

The combined configuration of PORT, 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](#) on page 444

#### 33.5.2. 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. Refer to *PM – Power Manager* for details on the different sleep modes.

### Related Links

[PM – Power Manager](#) on page 162

#### 33.5.3. Clocks

The SERCOM bus clock (CLK\_SERCOMx\_APB) is enabled by default, and can be enabled and disabled in the Main 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](#) on page 114

[Peripheral Clock Masking](#) on page 135

[Synchronization](#) on page 556

#### 33.5.4. 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](#) on page 334

#### 33.5.5. 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](#) on page 41

#### 33.5.6. Events

Not applicable.

#### 33.5.7. 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.

#### 33.5.8. 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](#) on page 46

### 33.5.9. Analog Connections

Not applicable.

## 33.6. Functional Description

### 33.6.1. Principle of Operation

The SPI is a high-speed synchronous data transfer interface. It allows high-speed communication between the device and peripheral devices.

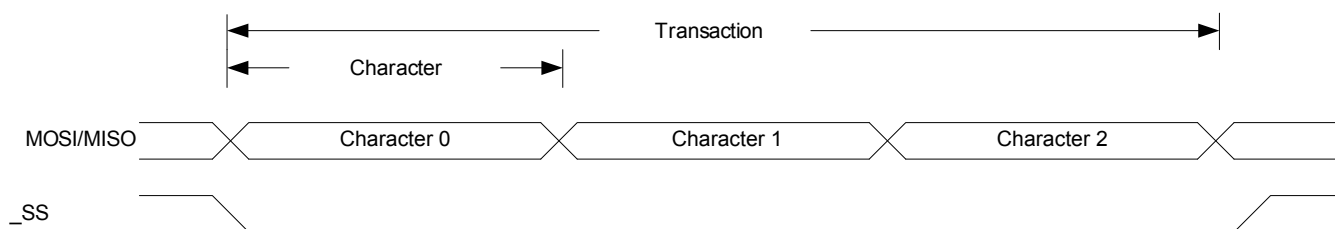
The SPI can operate as master or slave. As master, the SPI initiates and controls all data transactions. The SPI is single buffered for transmitting and double buffered for receiving.

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 33-2. 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.



## 33.6.2. Basic Operation

### 33.6.2.1. Initialization

The following registers are enable-protected, meaning that they can only be written when the SPI is disabled (CTRL.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).

### 33.6.2.2. 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.

### 33.6.2.3. 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](#) on page 501

[Asynchronous Arithmetic Mode BAUD Value Selection](#) on page 502

#### 33.6.2.4. 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.

#### 33.6.2.5. 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 33-3. SPI Transfer Modes**

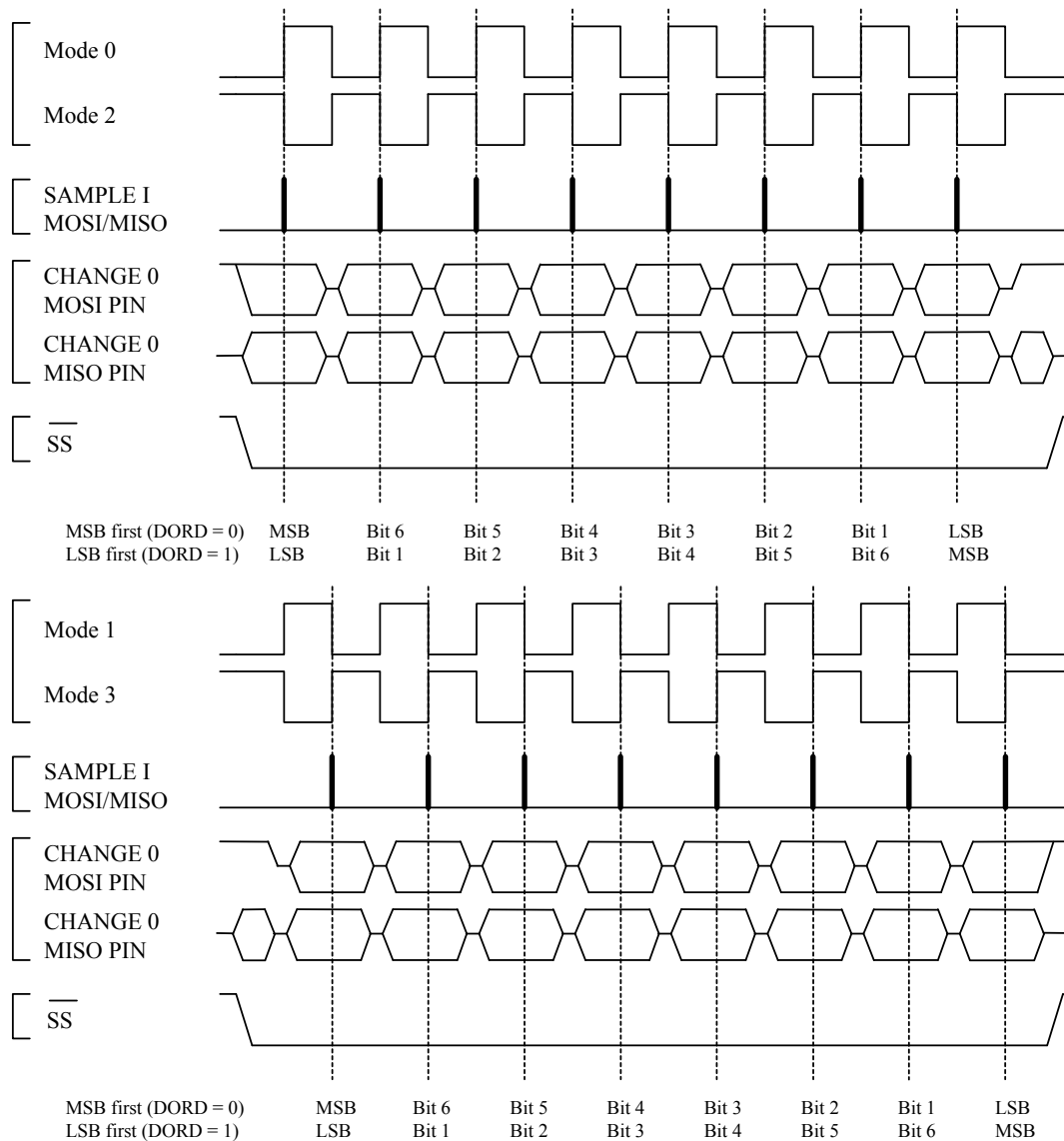
Mode	CPOL	CPHA	Leading Edge	Trailing Edge
0	0	0	Rising, sample	Falling, setup
1	0	1	Rising, setup	Falling, sample
2	1	0	Falling, sample	Rising, setup
3	1	1	Falling, setup	Rising, sample

#### Note:

Leading edge is the first clock edge in a clock cycle.

Trailing edge is the second clock edge in a clock cycle.

**Figure 33-3. SPI Transfer Modes**



### 33.6.2.6. 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](#) on page 553.

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.

#### 33.6.2.7. 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.

#### 33.6.3. Additional Features

##### 33.6.3.1. 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](#) on page 503

### 33.6.3.2. 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.

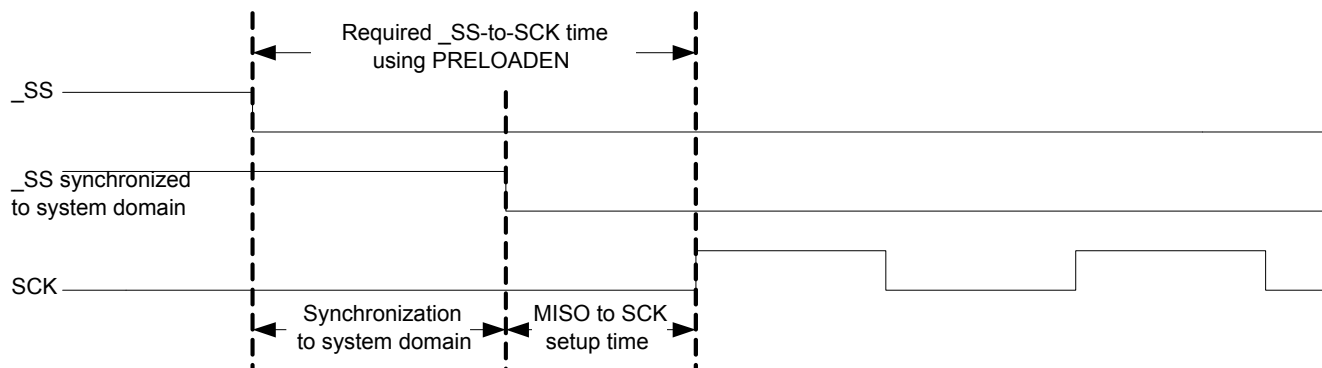
Preloading can be used to preload data into the shift register while  $\overline{SS}$  is high: this eliminates sending a dummy character when starting a transaction. If the shift register is not preloaded, the current contents of the shift register will be shifted out.

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).

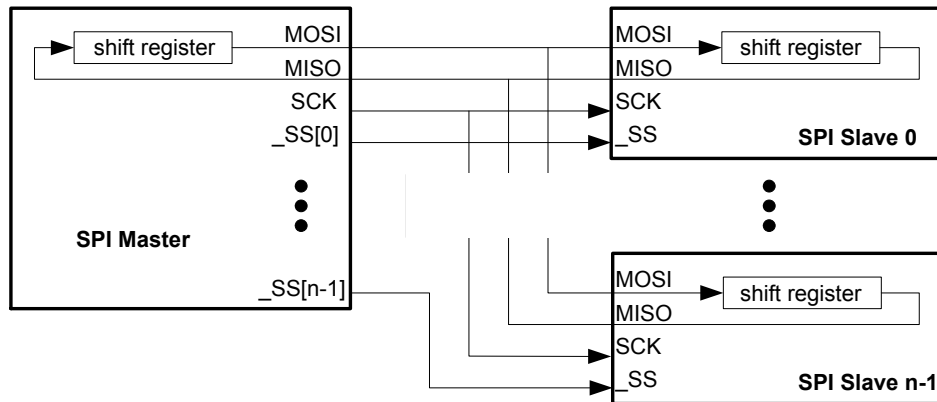
**Figure 33-4. Timing Using Preloading**



### 33.6.3.3. Master with Several Slaves

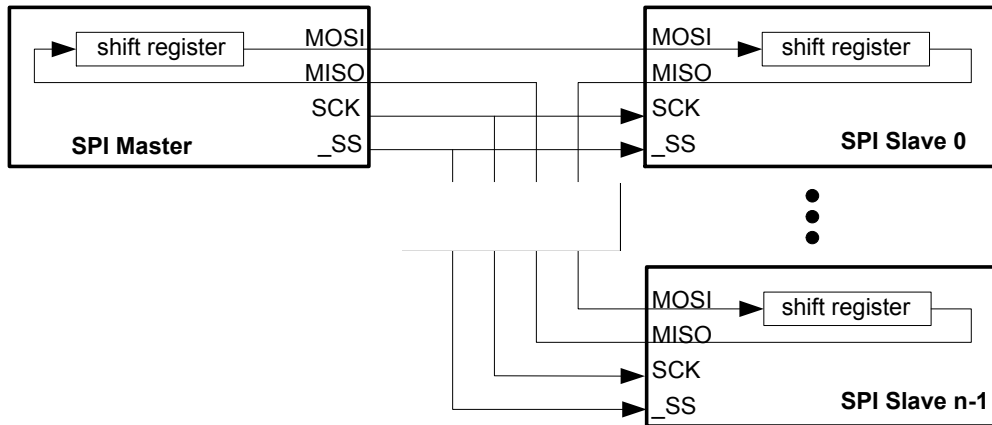
Master with multiple slaves in parallel is only available when Master Slave Select Enable (CTRLB.MSSEN) is set to zero and hardware  $\overline{SS}$  control is disabled. If the bus consists of several SPI slaves, an SPI master can use general purpose I/O pins to control the  $\overline{SS}$  line to 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 33-5. 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 33-6. Multiple Slaves in Series**



#### 33.6.3.4. Loop-Back Mode

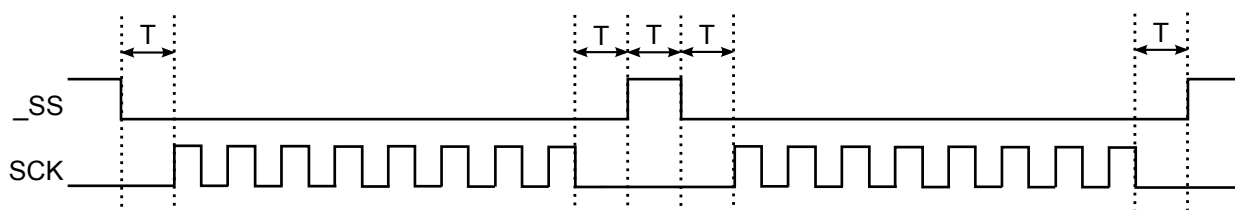
For loop-back mode, configure the Data In Pinout (CTRLA.DIPO) and Data Out Pinout (CTRLA.DOPO) to use the same data pins for transmit and receive. The loop-back is through the pad, so the signal is also available externally.

#### 33.6.3.5. 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 33-7. 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.

### 33.6.3.6. 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.

### 33.6.4. DMA, Interrupts, and Events

**Table 33-4. Module Request for SERCOM SPI**

Condition	Request		
	DMA	Interrupt	Event
Data Register Empty (DRE)	Yes (request cleared when data is written)	Yes	NA
Receive Complete (RXC)	Yes (request cleared when data is read)	Yes	
Transmit Complete (TXC)	NA	Yes	
Slave Select low (SSL)	NA	Yes	
Error (ERROR)	NA	Yes	

#### 33.6.4.1. DMA Operation

The SPI generates the following DMA requests:

- Data received (RX): The request is set when data is available in the receive FIFO. The request is cleared when DATA is read.
- Data transmit (TX): The request is set when the transmit buffer (TX DATA) is empty. The request is cleared when DATA is written.

#### 33.6.4.2. 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 SPI has one common interrupt request line for all the interrupt sources. 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](#) on page 41

#### 33.6.4.3. Events

Not applicable.

#### 33.6.5. 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: GCLK\_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.

#### 33.6.6. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

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](#) on page 564 for details.

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

#### Related Links



[Register Synchronization](#) on page 110

## 33.7. Register Summary

Offset	Name	Bit Pos.							
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]		ENABLE	SWRST
0x01		15:8							IBON
0x02		23:16			DIPO[1:0]				DOPO[1:0]
0x03		31:24		DORD	CPOL	CPHA		FORM[3:0]	
0x04	CTRLB	7:0		PLOADEN				CHSIZE[2:0]	
0x05		15:8		AMODE[1:0]	MSEN			SSDE	
0x06		23:16						RXEN	
0x07		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		
0x1B		15:8							
0x1C	SYNCBUSY	7:0					CTRLB	ENABLE	SWRST
0x1D		15:8							
0x1E		23:16							
0x1F		31:24							
0x20 ... 0x23	Reserved								
0x24	ADDR	7:0	ADDR[7:0]						
0x25		15:8							
0x26		23:16	ADDRMASK[7:0]						
0x27		31:24							
0x28	DATA	7:0	DATA[7:0]						
0x29		15:8							DATA[8:8]
0x2A ... 0x2F	Reserved								
0x30	DBGCTRL	7:0							DBGSTOP

### 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 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](#) on page 556

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](#) on page 547.

### 33.8.1. Control A

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

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.

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 CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.

Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.

This bit is not enable-protected.

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

### 33.8.2. Control B

**Name:** CTRLB  
**Offset:** 0x04  
**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
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

### 33.8.3. 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*.

### 33.8.4. Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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.

### 33.8.5. Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x16  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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.

### 33.8.6. 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.



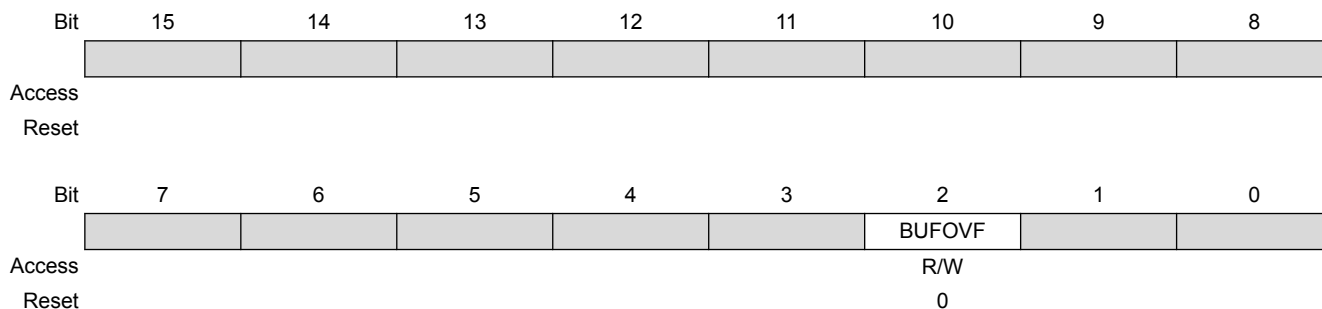
### 33.8.7. Status

**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.

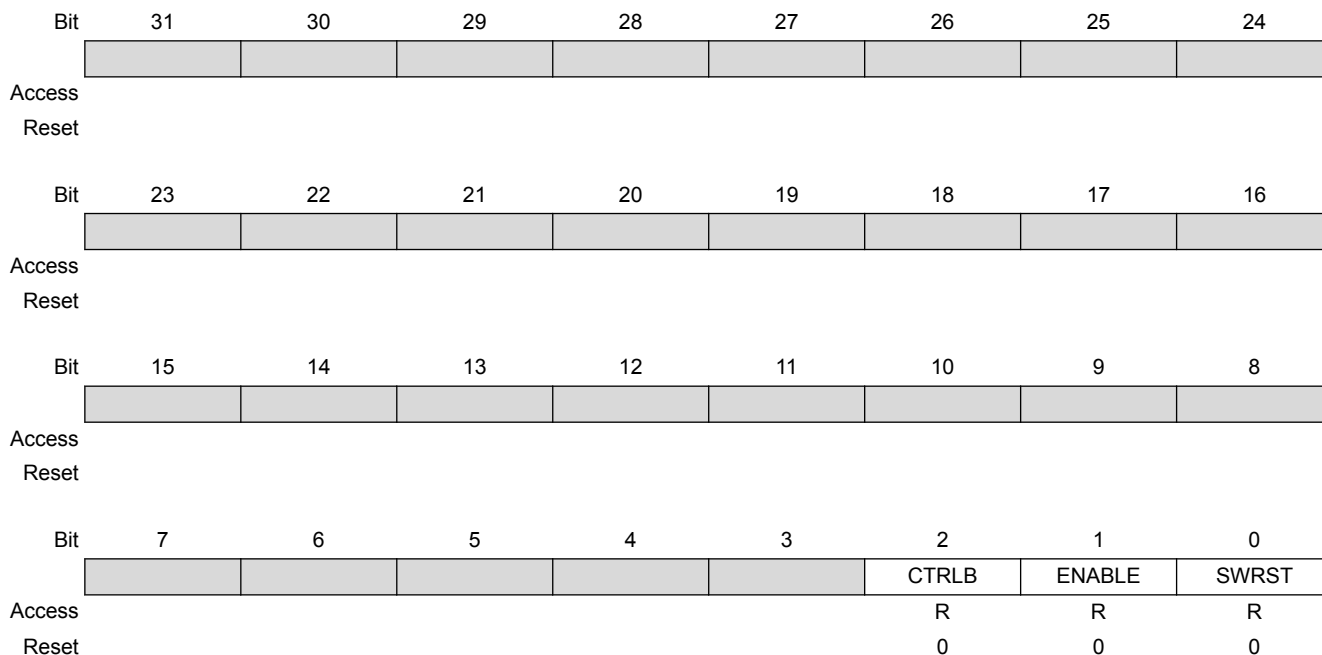
Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

Value	Description
0	No Buffer Overflow has occurred.
1	A Buffer Overflow has occurred.

### 33.8.8. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x1C  
**Reset:** 0x00000000  
**Property:** -



#### 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.

Writes to any register (except for CTRLA.SWRST) while enable synchronization is on-going will be discarded and an APB error will be generated.

Value	Description
0	Enable synchronization is not busy.
1	Enable synchronization is busy.

### Bit 0 – SWRST: Software Reset Synchronization Busy

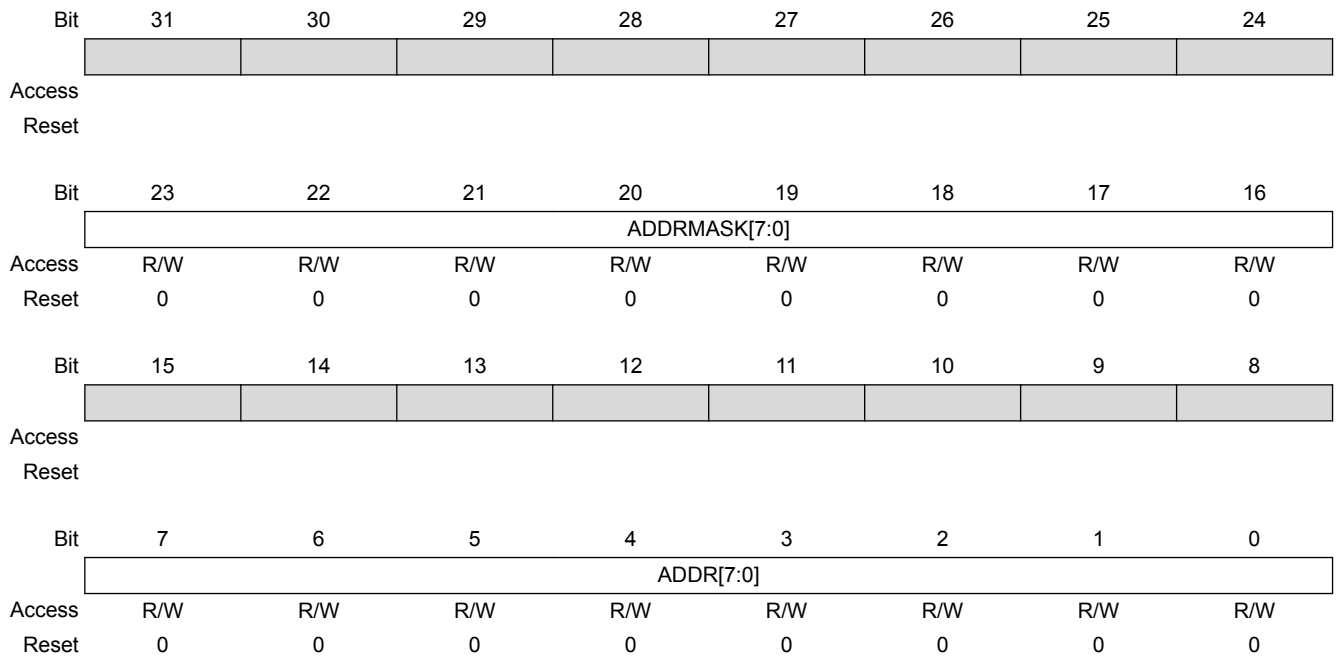
Resetting the SERCOM (CTRLA.SWRST) requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.SWRST=1 until synchronization is complete.

Writes to any register while synchronization is on-going will be discarded and an APB error will be generated.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

### 33.8.9. Address

**Name:** ADDR  
**Offset:** 0x24  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



#### 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).

### 33.8.10. 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.

### 33.8.11. 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.

## 34. SERCOM I<sup>2</sup>C – SERCOM Inter-Integrated Circuit

### 34.1. 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 34-1 I<sup>2</sup>C Single-Master Single-Slave Interconnection](#) on page 580. Labels in capital letters are registers accessible by the CPU, while lowercase labels are internal to the SERCOM. Each master and slave have a separate I<sup>2</sup>C 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](#) on page 497

### 34.2. Features

SERCOM I<sup>2</sup>C includes the following features:

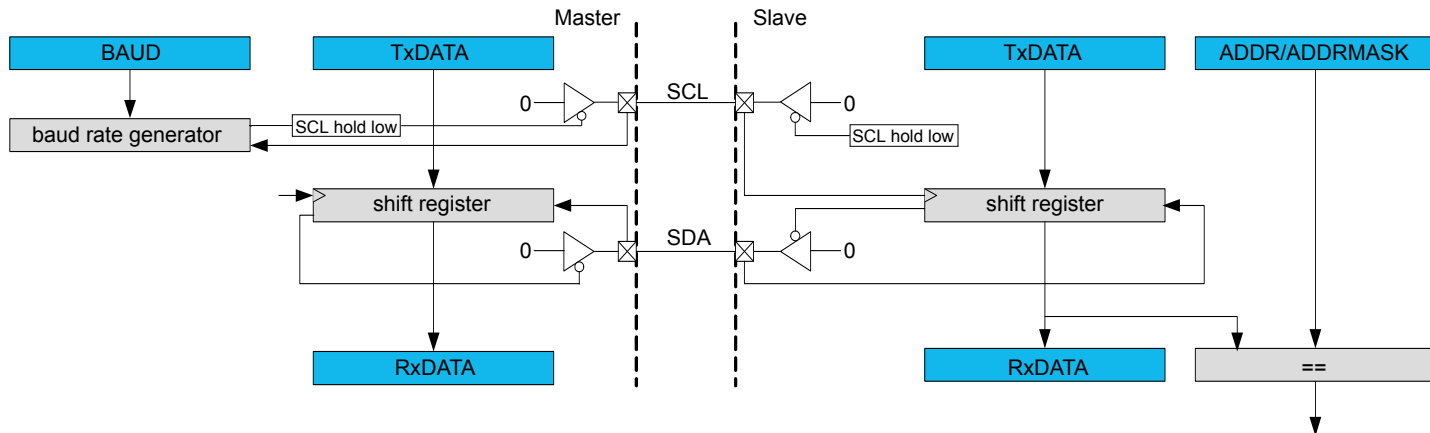
- Master or slave operation
- Can be used with DMA
- Philips I<sup>2</sup>C compatible
- SMBus™ compatible
- PMBus compatible
- Support of 100kHz and 400kHz, 1MHz and 3.4MHz I<sup>2</sup>C mode low system clock frequencies
- Physical interface includes:
  - Slew-rate limited outputs
  - Filtered inputs
- Slave operation:
  - Operation in all sleep modes
  - Wake-up on address match
  - 7-bit and 10-bit Address match in hardware for:
    - Unique address and/or 7-bit general call address
    - Address range
    - Two unique addresses can be used with DMA

#### Related Links

[Features](#) on page 497

### 34.3. Block Diagram

Figure 34-1. I<sup>2</sup>C Single-Master Single-Slave Interconnection



### 34.4. Signal Description

Signal Name	Type	Description
PAD[0]	Digital I/O	SDA
PAD[1]	Digital I/O	SCL
PAD[2]	Digital I/O	SDA_OUT (4-wire)
PAD[3]	Digital I/O	SDC_OUT (4-wire)

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](#) on page 27

### 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

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). Refer to *PORT - I/O Pin Controller* for details.

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](#) on page 444



### 34.5.2. 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. Refer to *PM – Power Manager* for details on the different sleep modes.

#### Related Links

[PM – Power Manager](#) on page 162

### 34.5.3. Clocks

The SERCOM bus clock (CLK\_SERCOMx\_APB) is enabled by default, and can be enabled and disabled in the Main Clock Controller and the Power Manager.

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 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](#) on page 599 for further details.

#### Related Links

[GCLK - Generic Clock Controller](#) on page 114

[Peripheral Clock Masking](#) on page 135

[PM – Power Manager](#) on page 162

### 34.5.4. 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](#) on page 334

### 34.5.5. 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](#) on page 41

### 34.5.6. Events

Not applicable.

### 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 - refer to the Debug Control (DBGCTRL) register for details.

Refer to the [DBGCTRL](#) on page 643 register for details.

### 34.5.8. 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](#) on page 46

### 34.5.9. Analog Connections

Not applicable.

## 34.6. Functional Description

### 34.6.1. Principle of Operation

The I<sup>2</sup>C interface uses two physical lines for communication:

- Serial Data Line (SDA) for packet transfer
- Serial Clock Line (SCL) for the bus clock

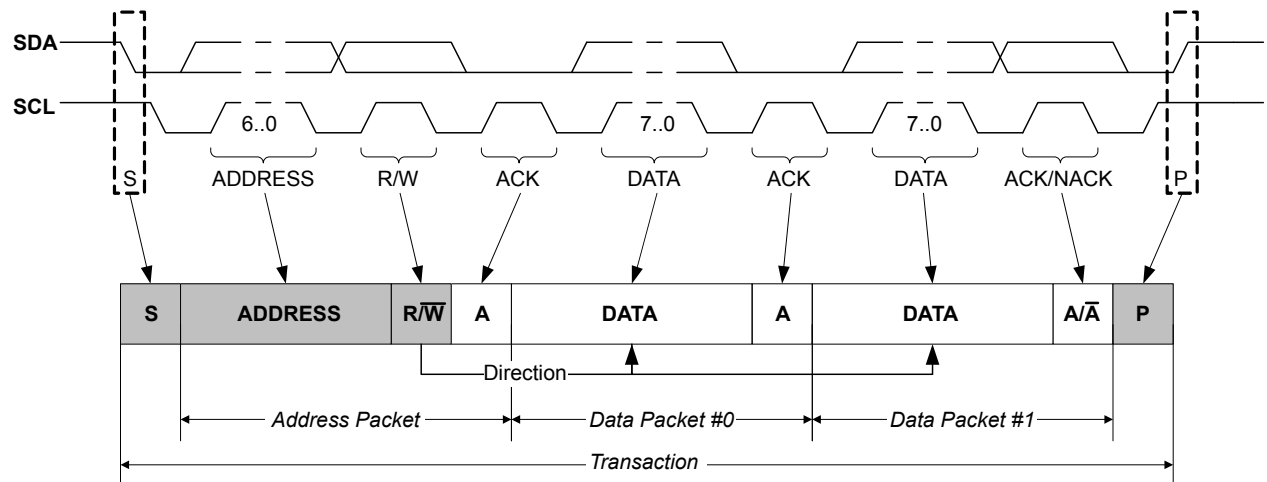
A transaction starts with the I<sup>2</sup>C master sending the start condition, followed by a 7-bit address and a direction bit (read or write to/from the slave).

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 34-2. Basic I<sup>2</sup>C Transaction Diagram



Transaction Diagram Symbols

Bus Driver

<div style="display: inline-block; width: 20px; height: 20px; background-color: #cccccc; border: 1px solid black; margin-bottom: 10px;"></div> Master driving bus
<div style="display: inline-block; width: 20px; height: 20px; background-color: #ffffff; border: 1px solid black; margin-bottom: 10px;"></div> Slave driving bus
<div style="display: inline-block; width: 20px; height: 20px; background-color: #ffffff; border: 1px solid black;"></div> Either Master or Slave driving bus

Special Bus Conditions

<div style="display: inline-block; width: 20px; height: 20px; background-color: #cccccc; border: 1px solid black; text-align: center; font-weight: bold; margin-bottom: 10px;">S</div> START condition
<div style="display: inline-block; width: 20px; height: 20px; background-color: #cccccc; border: 1px solid black; text-align: center; font-weight: bold; margin-bottom: 10px;">Sr</div> repeated START condition
<div style="display: inline-block; width: 20px; height: 20px; background-color: #cccccc; border: 1px solid black; text-align: center; font-weight: bold; margin-bottom: 10px;">P</div> STOP condition

Data Package Direction

<div style="display: inline-block; width: 20px; height: 20px; background-color: #cccccc; border: 1px solid black; text-align: center; font-weight: bold; margin-bottom: 10px;">R</div> Master Read '1'
<div style="display: inline-block; width: 20px; height: 20px; background-color: #cccccc; border: 1px solid black; text-align: center; font-weight: bold; margin-bottom: 10px;">W</div> Master Write '0'

Acknowledge

<div style="display: inline-block; width: 20px; height: 20px; background-color: #ffffff; border: 1px solid black; text-align: center; font-weight: bold; margin-bottom: 10px;">A</div> Acknowledge (ACK) '0'
<div style="display: inline-block; width: 20px; height: 20px; background-color: #ffffff; border: 1px solid black; text-align: center; font-weight: bold; margin-bottom: 10px;">Ā</div> Not Acknowledge (NACK) '1'

### 34.6.2. Basic Operation

#### 34.6.2.1. 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 or 0x5 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.

#### 34.6.2.2. Enabling, Disabling, and Resetting

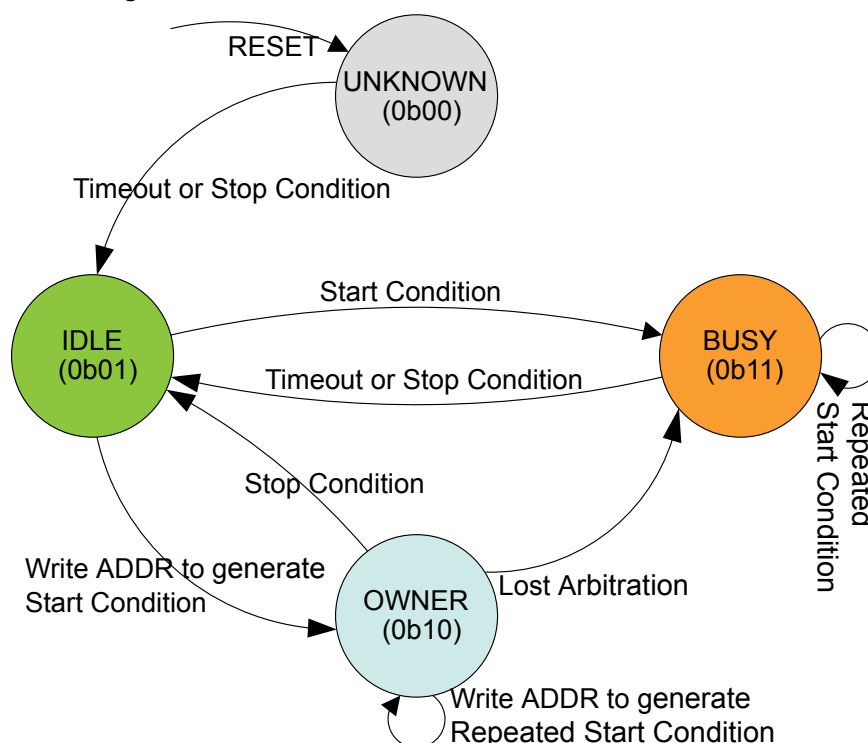
This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Refer to [CTRLA](#) on page 623 for details.

#### 34.6.2.3. 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. 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 34-3. 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.

Regardless of winning or losing arbitration, the entire address will be sent. If arbitration is lost, only 'ones' are transmitted from the point of losing arbitration and the rest of the address length.

#### 34.6.2.4. 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 events. The software driver complexity and code size are

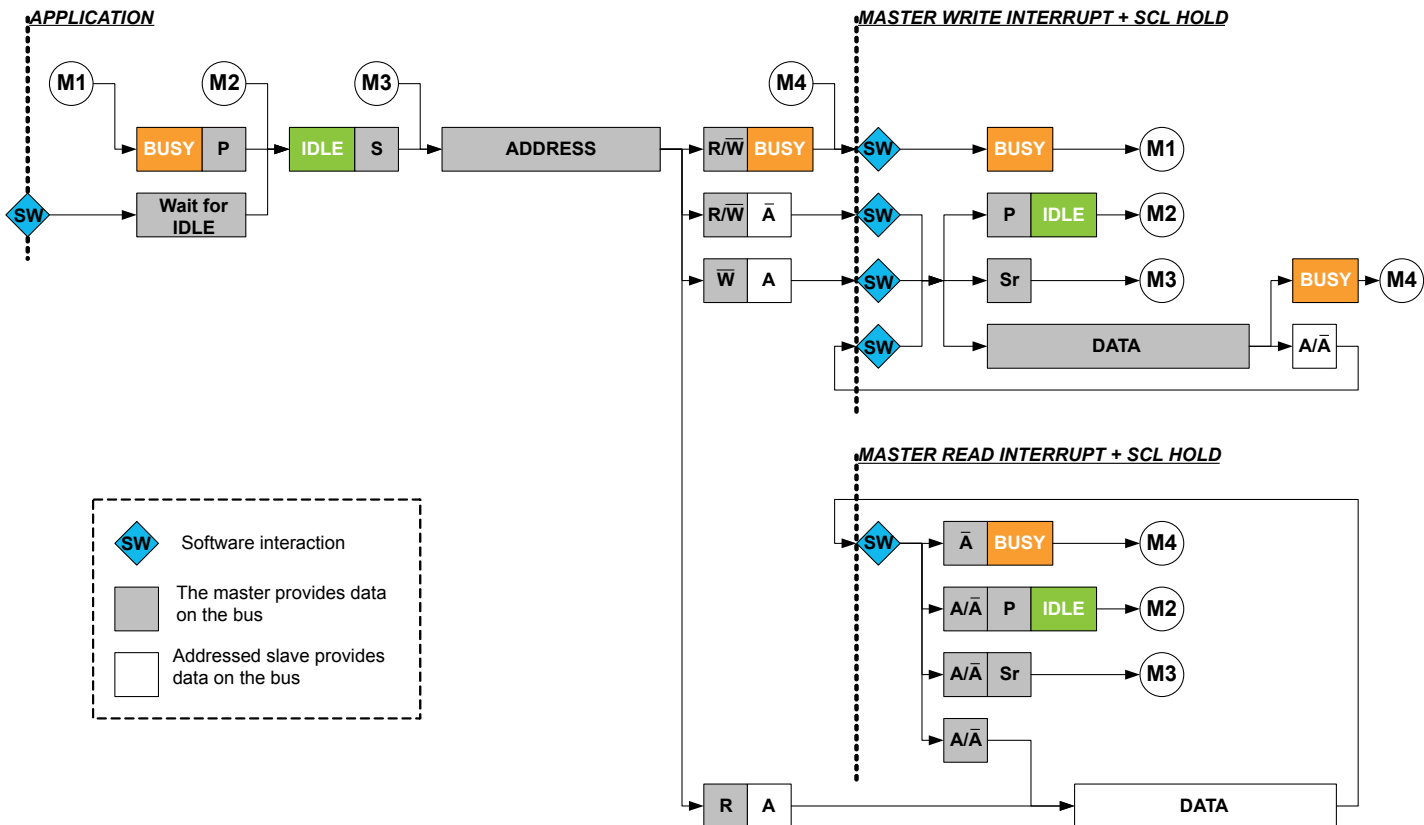
reduced by auto-triggering of operations, and a special smart mode, which can be enabled by the Smart Mode Enable bit in the Control A register (CTRLA.SMEN).

The I<sup>2</sup>C 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.

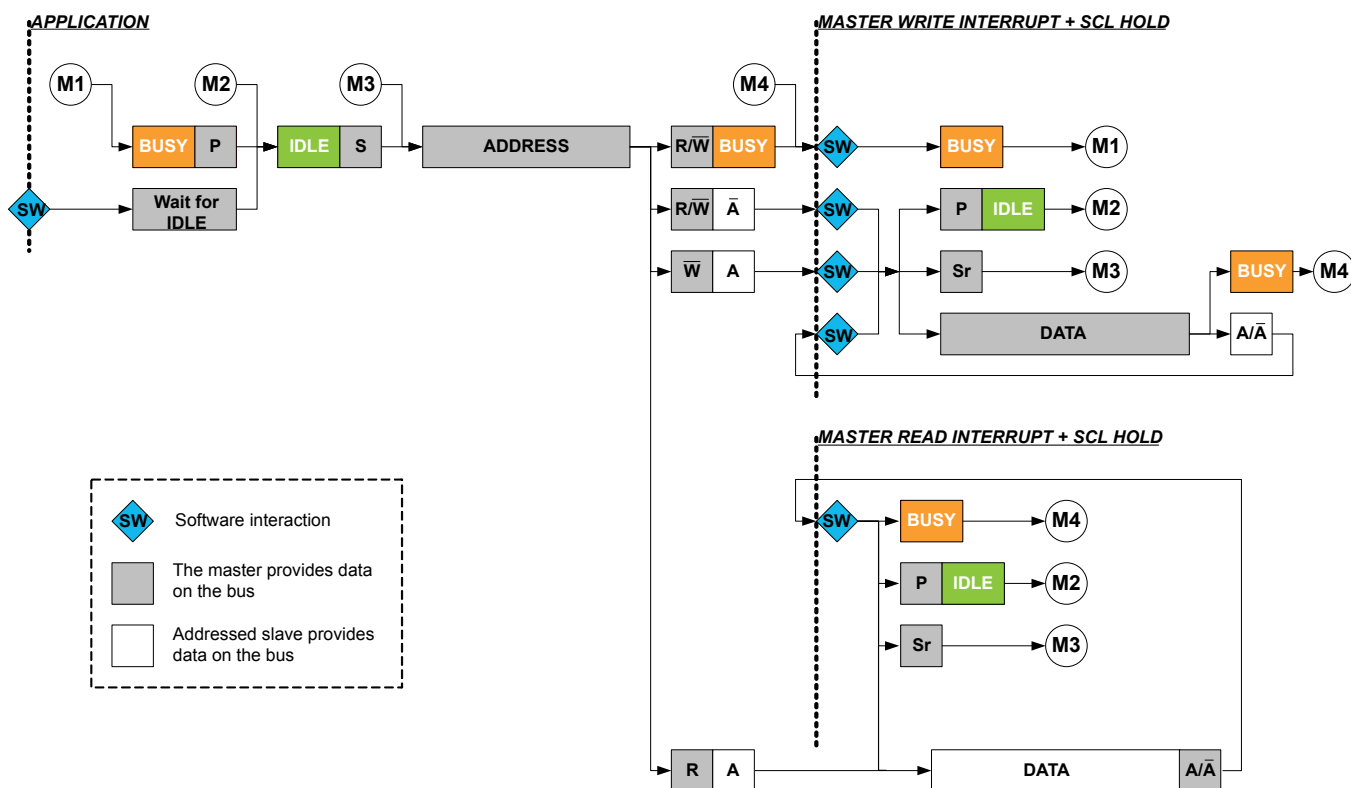
Figure 34-4. I<sup>2</sup>C Master Behavioral Diagram (SCLSM=0)



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 34-5. I<sup>2</sup>C Master Behavioral Diagram (SCLSM=1)



### Master Clock Generation

The SERCOM peripheral supports several I<sup>2</sup>C bi-directional 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

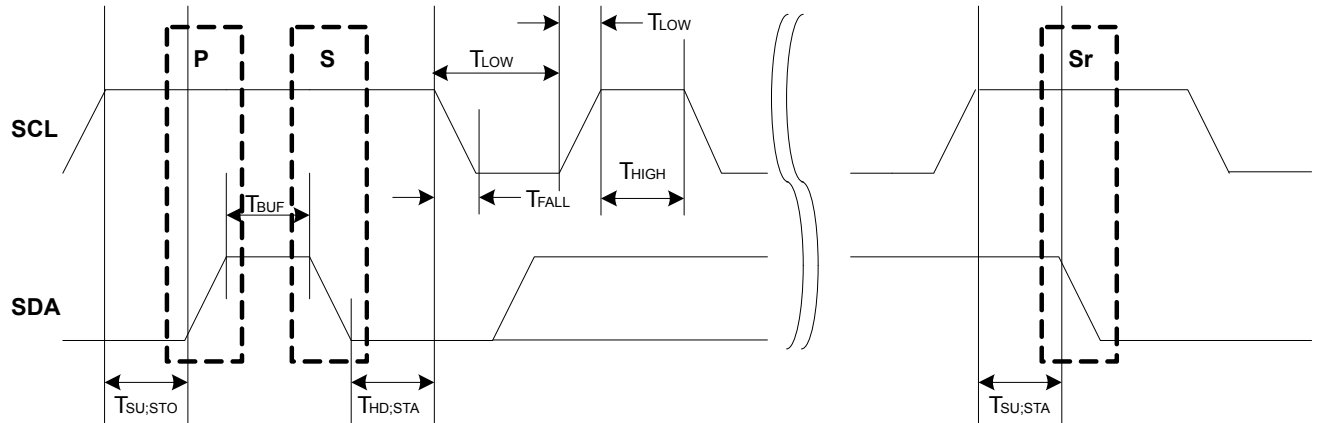
The Master clock configuration for *Sm*, *Fm*, and *Fm+* are described in [Clock Generation \(Standard-Mode, Fast-Mode, and Fast-Mode Plus\)](#) on page 587. For *Hs*, refer to [Master Clock Generation \(High-Speed Mode\)](#) on page 589.

### 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 34-6. 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.



## Related Links

[Electrical Characteristics](#) on page 1105

### **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.

$$f_{SCL} = \frac{f_{GCLK}}{2 + 2 \cdot HSBAUD}$$

When HSBAUDLOW is non-zero, the following formula determines the SCL frequency.

$$f_{SCL} = \frac{f_{GCLK}}{2 + HSBAUD + HSBAUDLOW}$$

**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](#) on page 582. 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 34-2 Basic I2C Transaction Diagram](#) on page 583) 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.

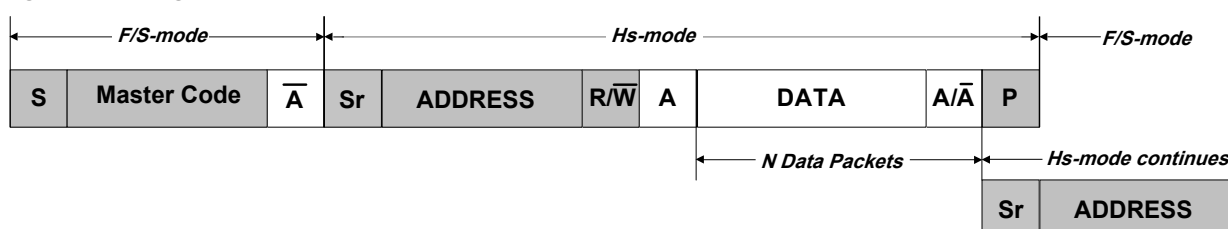
### High-Speed Mode

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 34-7. High Speed Transfer**



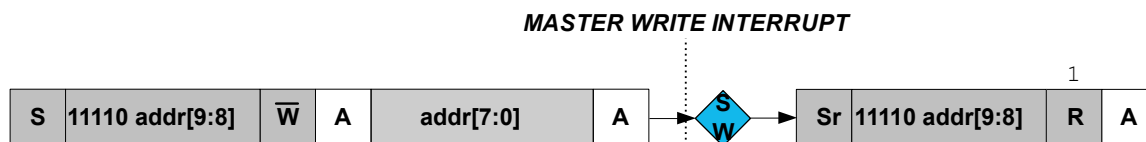
Transmitting in High-speed mode requires the I<sup>2</sup>C master to be configured in High-speed mode (CTRLA.SPEED=0x2) and the SCL clock stretch mode (CTRLA.SCLSM) bit set to '1'.

### 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 34-8. 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.

### 34.6.2.5. I<sup>2</sup>C Slave Operation

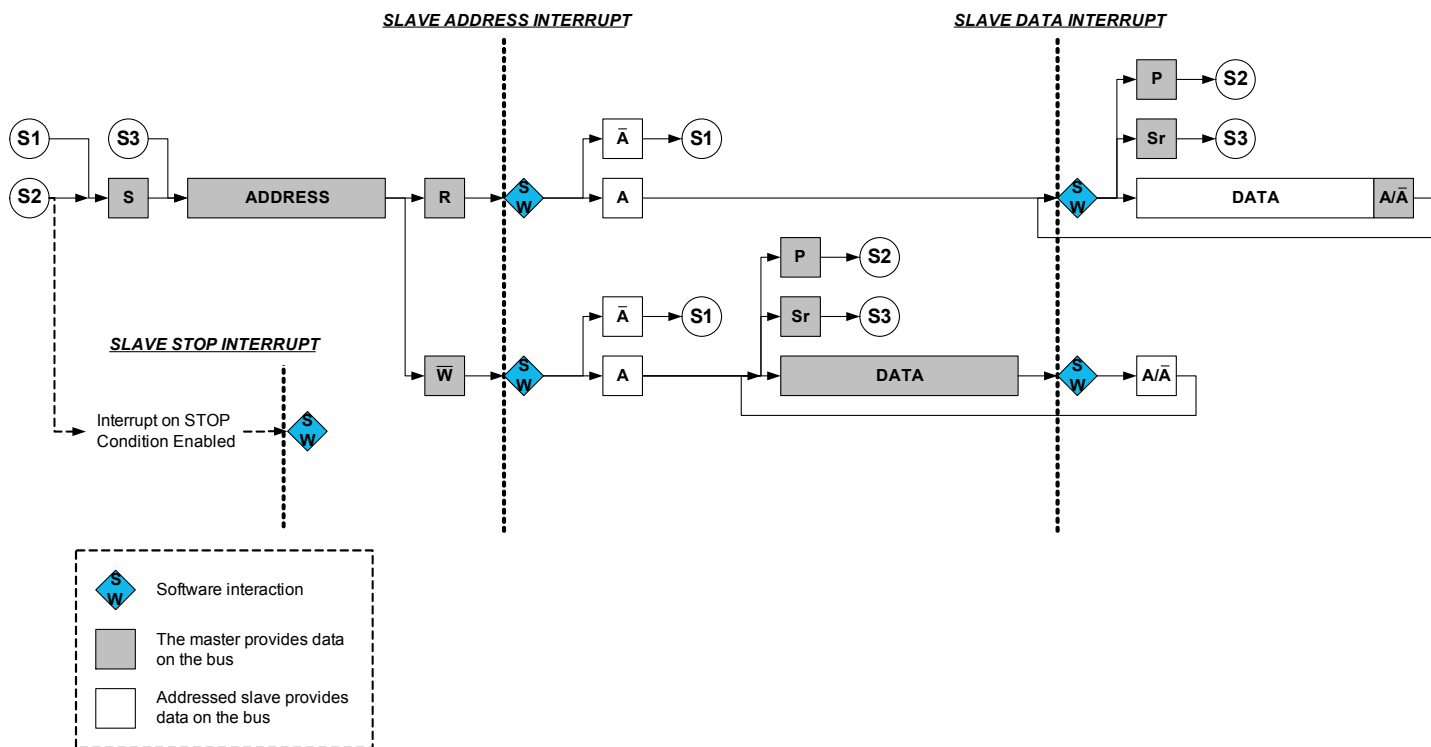
The I<sup>2</sup>C slave is byte-oriented and interrupt-based. The number of interrupts generated is kept at a minimum by automatic handling of most events. The software driver complexity and code size are reduced by auto-triggering of operations, and a special smart mode, which can be enabled by the 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.

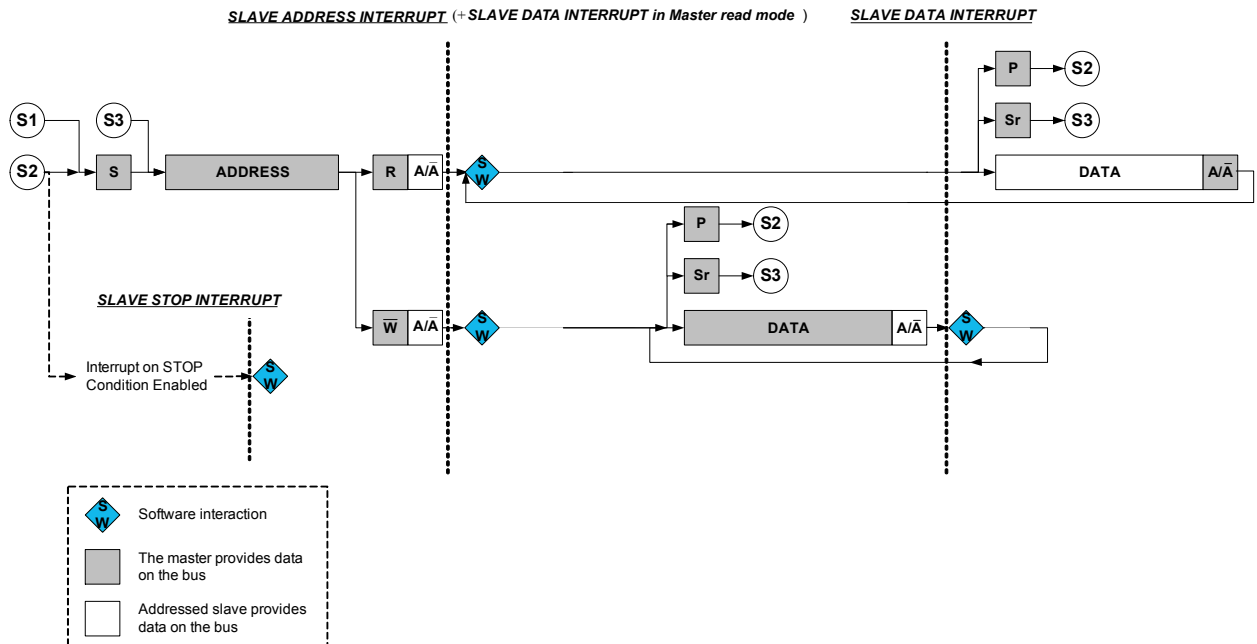
**Figure 34-9. I<sup>2</sup>C Slave Behavioral Diagram (SCLSM=0)**



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 34-10. Slave Behavioral Diagram (SCLSM=1)**



### Receiving Address Packets (SCLSM=0)

When CTRLA.SCLSM=0, the I2C slave stretches the SCL line according to [Figure 34-9 I2C Slave Behavioral Diagram \(SCLSM=0\)](#) on page 592. When the I2C 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 I2C 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 I2C slave clears INTFLAG.AMATCH. As the I2C 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 I2C 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 I2C 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 I2C master read operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, I2C 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 I2C 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 I2C 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.

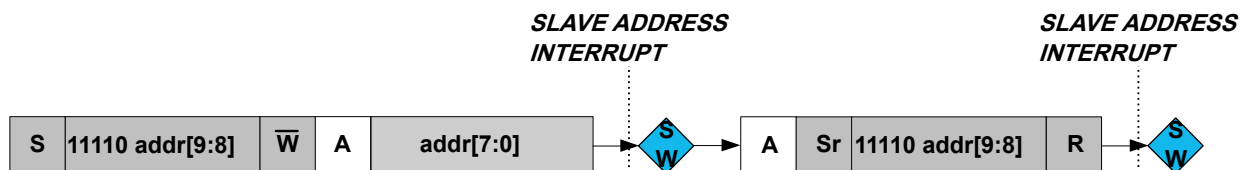
### 10-Bit Addressing

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 if it was addressed by the previous 10-bit address.

**Figure 34-11. 10-bit Addressing**



### PMBus Group Command

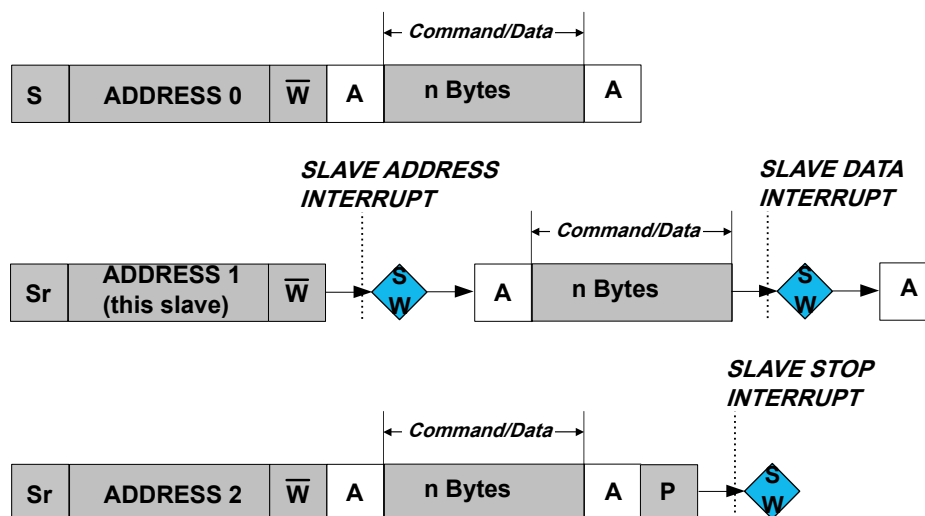
When the PMBus Group Command bit in the CTRLB register is set (CTRLB.GCMD=1) and 7-bit addressing is used, INTFLAG.PREC will be set when a STOP condition is detected on the bus. When CTRLB.GCMD=0, a STOP condition without address match will not be set INTFLAG.PREC.

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 34-12. PMBus Group Command Example



### 34.6.3. Additional Features

#### 34.6.3.1. 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<sub>TIMEOUT</sub>: SCL low time of 25..35ms – Measured for a single SCL low period. It is enabled by CTRLA.LOWTOUTEN.
- T<sub>LOW:SEXT</sub>: 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<sub>LOW:MEXT</sub>: 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.

#### 34.6.3.2. Smart Mode

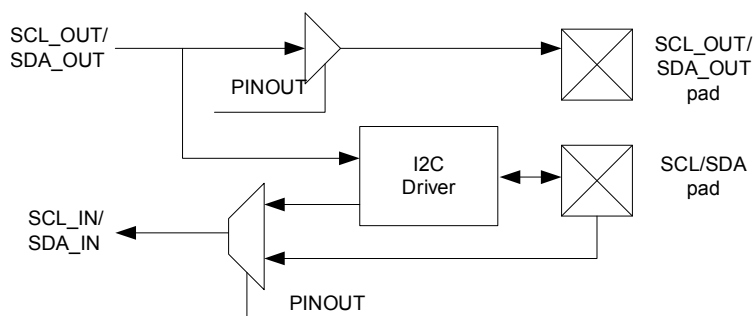
The I<sup>2</sup>C interface has a smart mode that simplifies application code and minimizes the user interaction needed to adhere to the I<sup>2</sup>C protocol. The smart mode accomplishes this by automatically issuing an ACK or NACK (based on the content of CTRLB.ACKACT) as soon as DATA.DATA is read.

#### 34.6.3.3. 4-Wire Mode

Selecting 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 34-13. I<sup>2</sup>C Pad Interface**



#### 34.6.3.4. 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.

#### 34.6.4. DMA, Interrupts and Events

**Table 34-1. 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 34-2. 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	

**34.6.4.1. 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:

- Write data received (RX): The request is set when master write data is received. The request is cleared when DATA is read.
- Read data needed for transmit (TX): The request is set when data is needed for a master read operation. The request is cleared when DATA is written.

**Master DMA**

When using the 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:

- Read data received (RX): The request is set when master read data is received. The request is cleared when DATA is read.
- Write data needed for transmit (TX): The request is set when data is needed for a master write operation. The request is cleared when DATA is written.

**34.6.4.2. 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 [INTFLAG](#) on page 633 register for details on how to clear interrupt flags.

The I<sup>2</sup>C has one common interrupt request line for all the interrupt sources. 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](#) on page 41

#### 34.6.4.3. Events

Not applicable.

#### 34.6.5. 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 GCLK\_SERCOMx\_CORE will also run in standby sleep mode. Any interrupt can wake up the device.

If CTRLA.RUNSTDBY=0, the GCLK\_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.

#### 34.6.6. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

The following bits are synchronized when written:

- Software Reset bit in the CTRLA register (CTRLA.SWRST)

- Enable bit in the CTRLA register (CTRLA.ENABLE)
- 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](#) on page 110

## 34.7. Register Summary - I2C Slave

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY				MODE[2:0]		ENABLE	SWRST
0x01		15:8								
0x02		23:16	SEXTTOEN			SDAHOLD[1:0]				PINOUT
0x03		31:24		LOWTOUT			SCLSM			SPEED[1:0]
0x04	CTRLB	7:0								
0x05		15:8		AMODE[1:0]				AACKEN	GCMD	SMEN
0x06		23:16						ACKACT		CMD[1:0]
0x07		31:24								
0x08	Reserved									
...										
0x13										
0x14		INTENCLR	7:0	ERROR					DRDY	AMATCH
0x15	Reserved									
0x16	INTENSET	7:0	ERROR					DRDY	AMATCH	PREC
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR					DRDY	AMATCH	PREC
0x19	Reserved									
0x1A	STATUS	7:0	CLKHOLD	LOWTOUT		SR	DIR	RXNACK	COLL	BUSERR
0x1B		15:8						LENERR	SEXTTOUT	
0x1C	SYNDBUSY	7:0							ENABLE	SWRST
0x1D		15:8								
0x1E		23:16								
0x1F		31:24								
0x20	Reserved									
...										
0x23										
0x24		ADDR	7:0					ADDR[6:0]		
0x25	15:8		TENBITEN						ADDR[9:7]	
0x26	23:16						ADDRMASK[6:0]			
0x27	31:24								ADDRMASK[9:7]	
0x28	DATA	7:0					DATA[7:0]			
0x29		15:8								

## 34.8. Register Description - I2C 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](#) on page 582.

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](#) on page 599.

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:** 0x00000000

**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

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 34-5 I2C Master Behavioral Diagram (SCLSM=1)</a> on page 587
1	SCL stretch only after ACK bit according to <a href="#">Figure 34-9 I2C Slave Behavioral Diagram (SCLSM=0)</a> on page 592

#### 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 CTRL.ENABLE will read back immediately and the Enable Synchronization Busy bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

This bit is not enable-protected.

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.

## 34.8.2. 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						ACKACT	CMD[1:0]	
Reset						R/W	R/W	R/W
Reset						0	0	0
Bit	15	14	13	12	11	10	9	8
Access	AMODE[1:0]					AACKEN	GCMD	SMEN
Reset	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
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 34-3. 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_ADDRS	The slave responds to the two unique addresses in ADDR.ADDR and ADDR.ADDRMASK.
0x2	RANGE	The slave responds to the range of addresses between and including ADDR.ADDR and ADDR.ADDRMASK. ADDR.ADDR is the upper limit.
0x3	-	Reserved.

**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 Received 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.

### 34.8.3. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x14

**Reset:** 0x00

**Property:** PAC Write-Protection

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.

### 34.8.4. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x16

**Reset:** 0x00

**Property:** PAC Write-Protection

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.



### 34.8.5. 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.

### 34.8.6. Status

**Name:** STATUS  
**Offset:** 0x1A  
**Reset:** 0x0000  
**Property:** -

Bit	15	14	13	12	11	10	9	8
						LENERR	SEXTTOUT	
Access						R/W	R/W	
Reset						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 10 – 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.

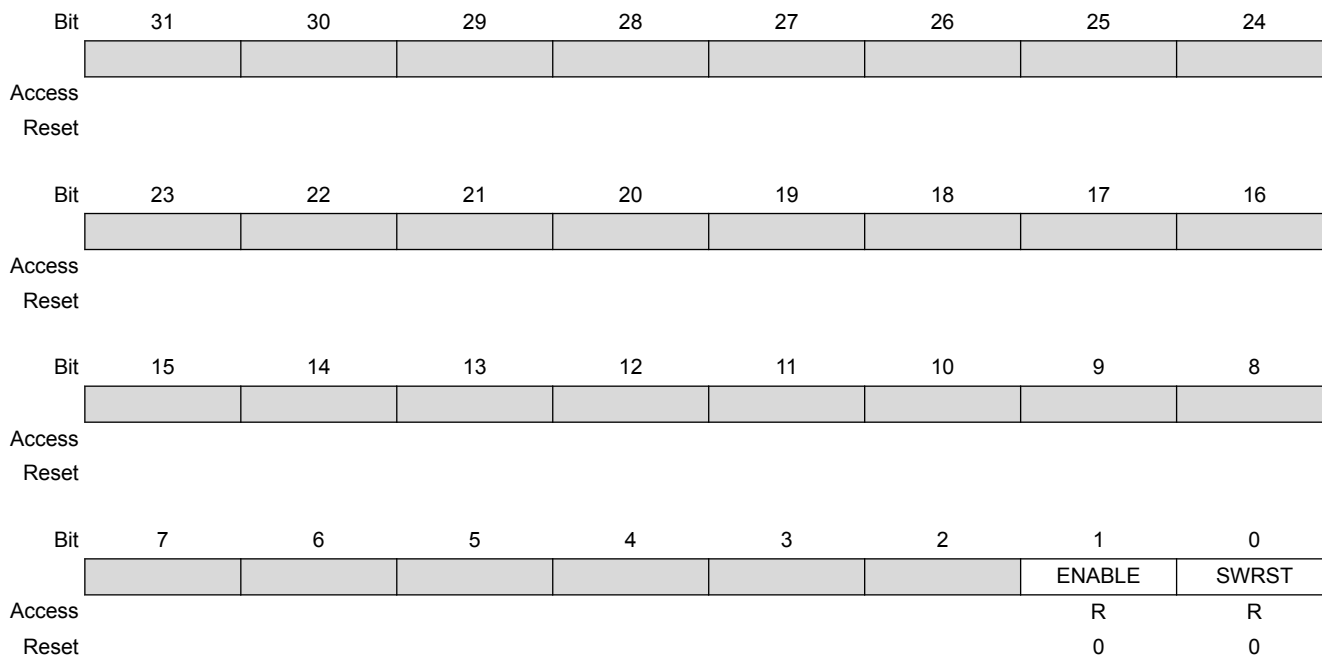
### 34.8.7. Synchronization Busy

**Name:** SYNCBUSY

**Offset:** 0x1C

**Reset:** 0x00000000

**Property:**



#### Bit 1 – ENABLE: SERCOM Enable Synchronization Busy

Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. When written, the SYNCBUSY.ENABLE bit will be set until synchronization is complete.

Writes to any register (except for CTRLA.SWRST) while enable synchronization is on-going will be discarded and an APB error will be generated.

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.

Writes to any register while synchronization is on-going will be discarded and an APB error will be generated.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

### 34.8.8. 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]								
Access	R/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.

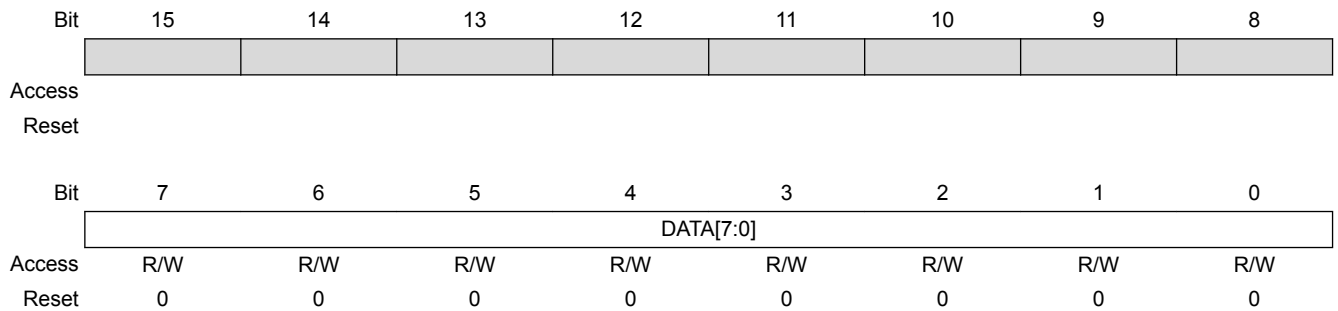
#### Bit 0 – GENCEN: General Call Address Enable

A general call address is an address consisting of all-zeroes, including the direction bit (master write).

Value	Description
0	General call address recognition disabled.
1	General call address recognition enabled.

### 34.8.9. 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.



## 34.9. Register Summary - I2C Master

Offset	Name	Bit Pos.							
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]		ENABLE	SWRST
0x01		15:8							
0x02		23:16	SEXTTOEN	MEXTTOEN	SDAHOLD[1:0]				PINOUT
0x03		31:24		LOWTOUT	INACTOUT[1:0]	SCLSM			SPEED[1:0]
0x04	CTRLB	7:0							
0x05		15:8						QCEN	SMEN
0x06		23:16					ACKACT		CMD[1:0]
0x07		31:24							
0x08 ... 0x0B	Reserved								
0x0C	BAUD	7:0	BAUD[7:0]						
0x0D		15:8	BAUDLOW[7:0]						
0x0E		23:16	HSBAUD[7:0]						
0x0F		31:24	HSBAUDLOW[7:0]						
0x10 ... 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]						
0x19		15:8							
0x1A	STATUS	7:0	CLKHOLD	LOWTOUT	BUSSTATE[1:0]		RXNACK	ARBLOST	BUSERR
0x1B		15:8					LENERR	SEXTTOUT	MEXTTOUT
0x1C	SYNDBUSY	7:0					SYSOP	ENABLE	SWRST
0x1D		15:8							
0x1E		23:16							
0x1F		31:24							
0x21 ... 0x23	Reserved								
0x24	ADDR	7:0							
0x25		15:8	TENBITEN	HS	LENEN			ADDR[2:0]	
0x26		23:16	LEN[7:0]						
0x27		31:24							
0x28 ... 0x2F	Reserved								
0x30	DBGCTRL	7:0							DBGSTOP

## 34.10. Register Description - I2C 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](#) on page 582.

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](#) on page 599.

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.

### 34.10.1. Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized

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 34-5 I2C Master Behavioral Diagram (SCLSM=1)</a> on page 587.
1	SCL stretch only after ACK bit.

#### 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 CTRLA.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.

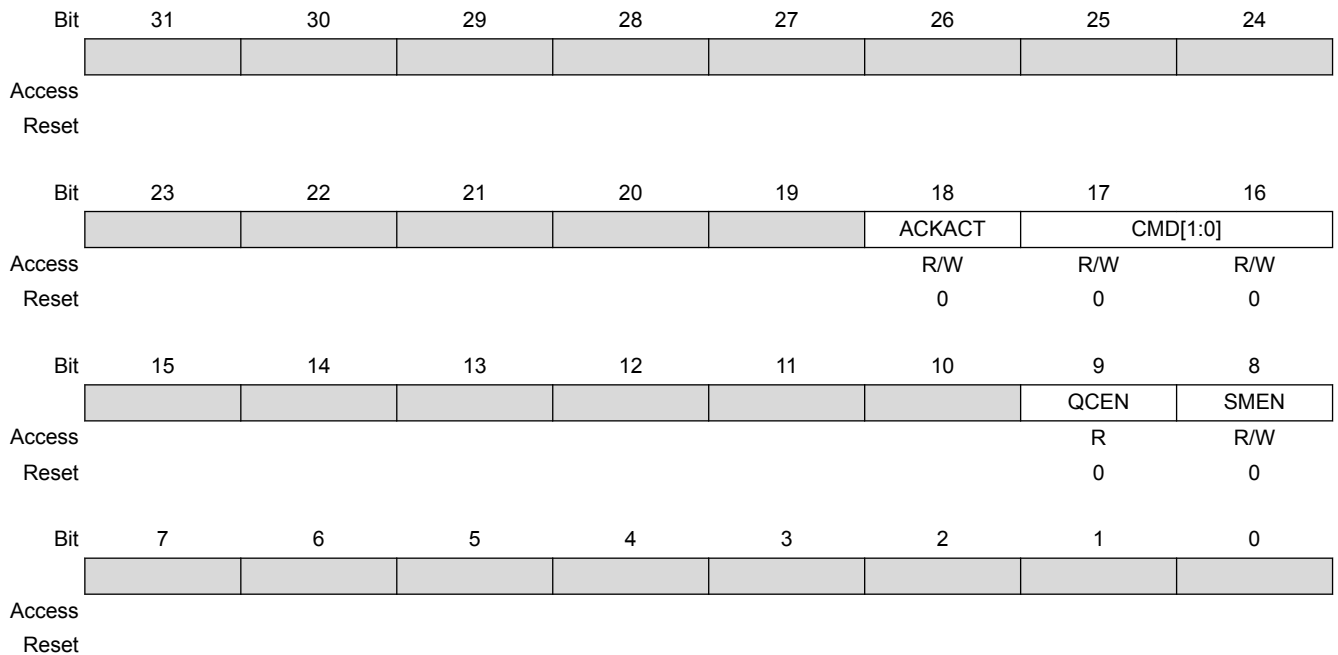
Due to synchronization there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.

This bit is not enable-protected.

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

### 34.10.2. Control B

**Name:** CTRLB  
**Offset:** 0x04  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized



#### 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 34-4. 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.



### 34.10.3. 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](#) on page 501.

#### 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](#) on page 501.

#### 34.10.4. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x14

**Reset:** 0x00

**Property:** PAC Write-Protection

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.

### 34.10.5. Interrupt Enable Clear

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

**Name:** INTENSET

**Offset:** 0x16

**Reset:** 0x00

**Property:** PAC Write-Protection

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.

### 34.10.6. 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.

### 34.10.7. 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	R		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.

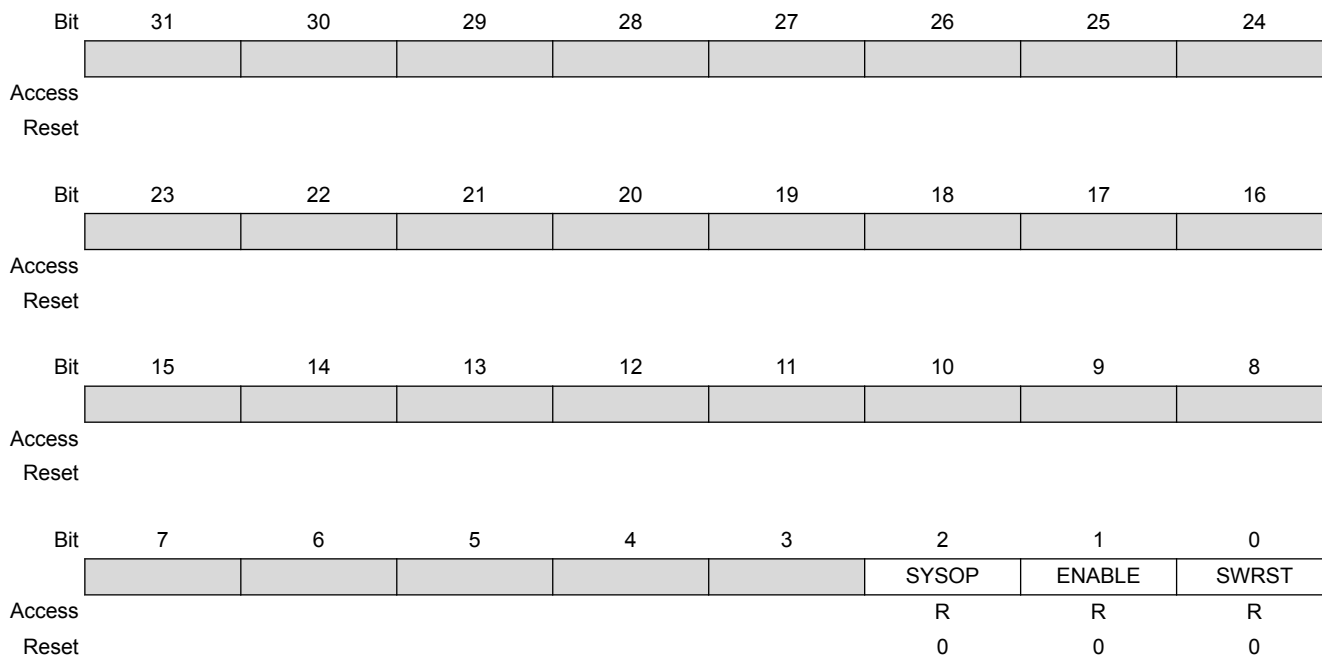
Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

This bit is not write-synchronized.

### 34.10.8. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x1C  
**Reset:** 0x00000000  
**Property:**



#### Bit 2 – SYSOP: System Operation Synchronization Busy

Writing CTRLB.CMD, STATUS.BUSSTATE, ADDR, or DATA when the SERCOM is enabled requires synchronization. When written, the SYNCBUSY.SYSOP bit will be set until synchronization is complete.

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.

Writes to any register (except for CTRLA.SWRST) while enable synchronization is on-going will be discarded and an APB error will be generated.

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.

Writes to any register while synchronization is on-going will be discarded and an APB error will be generated.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

### 34.10.9. 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
Access	LEN[7:0]							
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	TENBITEN	HS	LENEN			ADDR[2:0]		
Reset	0	0	0			0	0	0
Bit	7	6	5	4	3	2	1	0
Access								
Reset								

#### 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 10:8 – ADDR[2:0]: 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.

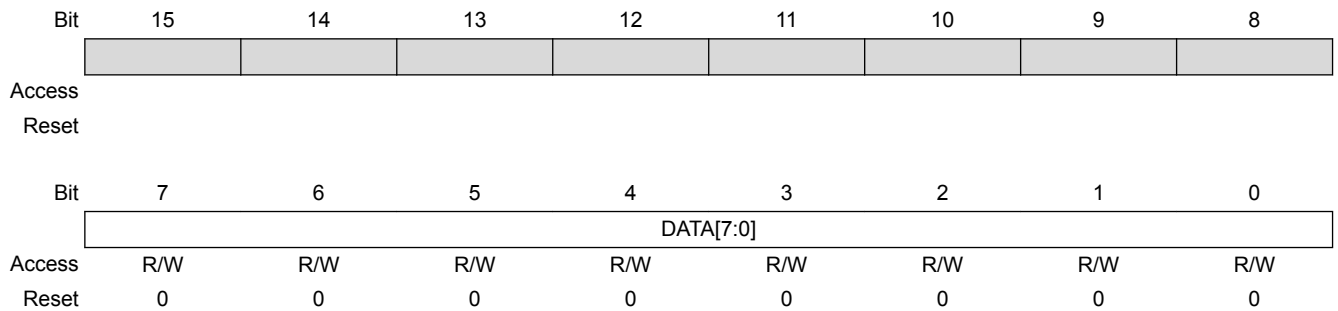
STATUS.BUSERR, STATUS.ARBLOST, INTFLAG.MB and INTFLAG.SB will be cleared when ADDR is written.

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.

### 34.10.10. 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.

### 34.10.11. 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.

## 35. CAN - Control Area Network

### 35.1. Overview

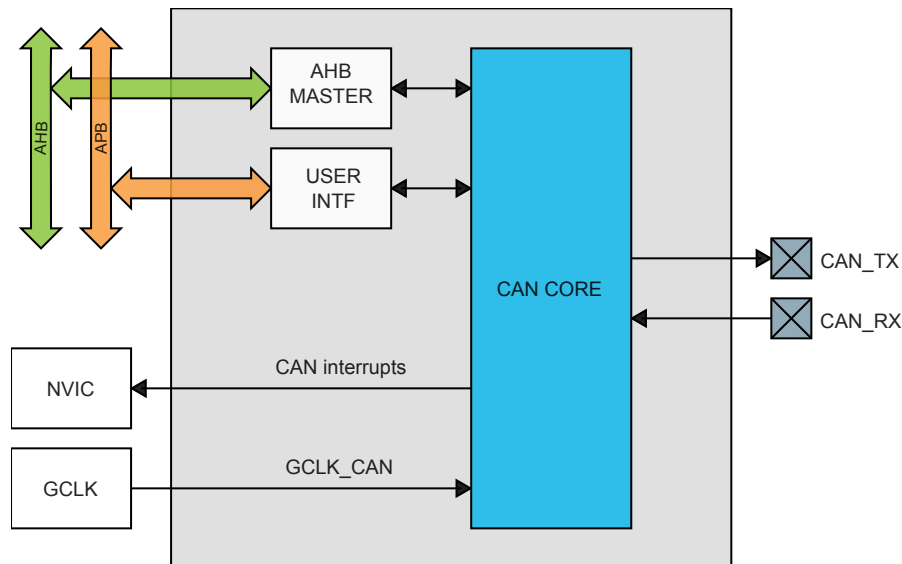
The Control Area Network (CAN) performs communication according to ISO 11898-1 (Bosch CAN specification 2.0 part A,B) and to Bosch CAN FD specification V1.0. The message storage is intended to be a single- or dual-ported Message RAM outside of the module.

### 35.2. Features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1
- CAN FD with up to 64 data bytes supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers and up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO, Transmit Queue, Transmit Event FIFO
- Direct Message RAM access for CPU
- Programmable loop-back test mode
- Maskable module interrupts
- Power-down support; Debug on CAN support

### 35.3. Block Diagram

Figure 35-1. CAN Block Diagram





## 35.4. Signal Description

Table 35-1. Signal Description

Signal	Description	Type
CAN_TX	CAN transmit	Digital output
CAN_RX	CAN receive	Digital input

Refer to for details on the pin mapping for this peripheral. One signal can be mapped to one of several pins.

### Related Links

[I/O Multiplexing and Considerations](#) on page 27

## 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 CAN's I/O lines requires the I/O pins to be configured.

### Related Links

[PORT - I/O Pin Controller](#) on page 444

### 35.5.2. Power Management

The CAN will continue to operate in any sleep mode where the selected source clock is running. The CAN interrupts can be used to wake up the device from sleep modes. Refer to the Power Manager chapter for details on the different sleep modes.

### 35.5.3. Clocks

The CAN bus clock (CLK\_CAN\_APB) can be enabled and disabled in the Main Clock module, and the default state of CLK\_CAN\_APB can be found in the Peripheral Clock Masking section.

A generic clock (GCLK\_CAN) is required to clock the CAN. This clock must be configured and enabled in the generic clock controller before using the CAN.

This generic clock is asynchronous to the bus clock (CLK\_CAN\_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains.

### Related Links

[Peripheral Clock Masking](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

### 35.5.4. DMA

The DMA request lines (or line if only one request) are connected to the DMA Controller (DMAC). Using the CAN DMA requests requires the DMA Controller to be configured first.

### Related Links

[DMAC – Direct Memory Access Controller](#) on page 334

### 35.5.5. Interrupts

The interrupt request lines are connected to the interrupt controller. Using the CAN interrupts requires the interrupt controller to be configured first.

## Related Links

[Nested Vector Interrupt Controller](#) on page 41

### 35.5.6. Events

Not applicable.

### 35.5.7. Debug Operation

Not applicable.

### 35.5.8. Register Access Protection

Not applicable.

### 35.5.9. Analog Connections

No analog connections.

## 35.6. Functional Description

### 35.6.1. Principle of Operation

The CAN performs communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 part A,B). In addition the CAN supports communication according to CAN FD specification V1.0.

The message storage is intended to be a single- or dual-ported Message RAM outside the module. It is connected to the CAN via AHB.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

### 35.6.2. Operating Modes

#### 35.6.2.1. Software Initialization

Software initialization is started by setting bit CCCR.INIT, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus\_Off. While CCCR.INIT is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output CAN\_TX is "recessive" (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR.INIT does not change any configuration register. Resetting CCCR.INIT finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive "recessive" bits (= Bus\_Idle) before it can take part in bus activities and start the message transfer.

Access to the CAN configuration registers is only enabled when both bits CCCR.INIT and CCCR.CCE are set (protected write).

CCCR.CCE can only be set/reset while CCCR.INIT = '1'. CCCR.CCE is automatically reset when CCCR.INIT is reset.

The following registers are reset when CCCR.CCE is set

- HPMS - High Priority Message Status
- RXF0S - Rx FIFO 0 Status
- RXF1S - Rx FIFO 1 Status
- TXFQS - Tx FIFO/Queue Status
- TXBRP - Tx Buffer Request Pending
- TXBTO - Tx Buffer Transmission Occurred
- TXBCF - Tx Buffer Cancellation Finished
- TXEFS - Tx Event FIFO Status

The Timeout Counter value TOCV.TOC is preset to the value configured by TOCC.TOP when CCCR.CCE is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR.CCE = '1'.

The following registers are only writable while CCCR.CCE = '0'

- TXBAR - Tx Buffer Add Request
- TXBCR - Tx Buffer Cancellation Request

CCCR.TEST and CCCR.MON can only be set by the CPU while CCCR.INIT = '1' and CCCR.CCE = '1'. Both bits may be reset at any time. CCCR.DAR can only be set/reset while CCCR.INIT = '1' and CCCR.CCE = '1'.

### 35.6.2.2. Normal Operation

Once the CAN is initialized and CCCR.INIT is reset to '0', the CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO0 or Rx FIFO1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

### 35.6.2.3. CAN FD Operation

There are two variants in the CAN FD frame format, first the CAN FD frame without bit rate switching where the data field of a CAN frame may be longer than 8 bytes. The second variant is the CAN FD frame where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the CAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled (CCCR.PXHD = '0'), this causes the operation state to change from Receiver (PSR.ACT = "10") to Integrating (PSR.ACT = "00") at the next sample point. In case Protocol Exception Handling is disabled (CCCR.PXHD = '1'), the CAN will treat a recessive res bit as a form error and will respond with an error frame.

CAN FD operation is enabled by programming CCCR.FDOE. In case CCCR.FDOE = '1', transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via

bit FDF in the respective Tx Buffer element. With `CCCR.FDOE = '0'`, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. `CCCR.FDOE` and `CCCR.BRSE` can only be changed while `CCCR.INIT` and `CCCR.CCE` are both set.

With `CCCR.FDOE = '0'`, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With `CCCR.FDOE = '1'` and `CCCR.BRSE = '0'`, only bit FDF of a Tx Buffer element is evaluated. With `CCCR.FDOE = '1'` and `CCCR.BRSE = '1'`, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Then all nodes switch back to Classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to the table below.

**Table 35-2. Coding of DLC in CAN FD**

DLC	9	10	11	12	13	14	15
Number of Data Bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register NBTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Data Bit Timing & Prescaler Register DBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (`GCLK_CAN`). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of  $4 t_q$ , the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

#### 35.6.2.4. Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin `CAN_TX` the CAN receives the transmitted data from its local CAN transceiver via pin `CAN_RX`. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the

transceiver loop delay, the delay compensation is introduced. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

#### Description

The CAN's protocol unit has implemented a delay compensation mechanism to compensate the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the CAN's transmit output CAN\_TX through the transceiver to the receive input CAN\_RX plus the transmitter delay compensation offset as configured by TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

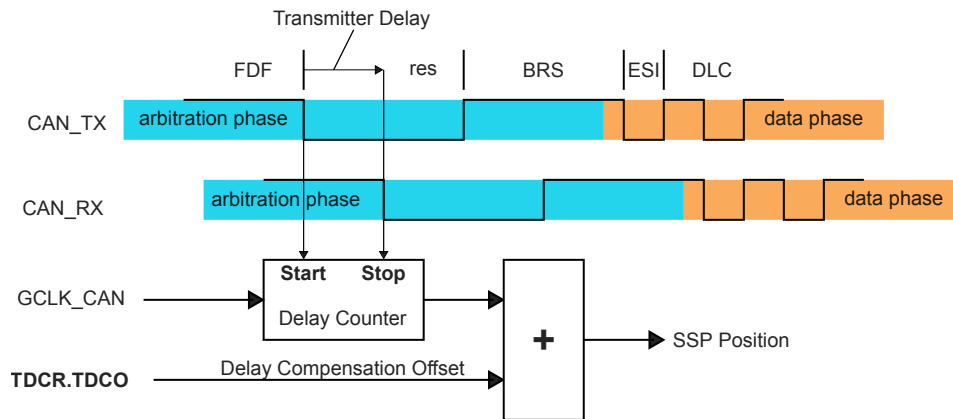
The following boundary conditions have to be considered for the transmitter delay compensation implemented in the CAN:

- The sum of the measured delay from CAN\_TX to CAN\_RX and the configured transceiver delay compensation offset FBTP.TDCO has to be less than 6 bit times in the data phase.
- The sum of the measured delay from CAN\_TX to CAN\_RX and the configured transceiver delay compensation offset FBTP.TDCO has to be less or equal to 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transceiver delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs.

#### Transmitter Delay Compensation Measurement

If transmitter delay compensation is enabled by programming DBTP.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input CAN\_TX of the transmitter. The resolution of this measurement is one mtq.

**Figure 35-2. Transceiver delay measurement**



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges of CAN\_RX, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCR.TDCF AND CAN\_RX is low.

#### 35.6.2.5. Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (ECR.REC, ECR.TEC) are frozen while Error Logging (ECR.CEL) is active. The CPU can set the CAN into Restricted Operation mode by setting bit CCCR.ASM. The bit can only be set by the CPU when both CCCR.CCE and CCCR.INIT are set to '1'. The bit can be reset by the CPU at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the CPU has to reset CCCR.ASM.

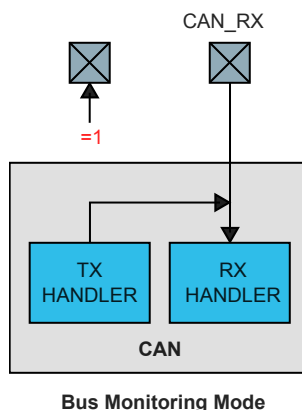
The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

#### 35.6.2.6. Bus Monitoring Mode

The CAN is set in Bus Monitoring Mode by programming CCCR.MON to '1'. In Bus Monitoring Mode (see ISO 11898-1, 10.12 Bus monitoring), the CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. The figure below shows the connection of signals CAN\_TX and CAN\_RX to the CAN in Bus Monitoring Mode.

**Figure 35-3. Pin Control in Bus Monitoring Mode**



#### 35.6.2.7. Disabled Automatic Retransmission

According to the CAN Specification (see ISO 11898-1, 6.3.3 Recovery Management), the CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR.DAR.

#### Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP.TRP<sub>x</sub> is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO.TO<sub>x</sub> set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF.CF<sub>x</sub> not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO.TO<sub>x</sub> set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF.CF<sub>x</sub> set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO.TO<sub>x</sub> not set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF.CF<sub>x</sub> set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

#### 35.6.2.8. Power Down (Sleep Mode)

The CAN can be set into power down mode via setting CC Control Register CCCR.CSR = '1'. When all pending transmission requests have completed, the CAN waits until bus idle state is detected. Then the CAN sets then CCCR.INIT to '1' to prevent any further CAN transfers. Now the CAN acknowledges that it is ready for power down by setting CCCR.CSA to '1'. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR.INIT will have no effect. Now the module clock inputs CLK\_CAN\_APB and GCLK\_CAN may be switched off.

To leave power down mode, the CPU has to turn on the module clocks before resetting CC Control Register CCCR.CSR = '0'. The CAN will acknowledge this by resetting CCCR.CSA = '0'. Afterwards, the application can restart CAN communication by resetting bit CCCR.INIT.



### 35.6.2.9. Test Modes

To enable write access to register TEST, bit CCCR.TEST has to be set to '1'. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin CAN\_TX by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the CAN's bit timing and it can drive constant dominant or recessive values. The actual value at pin CAN\_RX can be read from TEST.RX. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between GCLK\_CAN and GCLK\_CAN\_APB domains, there may be a delay of several GCLK\_CAN\_APB periods between writing to TEST.TX until the new configuration is visible at output pin CAN\_TX. This applies also when reading input pin CAN\_RX via TEST.RX.

Note: Test modes should be used for production tests or self test only. The software control for pin CAN\_TX interferes with all CAN protocol functions. It is not recommended to use test modes for application.

#### External Loop Back Mode

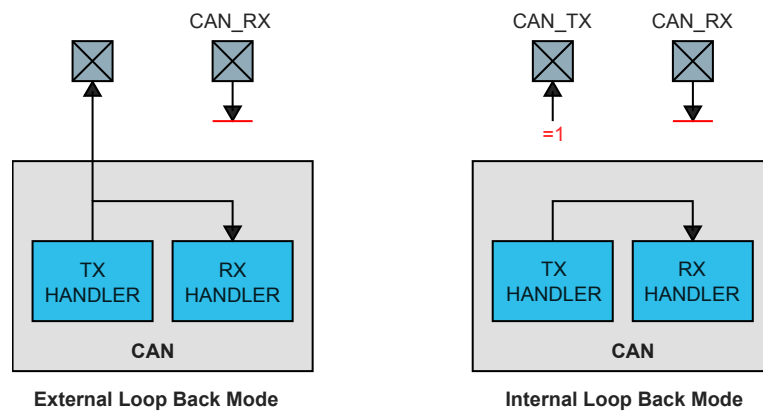
The CAN can be set in External Loop Back Mode by programming TEST.LBCK to '1'. In Loop Back Mode, the CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. The figure below shows the connection of signals CAN\_TX and CAN\_RX to the CAN in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN\_RX input pin is disregarded by the CAN. The transmitted messages can be monitored at the CAN\_TX pin.

#### Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits TEST.LBCK and CCCR.MON to '1'. This mode can be used for a "Hot Selftest", meaning the CAN can be tested without affecting a running CAN system connected to the pins CAN\_TX and CAN\_RX. In this mode pin CAN\_RX is disconnected from the CAN and pin CAN\_TX is held recessive. The figure below shows the connection of CAN\_TX and CAN\_RX to the CAN in case of Internal Loop Back Mode.

Figure 35-4. Pin Control in Loop Back Modes





### 35.6.3. Timestamp Generation

For timestamp generation the CAN supplies a 16-bit wrap-around counter. A prescaler TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1...16). The counter is readable via TSCV.TSC. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR.TSW is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS an external 16-bit timestamp can be used.

### 35.6.4. Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV.TOC. The Timeout Counter can only be started while CCCR.INIT = '0'. It is stopped when CCCR.INIT = '1', e.g. when the CAN enters Bus\_Off state.

The operation mode is selected by TOCC.TOS. When operating in Continuous Mode, the counter starts when CCCR.INIT is reset. A write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR.TOO is set. In Continuous Mode, the counter is immediately restarted at TOCC.TOP.

Note: The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

### 35.6.5. Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

#### 35.6.5.1. Acceptance Filtering

The CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration GFC
- Standard ID Filter Configuration SIDFC
- Extended ID Filter Configuration XIDFC
- Extended ID AND Mask XIDAM

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag IR.HPM
- Set High Priority Message interrupt flag IR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

#### Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.FLEC.

#### Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.FLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in [Rx FIFO Overwrite Mode](#) have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

### Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID for standard frames or EF1ID/EF2ID for extended frames.

There are two possibilities when range filtering is used together with extended frames:

**EFT = “00”** The Message ID of received frames is AND'ed with the Extended ID AND Mask (XIDAM) before the range filter is applied

**EFT = “11”** The Extended ID AND Mask (XIDAM) is not used for range filtering

### Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

### Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

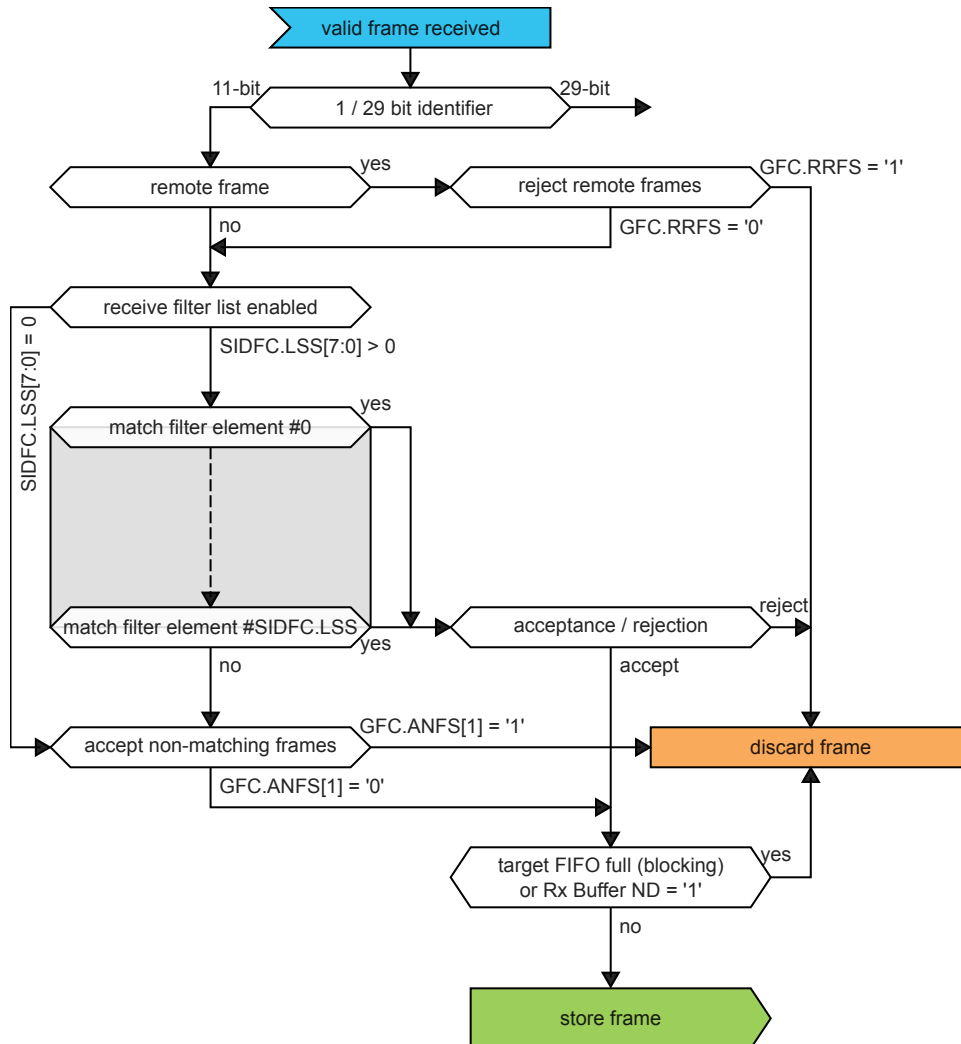
In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

### Standard Message ID Filtering

The figure below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in [Standard Message ID Filter Element](#) on page 754.

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

**Figure 35-5. Standard Message ID Filtering**



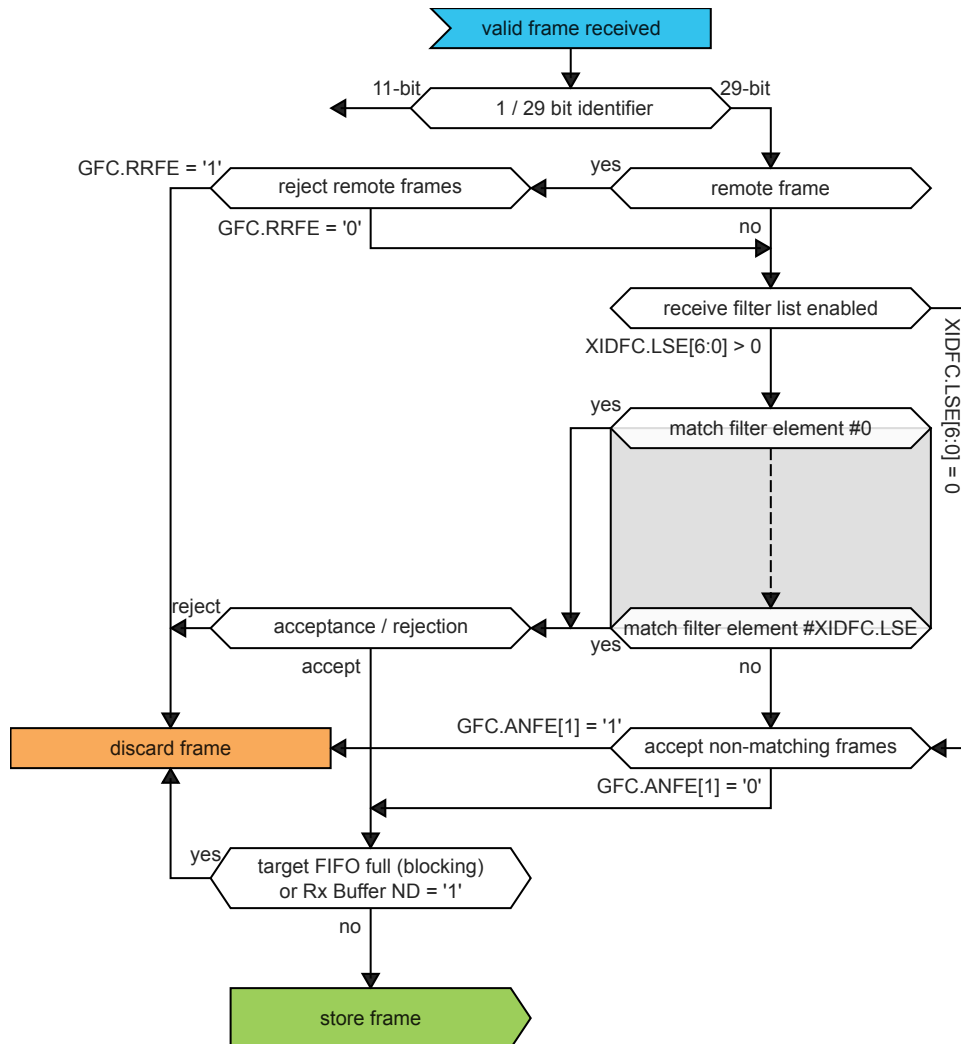
## Extended Message ID Filtering

The figure below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in [Extended Message ID Filter Element](#) on page 755.

Controlled by the Global Filter Configuration GFC and the Extended ID Filter Configuration XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask XIDAM is AND'ed with the received identifier before the filter list is executed.

**Figure 35-6. Extended Message ID Filtering**



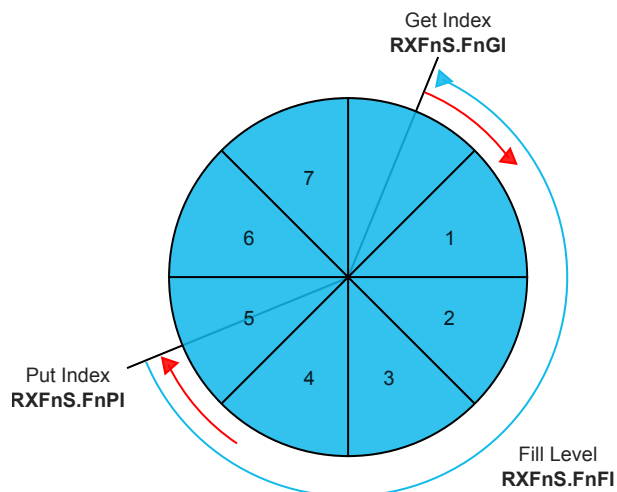
### 35.6.5.2. Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see [Acceptance Filtering](#) on page 653. The Rx FIFO element is described in [Rx Buffer and FIFO Element](#) on page 749.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by `RXFnC.FnWM`, interrupt flag `IR.RFnW` is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by `RXFnS.FnF`. In addition interrupt flag `IR.RFnF` is set.

**Figure 35-7. Rx FIFO Status**



When reading from an Rx FIFO, Rx FIFO Get Index `RXFnS.FnGI` • FIFO Element Size has to be added to the corresponding Rx FIFO start address `RXFnC.FnSA`.

**Table 35-3. Rx Buffer / FIFO Element Size**

<code>RXESC.RBDS[2:0]</code> <code>RXESC.FnDS[2:0]</code>	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

### Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by `RXFnC.FnOM = '0'`. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (`RXFnS.FnPI = RXFnS.FnGI`), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signaled by `RXFnS.FnF = '1'`. In addition interrupt flag `IR.RFnF` is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by `RXFnS.RFnL = '1'`. In addition interrupt flag `IR.RFnL` is set.

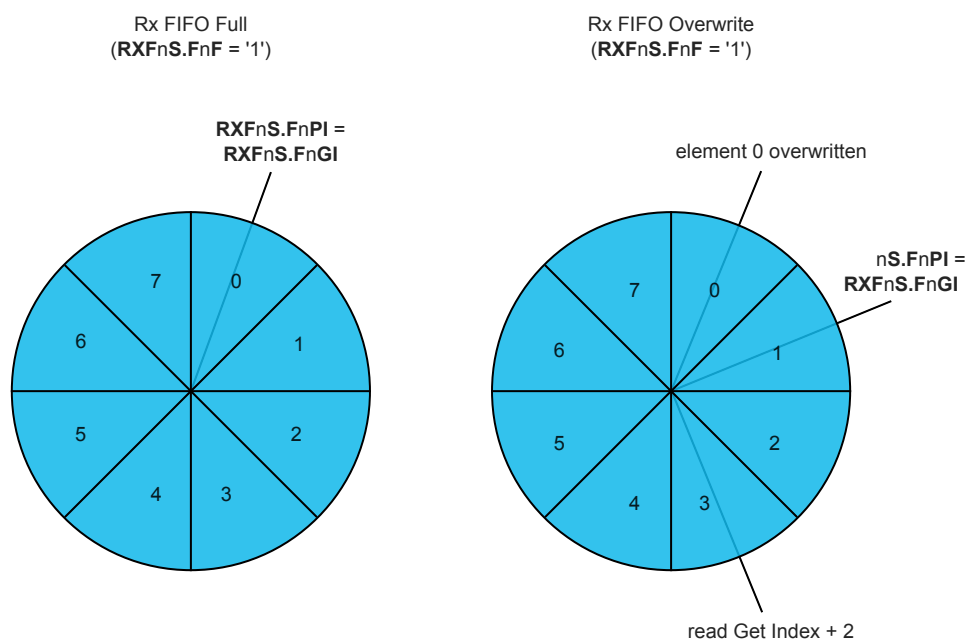
## Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by  $RXFnC.FnOM = '1'$ .

When an Rx FIFO full condition ( $RXFnS.FnPI = RXFnS.FnGI$ ) is signaled by  $RXFnS.FnF = '1'$ , the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. The figure below shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

**Figure 35-8. Rx FIFO Overflow Handling**



After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index  $RXFnA.FnA$ . This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset ( $RXFnS.FnF = '0'$ ).

### 35.6.5.3. Dedicated Rx Buffers

The CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via  $RXBC.RBSA$ .

For each Rx Buffer a Standard or Extended Message ID Filter Element with  $SFEC / EFEC = "111"$  and  $SFID2 / EFID2[10:9] = "00"$  has to be configured (see [Standard Message ID Filter Element](#) on page 754 and [Extended Message ID Filter Element](#) on page 755).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag  $IR.DRX$  (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

**Table 35-4. Example Filter Configuration for Rx Buffers**

Filter Element	SFID1[10:0] / EFID1[28:0]	SFID2[10:9] / EFID2[10:9]	SFID2[5:0] / EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1, NDAT2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the CPU by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

### Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

#### 35.6.5.4. Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (see [Rx Buffer and FIFO Element](#) on page 749 ).

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = "111" have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the CAN while DMA request is activated. The behavior is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets the DMA acknowledge. This resets DMA request. Now the CAN is prepared to receive the next set of debug messages.

### Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to "111". In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning (see [Standard Message ID Filter Element](#) on page 754 and [Extended Message ID Filter Element](#) on page 755). While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

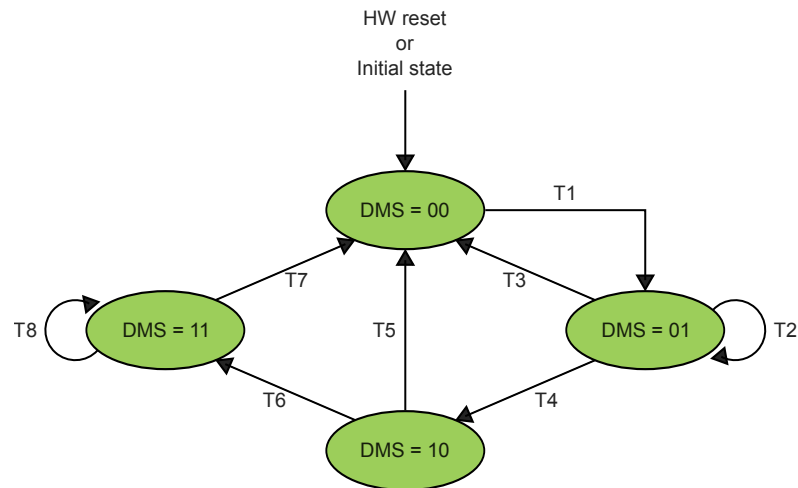
**Table 35-5. Example Filter Configuration for Debug Messages**

Filter Element	SFID1[10:0] / EFID1[28:0]	SFID2[10:9] / EFID2[10:9]	SFID2[5:0] / EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

### Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

**Figure 35-9. Debug Message Handling State Machine**



- T0: Reset DMA request output, enable reception of debug message A, B, and C
- T1: Reception of debug message A
- T2: Reception of debug message A
- T3: Reception of debug message C
- T4: Reception of debug message B
- T5: Reception of debug message A, B
- T6: Reception of debug message C
- T7: DMA transfer completed
- T8: Reception of debug message A, B, C (message rejected)

### 35.6.6. Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The CAN mode for transmission (Classic CAN or CAN FD) can be configured separately for each Tx Buffer element. The Tx Buffer element is described in [Tx Buffer Element](#) on page 751. The table below describes the possible configurations for frame transmission.



**Table 35-6. Possible Configurations for Frame Transmission**

CCCR		Tx Buffer Element		Frame Transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

#### 35.6.6.1. Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR.TXP. If the bit is set, the CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

#### 35.6.6.2. Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (refer to table below). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0...31) • Element Size to the Tx Buffer Start Address TXBC.TBSA.

**Table 35-7. Tx Buffer / FIFO / Queue Element Size**

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

**35.6.6.3. Tx FIFO**

Tx FIFO operation is configured by programming TXBC.TFQM to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS.TFGI. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The CAN calculates the Tx FIFO Free Level TXFQS.TFFL as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS.TFQPI. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS.TFQF = '1') is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is canceled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (refer to [Table 35-7 Tx Buffer / FIFO / Queue Element Size](#) on page 662). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0...31) • Element Size to the Tx Buffer Start Address TXBC.TBSA.

**35.6.6.4. Tx Queue**

Tx Queue operation is configured by programming TXBC.TFQM to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS.TFQPI. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS.TFQF = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

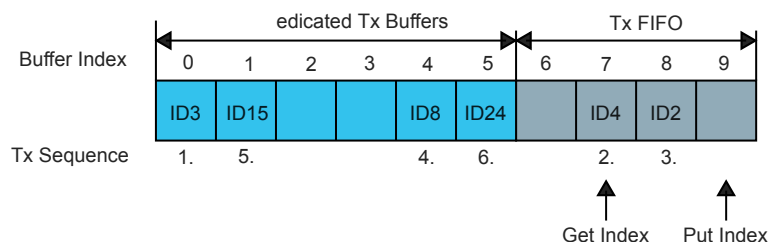
The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (refer to [Table 35-7 Tx Buffer / FIFO / Queue Element Size](#) on page 662). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0...31) • Element Size to the Tx Buffer Start Address TXBC.TBSA.

### 35.6.6.5. Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC.TFQS. In case TXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

**Figure 35-10. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO**



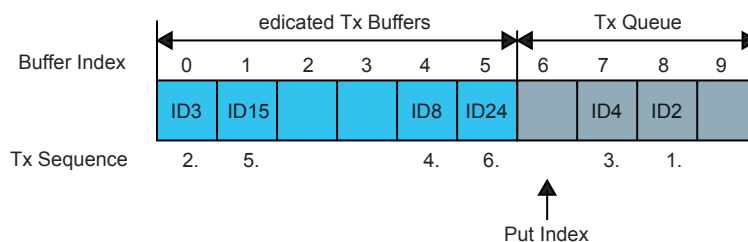
Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

### 35.6.6.6. Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC.NDTB. The number of Tx Queue Buffers is configured by TXBC.TFQS. In case TXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

**Figure 35-11. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue**



Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

### 35.6.6.7. Transmit Cancellation

The CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the CPU has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

**Note:** In case a pending transmission is canceled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

### 35.6.6.8. Tx Event Handling

To support Tx event handling the CAN has implemented a Tx Event FIFO. After the CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in [Tx Event FIFO Element](#) on page 753.

When a Tx Event FIFO full condition is signaled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC.EFWM, interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS.EFGI has to be added to the Tx Event FIFO start address TXEFC.EFSA.

### 35.6.7. FIFO Acknowledge Handling

The Get Indexes of Rx FIFO 0, Rx FIFO 1 and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (refer to [RXF0A](#) on page 724, [RXF1A](#) on page 730 and [TXEFA](#) on page 748). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

**Note:** The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The CAN does not check for erroneous values.

### 35.6.8. Interrupts

The CAN has the following interrupt sources:

- Access to Reserved Address
- Protocol Errors (Data Phase / Arbitration Phase)
- Watchdog Interrupt
- Bus\_Off Status
- Error Warning & Passive
- Error Logging Overflow
- Message RAM Bit Errors (Uncorrected / Corrected)
- Message stored to Dedicated Rx Buffer
- Timeout Occurred
- Message RAM Access Failure
- Timestamp Wraparound
- Tx Event FIFO statuses (Element Lost / Full / Watermark Reached / New Entry)
- Tx FIFO Empty
- Transmission Cancellation Finished
- Timestamp Completed
- High Priority Message
- Rx FIFO 1 Statuses (Message Lost / Full / Watermark Reached / New Message)
- Rx FIFO 0 Statuses (Message Lost / Full / Watermark Reached / New Message)

Each interrupt source has an interrupt flag associated with it. The interrupt flag register (IR) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing '1' or disabled by writing '0' to the corresponding bit in the interrupt enable register (IE). Each interrupt flag can be assigned to one of two interrupt service lines.

An interrupt request is generated when an interrupt flag is set, the corresponding interrupt enable is set, and the corresponding service line enable assigned to the interrupt is set. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled, the service line is disabled, or the CAN is reset. Refer to [IR](#) on page 695 for details on how to clear interrupt flags. All interrupt requests from the peripheral are sent to the NVIC. The user must read the IR 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](#) on page 41

### 35.6.9. Sleep Mode Operation

The CAN can be configured to operate in any sleep mode. To be able to run in standby, register MRCFG.RUNSTDBY must be written to '1'.

To prevent data corruption, it is recommended to allow the CAN to complete all pending transactions before setting the system on standby. This is performed by setting the Clock Stop Request register CCCR.CSR = '1'. Once all transactions are completed, the CAN will automatically set the Clock Stop Acknowledge register CCCR.CSA = '1'. The CAN has reverted back to its initial state and is now safe for the system to go to standby.

When the system wakes up, the CAN cannot be reprogrammed unless the Clock Stop Request register is cleared (CCCR.CSR = '0'). When the Clock Stop Acknowledge register CCCR.CSA returns a '0', the CAN is ready to be programmed.

#### **35.6.10. Synchronization**

Due to the asynchronicity between the main clock domain (CLK\_CAN\_APB) and the peripheral clock domain (GCLK\_CAN) some registers are synchronized when written. When a write-synchronized register is written, the read back value will not be updated until the register has completed synchronization.

The following bits and registers are write-synchronized:

I Initialization bit in CC Control register (CCCR.INIT)

## 35.7. Register Summary

Offset	Name	Bit Pos.									
0x00	CREL	7:0									
0x01		15:8									
0x02		23:16	SUBSTEP[3:0]								
0x03		31:24	REL[3:0]			STEP[3:0]					
0x04	ENDN	7:0	ETV[7:0]								
0x05		15:8	ETV[15:8]								
0x06		23:16	ETV[23:16]								
0x07		31:24	ETV[31:24]								
0x08	MRCFG	7:0	RUNSTDBY					DQOS[1:0]			
0x09		15:8									
0x0A		23:16									
0x0B		31:24									
0x0C	DBTP	7:0	DTSEG2[3:0]			DSJW[3:0]					
0x0D		15:8	DTSEG1[4:0]								
0x0E		23:16	TDC	DBRP[4:0]							
0x0F		31:24									
0x10	TEST	7:0	RX	TX[1:0]	LBCK						
0x11		15:8									
0x12		23:16									
0x13		31:24									
0x14	RWD	7:0	WDC[7:0]								
0x15		15:8	WDV[7:0]								
0x16		23:16									
0x17		31:24									
0x18	CCCR	7:0	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT	
0x19		15:8		TXP	EFBI	PXHD			BRSE	FDOE	
0x1A		23:16									
0x1B		31:24									
0x1C	NBTP	7:0	NTSEG2[6:0]								
0x1D		15:8	NTSEG1[7:0]								
0x1E		23:16	NBRP[7:0]								
0x1F		31:24	NSJW[6:0]			NBRP[8:8]					
0x20	TSCC	7:0	TSS[1:0]								
0x21		15:8									
0x22		23:16	TCP[3:0]								
0x23		31:24									
0x24	TSCV	7:0	TSC[7:0]								
0x25		15:8	TSC[14:8]								
0x26		23:16									
0x27		31:24									
0x28	TOCC	7:0	TOS[1:0]			ETOC					
0x29		15:8									
0x2A		23:16	TOP[7:0]								
0x2B		31:24	TOP[15:8]								

Offset	Name	Bit Pos.									
0x2C	TOCV	7:0	TOC[7:0]								
0x2D		15:8	TOC[15:8]								
0x2E		23:16									
0x2F		31:24									
0x30 ... 0x3F	Reserved										
0x40	ECR	7:0	TEC[7:0]								
0x41		15:8	RP	REC[6:0]							
0x42		23:16	CEL[7:0]								
0x43		31:24									
0x44	PSR	7:0	BO	EW	EP	ACT[1:0]			LEC[2:0]		
0x45		15:8		PXE	RFDF	RBRS	RESI	DLEC[2:0]			
0x46		23:16	TDCV[6:0]								
0x47		31:24									
0x48	TDCR	7:0	TDCF[6:0]								
0x49		15:8	TDCO[6:0]								
0x4A		23:16									
0x4B		31:24									
0x4C ... 0x4F	Reserved										
0x50	IR	7:0	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N	
0x51		15:8	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	
0x52		23:16	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW	
0x53		31:24			ARA	PED	PEA	WDI	BO	EW	
0x54	IE	7:0	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE	
0x55		15:8	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	
0x56		23:16	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE	
0x57		31:24			ARAE	PEDE	PEAE	WDIE	BOE	EWE	
0x58	ILS	7:0	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL	
0x59		15:8	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	
0x5A		23:16	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL	
0x5B		31:24			ARAL	PEDL	PEAL	WDIL	BOL	EWL	
0x5C	ILE	7:0							EINT1	EINT0	
0x5D		15:8									
0x5E		23:16									
0x5F		31:24									
0x60 ... 0x7F	Reserved										
0x80	GFC	7:0			ANFS[1:0]		ANFE[1:0]		RRFS	RRFE	
0x81		15:8									
0x82		23:16									
0x83		31:24									



Offset	Name	Bit Pos.									
0x84	SIDFC	7:0	FLSSA[7:0]								
0x85		15:8	FLSSA[15:8]								
0x86		23:16	LSS[7:0]								
0x87		31:24									
0x88	XIDFC	7:0	FLESA[7:0]								
0x89		15:8	FLESA[15:8]								
0x8A		23:16	LSE[6:0]								
0x8B		31:24									
0x8C ... 0x8F	Reserved										
0x90	XIDAM	7:0	EIDM[7:0]								
0x91		15:8	EIDM[15:8]								
0x92		23:16	EIDM[23:16]								
0x93		31:24	EIDM[28:24]								
0x94	HPMS	7:0	MSI[1:0]			BIDX[5:0]					
0x95		15:8	FLST		FIDX[6:0]						
0x96		23:16									
0x97		31:24									
0x98	NDAT1	7:0	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0	
0x99		15:8	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	
0x9A		23:16	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16	
0x9B		31:24	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	
0x9C	NDAT2	7:0	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0	
0x9D		15:8	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	
0x9E		23:16	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16	
0x9F		31:24	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	
0xA0	RXF0C	7:0	F0SA[7:0]								
0xA1		15:8	F0SA[15:8]								
0xA2		23:16	F0S[6:0]								
0xA3		31:24	F0OM		F0WM[6:0]						
0xA4	RXF0S	7:0	F0FL[6:0]								
0xA5		15:8	F0GI[5:0]								
0xA6		23:16	F0PI[5:0]								
0xA7		31:24								RF0L	F0F
0xA8	RXF0A	7:0	F0AI[5:0]								
0xA9		15:8									
0xAA		23:16									
0xAB		31:24									
0xAC	RXBC	7:0	RBSA[7:0]								
0xAD		15:8	RBSA[15:8]								
0xAE		23:16									
0xAF		31:24									
0xB0	RXF1C	7:0	F1SA[7:0]								
0xB1		15:8	F1SA[15:8]								
0xB2		23:16	F1S[6:0]								
0xB3		31:24	F1OM		F1WM[6:0]						

Offset	Name	Bit Pos.								
0xB4	RXF1S	7:0	F1FL[6:0]							
0xB5		15:8	F1GI[5:0]							
0xB6		23:16	F1PI[5:0]							
0xB7		31:24	DMS[1:0]						RF1L	F1F
0xB8	RXF1A	7:0	F1AI[5:0]							
0xB9		15:8								
0xBA		23:16								
0xBB		31:24								
0xBC	RXESC	7:0	F1DS[2:0]				F0DS[2:0]			
0xBD		15:8					RBDS[2:0]			
0xBE		23:16								
0xBF		31:24								
0xC0	TXBC	7:0	TBSA[7:0]							
0xC1		15:8	TBSA[15:8]							
0xC2		23:16			NDTB[5:0]					
0xC3		31:24		TFQM	TFQS[5:0]					
0xC4	TXFQS	7:0	TFFL[5:0]							
0xC5		15:8			TFGI[4:0]					
0xC6		23:16		TFQF	TFQPI[4:0]					
0xC7		31:24								
0xC8	TXESC	7:0	TBDS[2:0]							
0xC9		15:8								
0xCA		23:16								
0xCB		31:24								
0xCC	TXBRP	7:0	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
0xCD		15:8	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8
0xCE		23:16	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
0xCF		31:24	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24
0xD0	TXBAR	7:0	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
0xD1		15:8	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8
0xD2		23:16	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
0xD3		31:24	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24
0xD4	TXBCR	7:0	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
0xD5		15:8	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8
0xD6		23:16	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
0xD7		31:24	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24
0xD8	TXBTO	7:0	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
0xD9		15:8	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8
0xDA		23:16	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
0xDB		31:24	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24
0xDC	TXBCF	7:0	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
0xDD		15:8	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8
0xDE		23:16	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
0xDF		31:24	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24

Offset	Name	Bit Pos.								
0xE0	TXBTIE	7:0	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
0xE1		15:8	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8
0xE2		23:16	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
0xE3		31:24	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24
0xE4	TXBCIE	7:0	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
0xE5		15:8	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8
0xE6		23:16	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
0xE7		31:24	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24
0xE8 ... 0xEF	Reserved									
0xF0	TXEFC	7:0	EFSA[7:0]							
0xF1		15:8	EFSA[15:8]							
0xF2		23:16			EFS[5:0]					
0xF3		31:24			EFWM[5:0]					
0xF4	TXEFS	7:0			EFFI[4:0]					
0xF5		15:8			EFGI[4:0]					
0xF6		23:16			EFP[4:0]					
0xF7		31:24							TEFL	EFF
0xF8	TXEFA	7:0	EFAI[4:0]							
0xF9		15:8								
0xFA		23:16								
0xFB		31:24								

### 35.8. Register Description

Registers are 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.

### 35.8.1. Core Release

**Name:** CREL  
**Offset:** 0x00  
**Reset:** 0x31000000  
**Property:** Read-only

Bit	31	30	29	28	27	26	25	24
	REL[3:0]				STEP[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	1	1	0	0	0	1
Bit	23	22	21	20	19	18	17	16
	SUBSTEP[3:0]							
Access	R	R	R	R				
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
Access								
Reset								

**Bits 31:28 – REL[3:0]: Core Release**

One digit, BCD-coded.

**Bits 27:24 – STEP[3:0]: Step of Core Release**

One digit, BCD-coded.

**Bits 23:20 – SUBSTEP[3:0]: Sub-step of Core Release**

One digit, BCD-coded.

### 35.8.2. Endian

**Name:** ENDN  
**Offset:** 0x04  
**Reset:** 0x87654321  
**Property:** Read-only

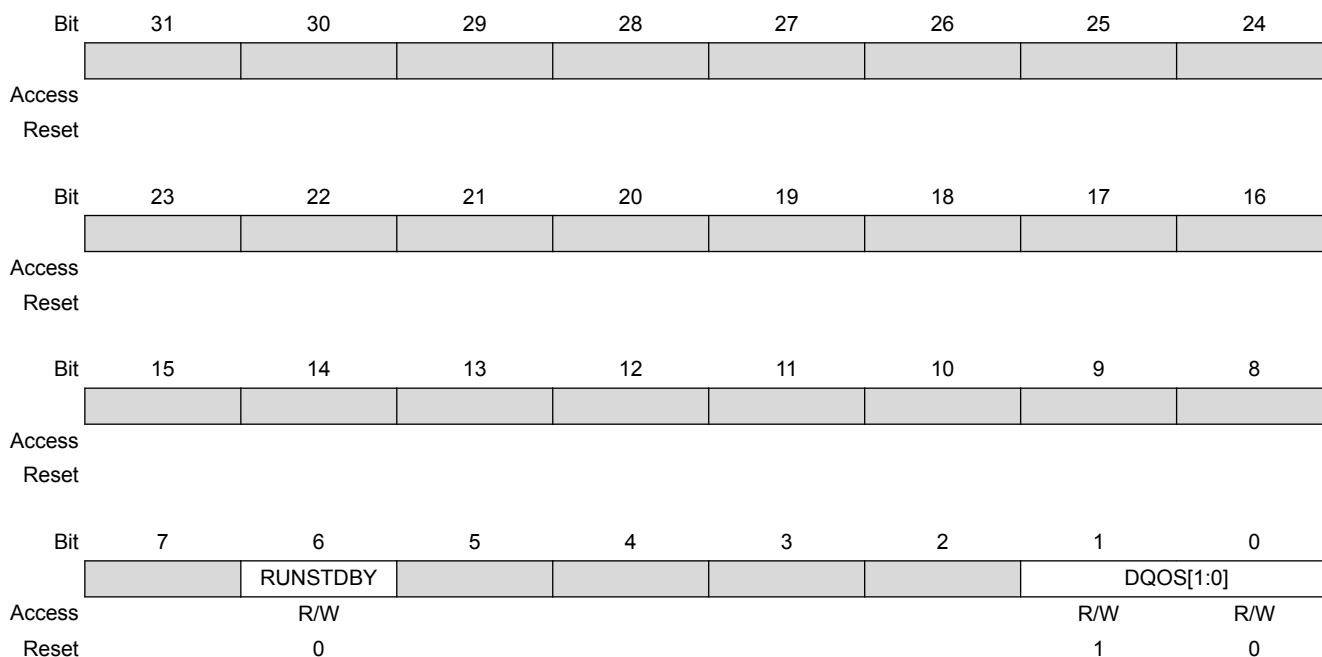
Bit	31	30	29	28	27	26	25	24
ETV[31:24]								
Access	R	R	R	R	R	R	R	R
Reset	1	0	0	0	0	1	1	1
Bit	23	22	21	20	19	18	17	16
ETV[23:16]								
Access	R	R	R	R	R	R	R	R
Reset	0	1	1	0	0	1	0	1
Bit	15	14	13	12	11	10	9	8
ETV[15:8]								
Access	R	R	R	R	R	R	R	R
Reset	0	1	0	0	0	0	1	1
Bit	7	6	5	4	3	2	1	0
ETV[7:0]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

**Bits 31:0 – ETV[31:0]: Endianness Test Value**

The endianness test value is 0x87654321

### 35.8.3. Message RAM Configuration

**Name:** MRCFG  
**Offset:** 0x08  
**Reset:** 0x00000002  
**Property:** -



#### Bit 6 – RUNSTDBY: Run in Standby

This bit controls the behavior of the CAN during standby sleep mode.

Value	Description
0	The CAN GCLK request is always disabled during sleep to conserve power consumption.
1	The CAN GCLK request is disabled during sleep when CCCR.CSA = 1.

#### Bits 1:0 – DQOS[1:0]: Data Quality of Service

This field defines the memory priority access during the Message RAM read/write data operation.

Value	Name	Description
0x0	DISABLE	Background (no sensitive operation)
0x1	LOW	Sensitive bandwidth
0x2	MEDIUM	Sensitive latency
0x3	HIGH	Critical latency

### 35.8.4. Data Bit Timing and Prescaler

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

The CAN bit time may be programmed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 GCLK\_CAN periods.  $t_q = (DBRP + 1) mtq$ .

**Note:**

With a GCLK\_CAN of 8MHz, the reset value 0x00000A33 configures the CAN for a fast bit rate of 500 kBits/s.

The bit rate configured for the CAN FD data phase via DBTP must be higher or equal to the bit rate configured for the arbitration phase via NBTP.

**Name:** DBTP  
**Offset:** 0x0C  
**Reset:** 0x00000A33  
**Property:** Write-restricted

Bit	31	30	29	28	27	26	25	24	
Access									
Reset									
Bit	23	22	21	20	19	18	17	16	
Access	TDC			DBRP[4:0]					
Reset	0			0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	
Access				DTSEG1[4:0]					
Reset				0	1	0	1	0	
Bit	7	6	5	4	3	2	1	0	
Access	DTSEG2[3:0]			DSJW[3:0]					
Reset	0	0	1	1	0	0	1	1	

#### Bit 23 – TDC: Transceiver Delay Compensation

Value	Description
0	Transceiver Delay Compensation disabled.
1	Transceiver Delay Compensation enabled.

#### Bits 20:16 – DBRP[4:0]: Data Baud Rate Prescaler

Value	Description
0x00 - 0x1F	The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

**Bits 12:8 – DTSEG1[4:0]: Fast time segment before sample point**

Value	Description
0x00 - 0x1F	Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. DTSEG1 is the sum of Prop_Seg and Phase_Seg1.

**Bits 7:4 – DTSEG2[3:0]: Data time segment after sample point**

Value	Description
0x0 - 0xF	Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. DTSEG2 is Phase_Seg2.

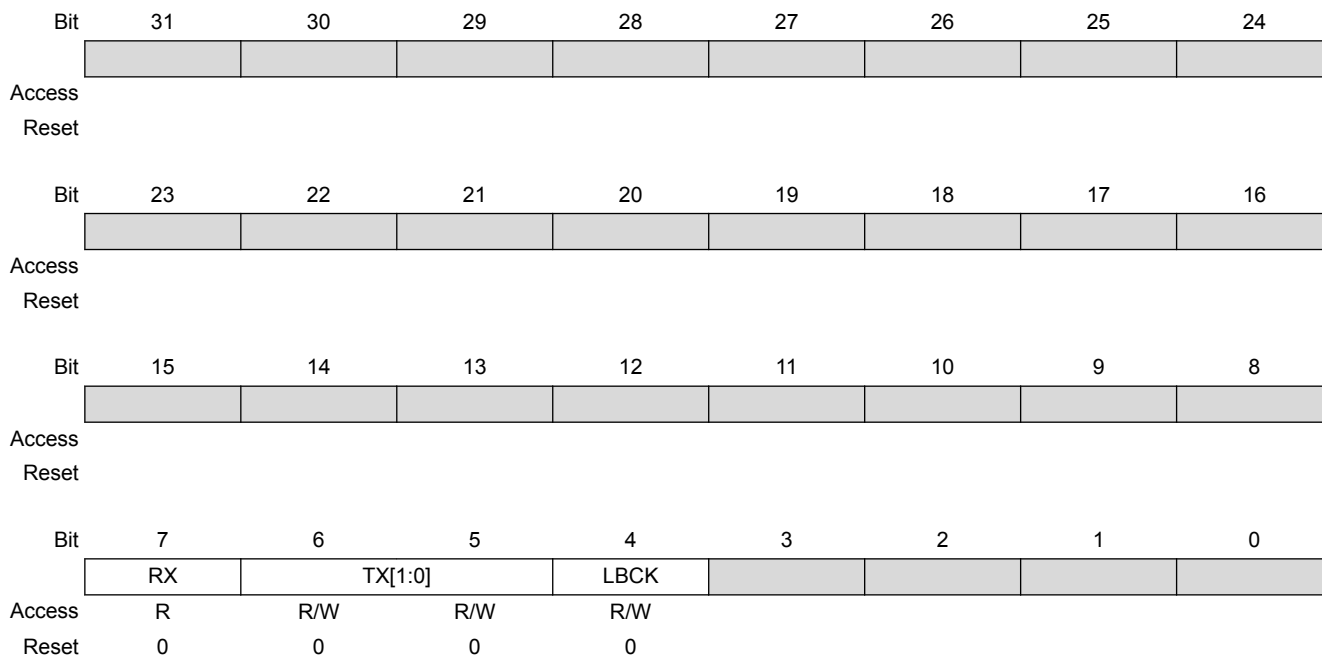
**Bits 3:0 – DSJW[3:0]: Data (Re)Synchronization Jump Width**

Value	Description
0x0 - 0xF	Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.



### 35.8.5. Test

**Name:** TEST  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** Read-only, Write-restricted



#### Bit 7 – RX: Receive Pin

Monitors the actual value of pin CAN\_RX

Value	Description
0	The CAN bus is dominant (CAN_RX = 0).
1	The CAN bus is recessive (CAN_RX = 1).

#### Bits 6:5 – TX[1:0]: Control of Transmit Pin

This field defines the control of the transmit pin.

Value	Name	Description
0x0	CORE	Reset value, CAN_TX controlled by CAN core, updated at the end of CAN bit time.
0x1	SAMPLE	Sample Point can be monitored at pin CAN_TX.
0x2	DOMINANT	Dominant ('0') level at pin CAN_TX.
0x3	RECESSIVE	Recessive ('1') level at pin CAN_TX.

#### Bit 4 – LBCK: Loop Back Mode

Value	Description
0	Loop Back Mode is disabled.
1	Loop Back Mode is enabled.

### 35.8.6. RAM Watchdog

This register is write-restricted and only writable if bit fields `CCCR.CCE = 1` and `CCCR.INIT = 1`.

The RAM Watchdog monitors the `READY` output of the Message RAM. A Message RAM access via the CAN's AHB Master Interface starts the Message RAM Watchdog Counter with the value configured by `RWD.WDC`. The counter is reloaded with `RWD.WDC` when the Message RAM signals successful completion by activating its `READY` output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt `IR.WDI` is set.

**Name:** RWD

**Offset:** 0x14

**Reset:** 0x00000000

**Property:** Read-only, Write-restricted

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
	WDV[7:0]							
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
	WDC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 15:8 – WDV[7:0]: Watchdog Value

Actual Message RAM Watchdog Counter Value.

#### Bits 7:0 – WDC[7:0]: Watchdog Configuration

Start value of the Message RAM Watchdog Counter. With the reset value of 0x00 the counter is disabled.

### 35.8.7. CC Control

**Name:** CCCR  
**Offset:** 0x18  
**Reset:** 0x00000001  
**Property:** Read-only, Write-restricted

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		TXP	EFBI	PXHD			BRSE	FDOE
Reset		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
Access	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
Reset	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

#### Bit 14 – TXP: Transmit Pause

This bit field is write-restricted and only writable if bit fields CCE = 1 and INIT = 1.

Value	Description
0	Transmit pause disabled.
1	Transmit pause enabled. The CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame.

#### Bit 13 – EFBI: Edge Filtering during Bus Integration

Value	Description
0	Edge filtering is disabled.
1	Two consecutive dominant tq required to detect an edge for hard synchronization.

#### Bit 12 – PXHD: Protocol Exception Handling Disable

**Note:** When protocol exception handling is disabled, the CAN will transmit an error frame when it detects a protocol exception condition.

Value	Description
0	Protocol exception handling enabled.
1	Protocol exception handling disabled.

#### Bit 9 – BRSE: Bit Rate Switch Enable

**Note:** When CAN FD operation is disabled FDOE = 0, BRSE is not evaluated.

Value	Description
0	Bit rate switching for transmissions disabled.
1	Bit rate switching for transmissions enabled.

#### Bit 8 – FDOE: FD Operation Enable

Value	Description
0	FD operation disabled.
1	FD operation enabled.

#### Bit 7 – TEST: Test Mode Enable

This bit field is write-restricted.

Writing a 0 to this field is always allowed.

Writing a 1 to this field is only allowed if bit fields CCE = 1 and INIT = 1.

Value	Description
0	Normal operation. Register TEST holds reset values.
1	Test Mode, write access to register TEST enabled.

#### Bit 6 – DAR: Disable Automatic Retransmission

This bit field is write-restricted and only writable if bit fields CCE = 1 and INIT = 1.

Value	Description
0	Automatic retransmission of messages not transmitted successfully enabled.
1	Automatic retransmission disabled.

#### Bit 5 – MON: Bus Monitoring Mode

This bit field is write-restricted.

Writing a 0 to this field is always allowed.

Writing a 1 to this field is only allowed if bit fields CCE = 1 and INIT = 1.

Value	Description
0	Bus Monitoring Mode is disabled.
1	Bus Monitoring Mode is enabled.

#### Bit 4 – CSR: Clock Stop Request

Value	Description
0	No clock stop is requested.
1	Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.

### Bit 3 – CSA: Clock Stop Acknowledge

Value	Description
0	No clock stop acknowledged.
1	CAN may be set in power down by stopping CLK_CAN_APB and GCLK_CAN.

### Bit 2 – ASM: Restricted Operation Mode

This bit field is write-restricted.

Writing a 0 to this field is always allowed.

Writing a 1 to this field is only allowed if bit fields CCE = 1 and INIT = 1.

Value	Description
0	Normal CAN operation.
1	Restricted Operation Mode active.

### Bit 1 – CCE: Configuration Change Enable

This bit field is write-restricted and only writable if bit field INIT = 1.

Value	Description
0	The CPU has no write access to the protected configuration registers.
1	The CPU has write access to the protected configuration registers (while CCCR.INIT = 1).

### Bit 0 – INIT: Initialization

Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. The programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

Value	Description
0	Normal Operation.
1	Initialization is started.

### 35.8.8. Nominal Bit Timing and Prescaler

This register is write-restricted and only writable if bit fields `CCCR.CCE = 1` and `CCCR.INIT = 1`.

The CAN bit time may be programmed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 512 `GCLK_CAN` periods.  $t_q = (NBRP + 1) mtq$ .

**Note:** With a CAN clock (`GCLK_CAN`) of 8MHz, the reset value `0x06000A03` configures the CAN for a bit rate of 500 kBits/s.

**Name:** NBTP

**Offset:** 0x1C

**Reset:** 0x00000A33

**Property:** Write-restricted

Bit	31	30	29	28	27	26	25	24
	NSJW[6:0]							NBRP[8:8]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0
Bit	23	22	21	20	19	18	17	16
	NBRP[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
	NTSEG1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	1	0	1	0
Bit	7	6	5	4	3	2	1	0
		NTSEG2[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	1	1

#### Bits 31:25 – NSJW[6:0]: Nominal (Re)Synchronization Jump Width

Value	Description
0x00 - 0x7F	Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

#### Bits 24:16 – NBRP[8:0]: Nominal Baud Rate Prescaler

Value	Description
0x000 - 0x1FF	The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

#### Bits 15:8 – NTSEG1[7:0]: Nominal Time segment before sample point

Value	Description
0x00 - 0x7F	Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. NTSEG1 is the sum of Prop_Seg and Phase_Seg1.

**Bits 6:0 – NTSEG2[6:0]: Time segment after sample point**

Value	Description
0x00 - 0x7F	Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. NTSEG2 is Phase_Seg2.



### 35.8.9. Timestamp Counter Configuration

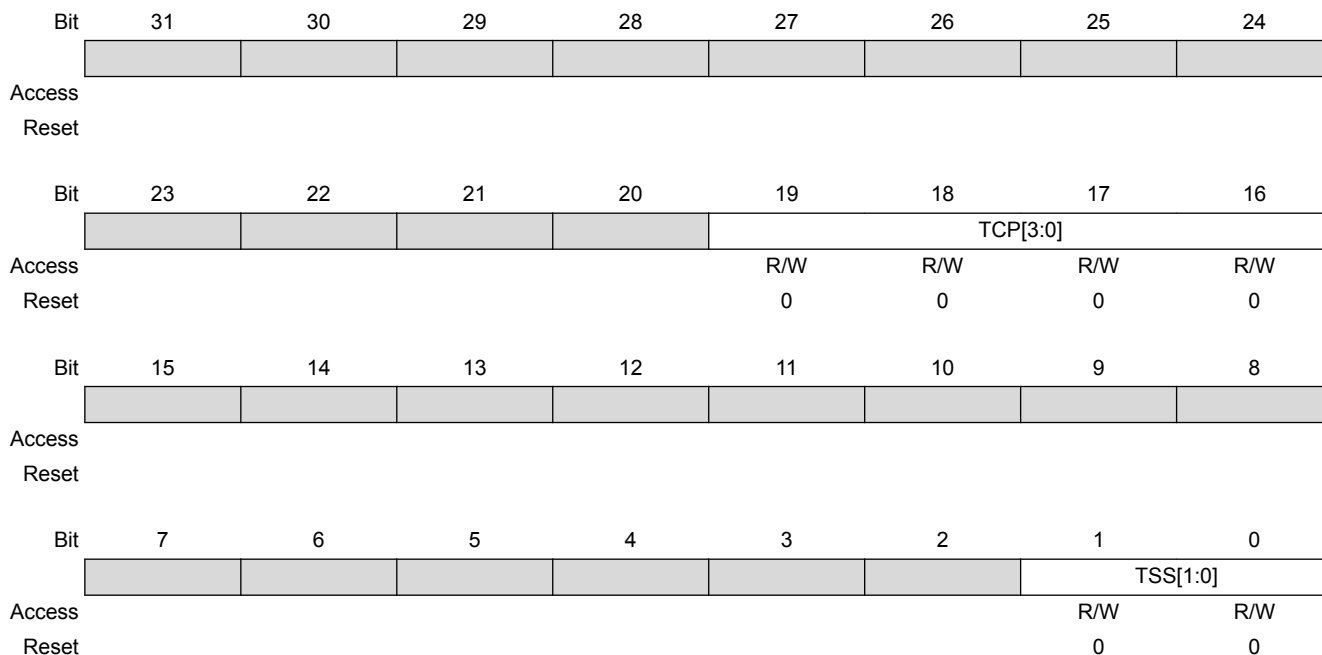
This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

**Name:** TSCC

**Offset:** 0x20

**Reset:** 0x00000000

**Property:** Write-restricted



#### Bits 19:16 – TCP[3:0]: Timestamp Counter Prescaler

**Note:** With CAN FD an external counter is required for timestamp generation (TSS = 0x2).

Value	Description
0x0 - 0xF	Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1...16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

#### Bits 1:0 – TSS[1:0]: Timestamp Select

This field defines the timestamp counter selection.

Value	Name	Description
0x0 or 0x3	ZERO	Timestamp counter value always 0x0000.
0x1	INC	Timestamp counter value incremented by TCP.
0x2	EXT	External timestamp counter value used.

### 35.8.10. Timestamp Counter Value

**Note:**

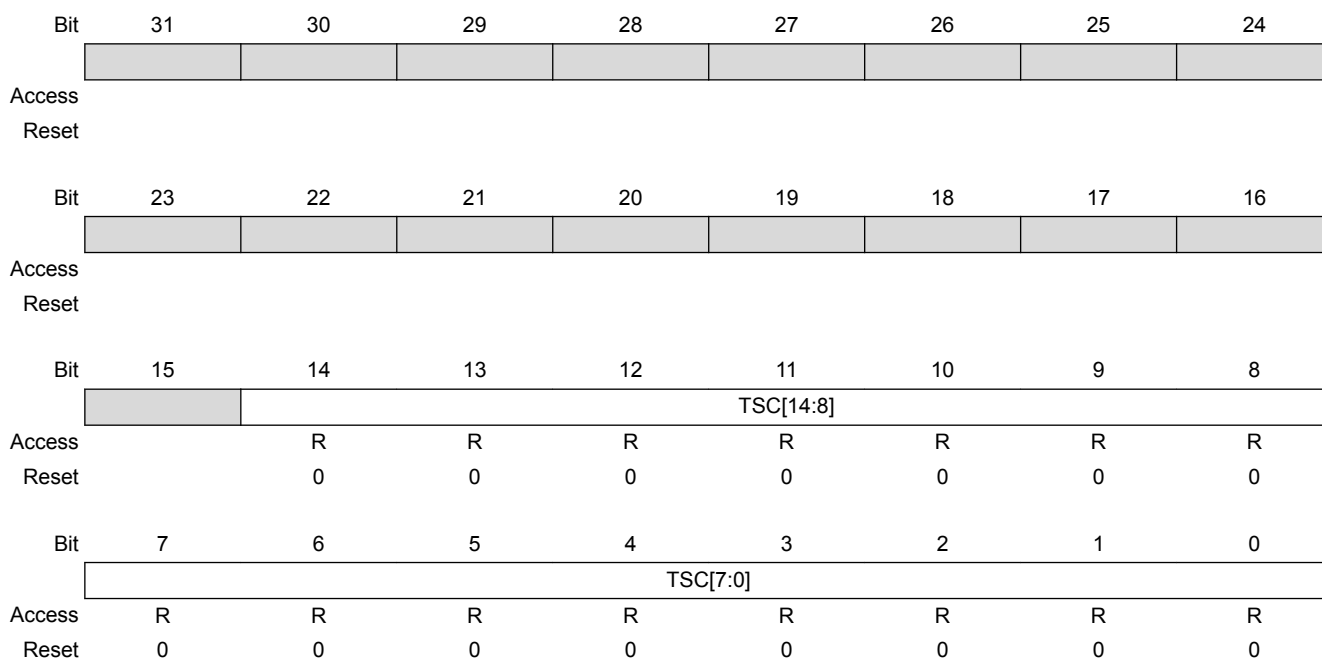
1. A write access to TSCV while in internal mode clears the Timestamp Counter value. A write access to TSCV while in external mode has no impact.
2. A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by the write access to TSCV.

**Name:** TSCV

**Offset:** 0x24

**Reset:** 0x00000000

**Property:** Read-only



**Bits 14:0 – TSC[14:0]: Timestamp Counter**

The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC.TSS = 0x1, the Timestamp Counter is incremented in multiples of CAN bit times [1...16] depending on the configuration of TSCC.TCP. A wrap around sets interrupt flag IR.TSW. When TSCC.TSS = 0x2, TSC reflects the external Timestamp Counter value.

### 35.8.11. Timeout Counter Configuration

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

**Name:** TOCC

**Offset:** 0x28

**Reset:** 0xFFFF0000

**Property:** Write-restricted

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

#### Bits 31:16 – TOP[15:0]: Timeout Period

Start value of the Timeout Counter (down-counter). Configures the Timeout Period.

#### Bits 2:1 – TOS[1:0]: Timeout Select

When operating in Continuous mode, a write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC.TOP. Down-counting is started when the first FIFO element is stored.

Value	Name	Description
0x0	CONT	Continuous operation.
0x1	TXEF	Timeout controlled by TX Event FIFO.
0x2	RXF0	Timeout controlled by Rx FIFO 0.
0x3	RXF1	Timeout controlled by Rx FIFO 1.

#### Bit 0 – ETOC: Enable Timeout Counter

Value	Description
0	Timeout Counter disabled.
1	Timeout Counter enabled.

### 35.8.12. Timeout Counter Value

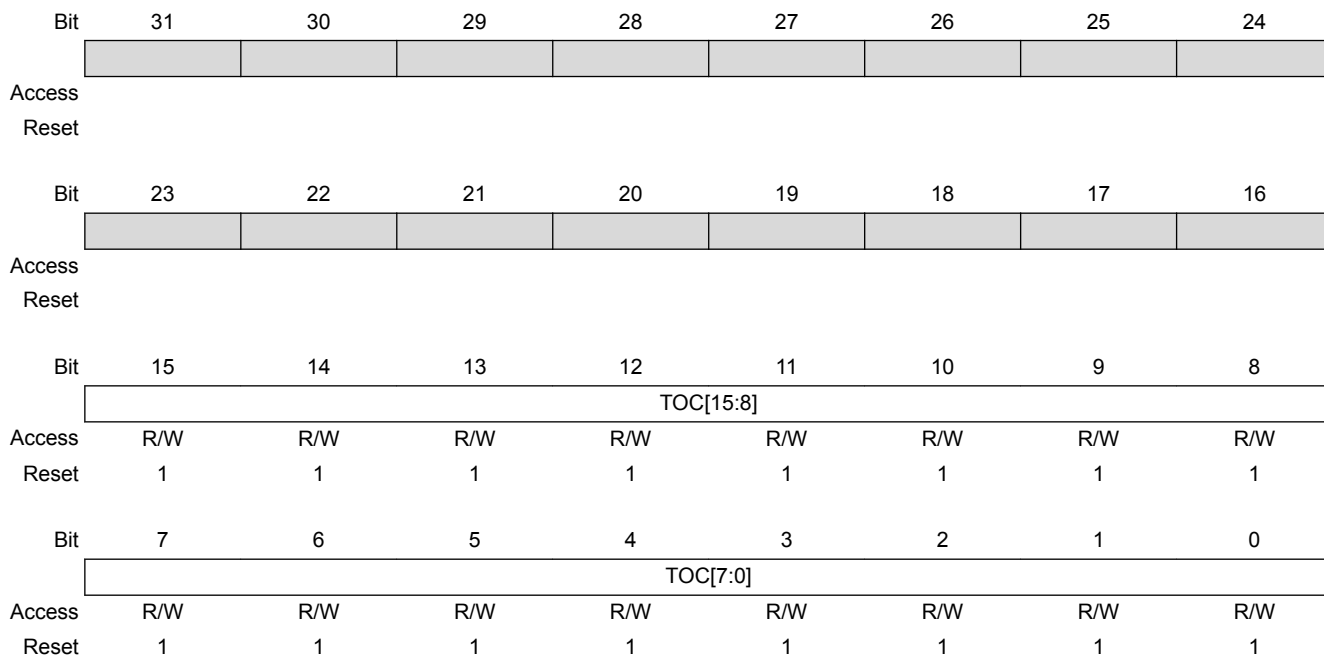
**Note:** A write access to TOCV reloads the Timeout Counter with the value of TOCV.TOP.

**Name:** TOCV

**Offset:** 0x2C

**Reset:** 0x0000FFFF

**Property:** Read-only



#### Bits 15:0 – TOC[15:0]: Timeout Counter

The Timeout Counter is decremented in multiples of CAN bit times [1...16] depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

### 35.8.13. Error Counter

**Note:** When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

**Name:** ECR  
**Offset:** 0x40  
**Reset:** 0x00000000  
**Property:** Read-only

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
Access								
Reset								

#### Bits 23:16 – CEL[7:0]: CAN Error Logging

The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR.ELO.

#### Bit 15 – RP: Receive Error Passive

#### Bits 14:8 – REC[6:0]: Receive Error Counter

Actual state of the Receive Error Counter, values between 0 and 127.

#### Bits 7:0 – TEC[7:0]: Transmit Error Counter

Actual state of the Transmit Error Counter, values between 0 and 255.

### 35.8.14. Protocol Status

**Note:**

1. When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.
2. The Bus\_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO 11898-1) cannot be shortened by setting or resetting CCCR.INIT. If the device goes Bus\_Off, it will set CCCR.INIT of its own accord, stopping all bus activities. Once CCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 \* 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus\_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0 Error code is written to PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus\_Off recovery sequence. ECR.REC is used to count these sequences.

**Name:** PSR  
**Offset:** 0x44  
**Reset:** 0x00000707  
**Property:** Read-only

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

**Bits 22:16 – TDCV[6:0]: Transmitter Delay Compensation Value**

Value	Description
0x00 - 0x7F	Position of the secondary sample point, defined by the sum of the measured delay from CAN_TX to CAN_RX and TDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

**Bit 14 – PXE: Protocol Exception Event**

This field is cleared on read access.

Value	Description
0	No protocol exception event occurred since last read access.
1	Protocol exception event occurred.

**Bit 13 – RFDF: Received a CAN FD Message**

This field is cleared on read access.

Value	Description
0	Since this bit was reset by the CPU, no CAN FD message has been received.
1	Message in CAN FD format with FDF flag set has been received. This bit is set independent of acceptance filtering.

**Bit 12 – RBRS: BRS flag of last received CAN FD Message**

This field is cleared on read access.

Value	Description
0	Last received CAN FD message did not have its BRS flag set.
1	Last received CAN FD message had its BRS flag set. This bit is set together with RFDF, independent of acceptance filtering.

**Bit 11 – RESI: ESI flag of last received CAN FD Message**

This field is cleared on read access.

Value	Description
0	Last received CAN FD message did not have its ESI flag set.
1	Last received CAN FD message had its ESI flag set.

**Bits 10:8 – DLEC[2:0]: Data Last Error Code**

Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.

**Bit 7 – BO: Bus\_Off Status**

Value	Description
0	The CAN is not Bus_Off.
1	The CAN is in Bus_Off state.

**Bit 6 – EW: Error Warning Status**

Value	Description
0	Both error counters are below the Error_Warning limit of 96.
1	At least one of the error counter has reached the Error_Warning limit of 96.



### Bit 5 – EP: Error Passive

Value	Description
0	The CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.
1	The CAN is in the Error_Passive state.

### Bits 4:3 – ACT[1:0]: Activity

Monitors the module's CAN communication state.

Value	Name	Description
0x0	SYNC	Node is synchronizing on CAN communication.
0x1	IDLE	Node is neither receiver nor transmitter.
0x2	RX	Node is operating as receiver.
0x3	TX	Node is operating as transmitter.

### Bits 2:0 – LEC[2:0]: Last Error Code

The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.

This field is set on read access.

Value	Name	Description
0x0	NONE	No Error: No error occurred since LEC has been reset by successful reception or transmission.
0x1	STUFF	Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
0x2	FORM	Form Error: A fixed format part of a received frame has the wrong format.
0x3	ACK	Ack Error: The message transmitted by the CAN was not acknowledged by another node.
0x4	BIT1	Bit1 Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus was dominant.
0x5	BIT0	Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits have been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
0x6	CRC	CRC Error: The CRC checksum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
0x7	NC	No Change: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

### 35.8.15. Transmitter Delay Compensation

**Name:** TDCR  
**Offset:** 0x48  
**Reset:** 0x00000000  
**Property:** Write-restricted

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		TDCO[6:0]						
Reset		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
Access		TDCF[6:0]						
Reset		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

#### Bits 14:8 – TDCO[6:0]: Transmitter Delay Compensation Offset

Value	Description
0x00 - 0x7F	Offset value defining the distance between the measured delay from CAN_TX to CAN_RX and the secondary sample point. Valid values are 0 to 127 mtq.

#### Bits 6:0 – TDCF[6:0]: Transmitter Delay Compensation Filter Window Length

Value	Description
0x00 - 0x7F	Defines the minimum value for the SSP position, dominant edges on CAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq.

### 35.8.16. Interrupt

The flags are set when one of the listed conditions is detected (edge-sensitive). A flag is cleared by writing a 1 to the corresponding bit field. Writing a 0 has no effect. A hard reset will clear the register.

**Name:** IR  
**Offset:** 0x50  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
			ARA	PED	PEA	WDI	BO	EW
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
	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
Access	R/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
	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM
Access	R/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
	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
Access	R/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 29 – ARA: Access to Reserved Address

Value	Description
0	No access to reserved address occurred.
1	Access to reserved address occurred.

#### Bit 28 – PED: Protocol Error in Data Phase

Value	Description
0	No protocol error in data phase.
1	Protocol error in data phase detected (PSR.DLEC != 0,7).

#### Bit 27 – PEA: Protocol Error in Arbitration Phase

Value	Description
0	No protocol error in arbitration phase.
1	Protocol error in arbitration phase detected (PSR.LEC != 0,7).

#### Bit 26 – WDI: Watchdog Interrupt

Value	Description
0	No Message RAM Watchdog event occurred.
1	Message RAM Watchdog event due to missing READY.

#### Bit 25 – BO: Bus\_Off Status

Value	Description
0	Bus_Off status unchanged.
1	Bus_Off status changed.

#### Bit 24 – EW: Error Warning Status

Value	Description
0	Error_Warning status unchanged.
1	Error_Warning status changed.

#### Bit 23 – EP: Error Passive

Value	Description
0	Error_Passive status unchanged.
1	Error_Passive status changed.

#### Bit 22 – ELO: Error Logging Overflow

Value	Description
0	CAN Error Logging Counter did not overflow.
1	Overflow of CAN Error Logging Counter occurred.

#### Bit 21 – BEU: Bit Error Uncorrected

Message RAM bit error detected, uncorrected. Generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit sets CCCR.INIT to 1. This is done to avoid transmission of corrupted data.

Value	Description
0	Not bit error detected when reading from Message RAM.
1	Bit error detected, uncorrected (e.g. parity logic).

#### Bit 20 – BEC: Bit Error Corrected

Message RAM bit error detected and corrected. Generated by an optional external parity / ECC logic attached to the Message RAM.

Value	Description
0	Not bit error detected when reading from Message RAM.
1	Bit error detected and corrected (e.g. ECC).

#### Bit 19 – DRX: Message stored to Dedicated Rx Buffer

The flag is set whenever a received message has been stored into a dedicated Rx Buffer.

Value	Description
0	No Rx Buffer updated.
1	At least one received message stored into a Rx Buffer.

#### Bit 18 – TOO: Timeout Occurred

Value	Description
0	No timeout.
1	Timeout reached.

#### Bit 17 – MRAF: Message RAM Access Failure

The flag is set, when the Rx Handler

- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message.
- was not able to write a message to the Message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location.

The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the CAN is switched into Restricted Operation Mode. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM.

Value	Description
0	No Message RAM access failure occurred.
1	Message RAM access failure occurred.

#### Bit 16 – TSW: Timestamp Wraparound

Value	Description
0	No timestamp counter wrap-around.
1	Timestamp counter wrapped around.

#### Bit 15 – TEFL: Tx Event FIFO Element Lost

Value	Description
0	No Tx Event FIFO element lost.
1	Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.

#### Bit 14 – TEFF: Tx Event FIFO Full

Value	Description
0	Tx Event FIFO not full.
1	Tx Event FIFO full.

**Bit 13 – TEFW: Tx Event FIFO Watermark Reached**

Value	Description
0	Tx Event FIFO fill level below watermark.
1	Tx Event FIFO fill level reached watermark.

**Bit 12 – TEFN: Tx Event FIFO New Entry**

Value	Description
0	Tx Event FIFO unchanged.
1	Tx Handler wrote Tx Event FIFO element.

**Bit 11 – TFE: Tx FIFO Empty**

Value	Description
0	Tx FIFO non-empty.
1	Tx FIFO empty.

**Bit 10 – TCF: Transmission Cancellation Finished**

Value	Description
0	No transmission cancellation finished.
1	Transmission cancellation finished.

**Bit 9 – TC: Timestamp Completed**

Value	Description
0	No transmission completed.
1	Transmission completed.

**Bit 8 – HPM: High Priority Message**

Value	Description
0	No high priority message received.
1	High priority message received.

**Bit 7 – RF1L: Rx FIFO 1 Message Lost**

Value	Description
0	No Rx FIFO 1 message lost.
1	Rx FIFO 1 message lost. also set after write attempt to Rx FIFO 1 of size zero.

**Bit 6 – RF1F: Rx FIFO 1 Full**

Value	Description
0	Rx FIFO 1 not full.
1	Rx FIFO 1 full.

**Bit 5 – RF1W: Rx FIFO 1 Watermark Reached**

Value	Description
0	Rx FIFO 1 fill level below watermark.
1	Rx FIFO 1 fill level reached watermark.

**Bit 4 – RF1N: Rx FIFO 1 New Message**

Value	Description
0	No new message written to Rx FIFO 1.
1	New message written to Rx FIFO 1.

**Bit 3 – RF0L: Rx FIFO 0 Message Lost**

Value	Description
0	No Rx FIFO 0 message lost.
1	Rx FIFO 0 message lost. also set after write attempt to Rx FIFO 0 of size zero.

**Bit 2 – RF0F: Rx FIFO 0 Full**

Value	Description
0	Rx FIFO 0 not full.
1	Rx FIFO 0 full.

**Bit 1 – RF0W: Rx FIFO 0 Watermark Reached**

Value	Description
0	Rx FIFO 0 fill level below watermark.
1	Rx FIFO 0 fill level reached watermark.

**Bit 0 – RF0N: Rx FIFO 0 New Message**

Value	Description
0	No new message written to Rx FIFO 0.
1	New message written to Rx FIFO 0.

### 35.8.17. Interrupt Enable

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signalled on an interrupt line.

**Name:** IE  
**Offset:** 0x54  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
			ARAE	PEDE	PEAE	WDIE	BOE	EWE
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
	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
Access	R/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
	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME
Access	R/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
	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
Access	R/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 29 – ARAE: Access to Reserved Address Interrupt Enable

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

#### Bit 28 – PEDE: Protocol Error in Data Phase Interrupt Enable

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

#### Bit 27 – PEAE: Protocol Error in Arbitration Phase Interrupt Enable

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

#### Bit 26 – WDIE: Watchdog Interrupt Enable



Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 25 – BOE: Bus\_Off Status Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 24 – EWE: Error Warning Status Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 23 – EPE: Error Passive Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 22 – ELOE: Error Logging Overflow Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 21 – BEUE: Bit Error Uncorrected Interrupt Enable.**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 20 – BECE: Bit Error Corrected Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 19 – DRXE: Message stored to Dedicated Rx Buffer Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 18 – TOOE: Timeout Occurred Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 17 – MRAFE: Message RAM Access Failure Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 16 – TSWE: Timestamp Wraparound Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 15 – TEFLE: Tx Event FIFO Event Lost Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 14 – TEFWE: Tx Event FIFO Full Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 13 – TEFWE: Tx Event FIFO Watermark Reached Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 12 – TEFNE: Tx Event FIFO New Entry Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 11 – TFEE: Tx FIFO Empty Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 10 – TCFE: Transmission Cancellation Finished Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 9 – TCE: Transmission Completed Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 8 – HPME: High Priority Message Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 7 – RF1LE: Rx FIFO 1 Message Lost Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 6 – RF1FE: Rx FIFO 1 Full Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 5 – RF1WE: Rx FIFO 1 Watermark Reached Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 4 – RF1NE: Rx FIFO 1 New Message Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 3 – RF0LE: Rx FIFO 0 Message Lost Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 2 – RF0FE: Rx FIFO 0 Full Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 1 – RF0WE: Rx FIFO 0 Watermark Reached Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

**Bit 0 – RF0NE: Rx FIFO 0 New Message Interrupt Enable**

Value	Description
0	Interrupt disabled.
1	Interrupt enabled.

### 35.8.18. Interrupt Line Select

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from IR to one of the two module interrupt lines.

**Name:** ILS  
**Offset:** 0x58  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
			ARAL	PEDL	PEAL	WDIL	BOL	EWL
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
	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
Access	R/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
	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML
Access	R/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
	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
Access	R/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 29 – ARAL: Access to Reserved Address Interrupt Line

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

#### Bit 28 – PEDL: Protocol Error in Data Phase Interrupt Line

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

#### Bit 27 – PEAL: Protocol Error in Arbitration Phase Interrupt Line

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

#### Bit 26 – WDIL: Watchdog Interrupt Line

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 25 – BOL: Bus\_Off Status Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 24 – EWL: Error Warning Status Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 23 – EPL: Error Passive Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 22 – ELOL: Error Logging Overflow Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 21 – BEUL: Bit Error Uncorrected Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 20 – BECL: Bit Error Corrected Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 19 – DRXL: Message stored to Dedicated Rx Buffer Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 18 – TOOL: Timeout Occurred Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 17 – MRAFL: Message RAM Access Failure Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 16 – TSWL: Timestamp Wraparound Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 15 – TEFL: Tx Event FIFO Event Lost Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 14 – TEFFL: Tx Event FIFO Full Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 13 – TEFWL: Tx Event FIFO Watermark Reached Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 12 – TEFNL: Tx Event FIFO New Entry Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 11 – TFEL: Tx FIFO Empty Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 10 – TCFL: Transmission Cancellation Finished Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 9 – TCL: Transmission Completed Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 8 – HPML: High Priority Message Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 7 – RF1LL: Rx FIFO 1 Message Lost Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 6 – RF1FL: Rx FIFO 1 Full Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 5 – RF1WL: Rx FIFO 1 Watermark Reached Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 4 – RF1NL: Rx FIFO 1 New Message Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.



**Bit 3 – RF0LL: Rx FIFO 0 Message Lost Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 2 – RF0FL: Rx FIFO 0 Full Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 1 – RF0WL: Rx FIFO 0 Watermark Reached Interrupt Line**

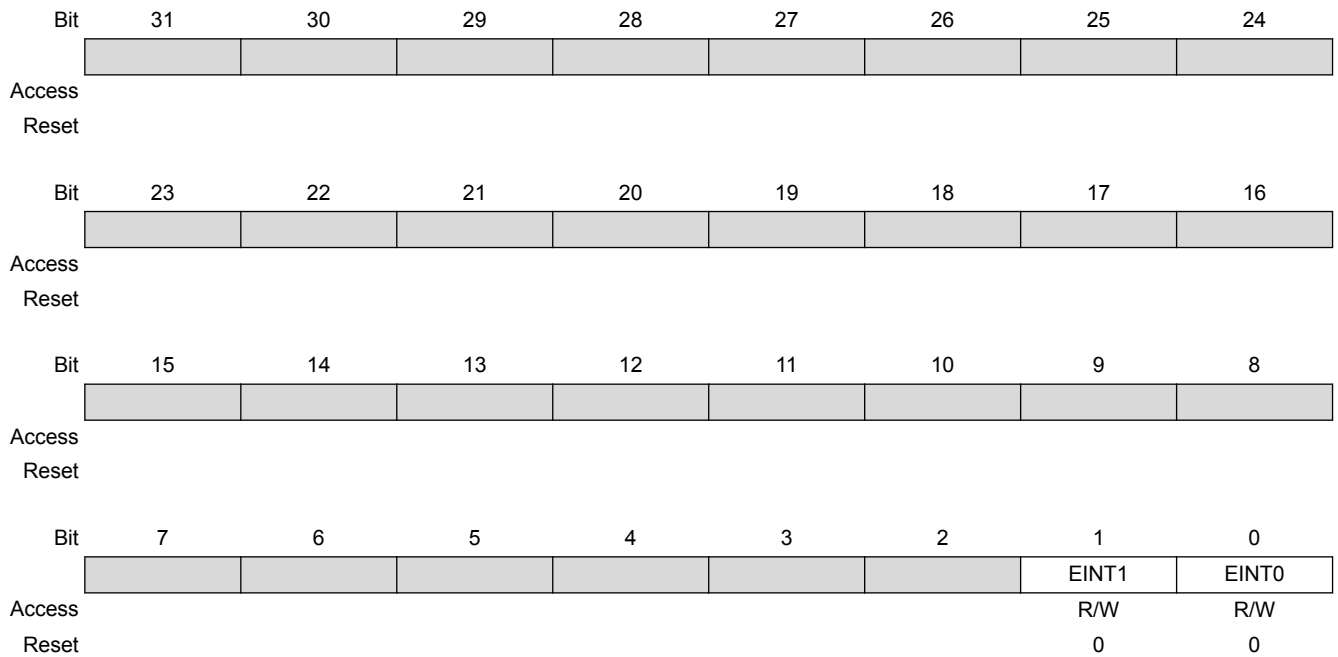
Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

**Bit 0 – RF0NL: Rx FIFO 0 New Message Interrupt Line**

Value	Description
0	Interrupt assigned to CAN interrupt line 0.
1	Interrupt assigned to CAN interrupt line 1.

### 35.8.19. Interrupt Line Enable

**Name:** ILE  
**Offset:** 0x5C  
**Reset:** 0x00000000  
**Property:** -



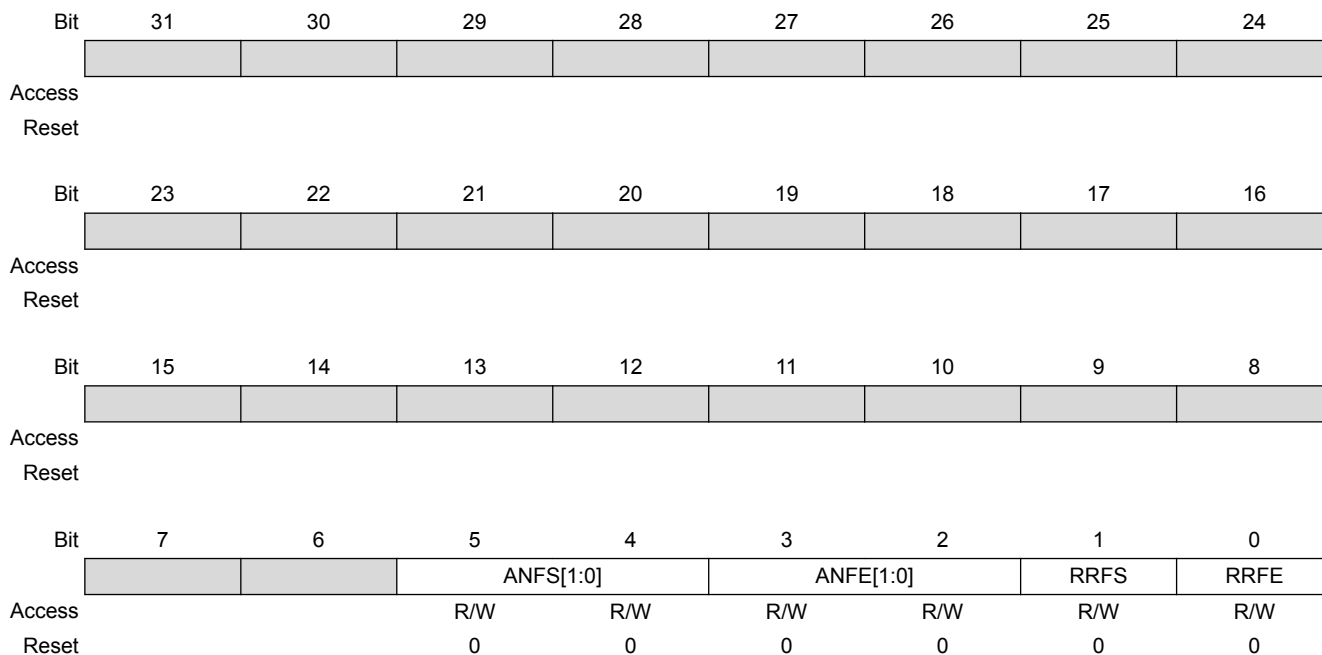
#### Bits 1:0 – EINTn: Enable Interrupt Line n [n = 1,0]

Value	Description
0	CAN interrupt line n disabled.
1	CAN interrupt line n enabled.

### 35.8.20. Global Filter Configuration

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

**Name:** GFC  
**Offset:** 0x80  
**Reset:** 0x00000000  
**Property:** Write-restricted



#### Bits 5:4 – ANFS[1:0]: Accept Non-matching Frames Standard

Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.

Value	Name	Description
0x0	RXF0	Accept in Rx FIFO 0.
0x1	RXF1	Accept in Rx FIFO 1.
0x2 or 0x3	REJECT	Reject

#### Bits 3:2 – ANFE[1:0]: Accept Non-matching Frames Extended

Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.

Value	Name	Description
0x0	RXF0	Accept in Rx FIFO 0.
0x1	RXF1	Accept in Rx FIFO 1.
0x2 or 0x3	REJECT	Reject

#### Bit 1 – RRFS: Reject Remote Frames Standard

Value	Description
0	Filter remote frames with 11-bit standard IDs.
1	Reject all remote frames with 11-bit standard IDs.

**Bit 0 – RRFE: Reject Remote Frames Extended**

Value	Description
0	Filter remote frames with 29-bit extended IDs.
1	Reject all remote frames with 29-bit extended IDS.

### 35.8.21. Standard ID Filter Configuration

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

**Name:** SIDFC  
**Offset:** 0x84  
**Reset:** 0x00000000  
**Property:** Write-restricted

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	LSS[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
	FLSSA[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
	FLSSA[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 – LSS[7:0]: List Size Standard

Value	Description
0	No standard Message ID filter.
1 - 128	Number of standard Message ID filter elements.
> 128	Values greater than 128 are interpreted as 128.

#### Bits 15:0 – FLSSA[15:0]: Filter List Standard Start Address

Start address of standard Message ID filter list. When the CAN module addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored. Bits 1 to 0 will always be read back as “00”.

### 35.8.22. Extended ID Filter Configuration

**Name:** XIDFC  
**Offset:** 0x88  
**Reset:** 0x00000000  
**Property:** Write-restricted

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

#### Bits 22:16 – LSE[6:0]: List Size Extended

Value	Description
0	No extended Message ID filter.
1 - 64	Number of Extended Message ID filter elements.
> 64	Values greater than 64 are interpreted as 64.

#### Bits 15:0 – FLESA[15:0]: Filter List Extended Start Address

Start address of extended Message ID filter list. When the CAN module addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored. Bits 1 to 0 will always be read back as “00”.

### 35.8.23. Extended ID AND Mask

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

**Name:** XIDAM

**Offset:** 0x90

**Reset:** 0x1FFFFFFF

**Property:** Write-restricted

Bit	31	30	29	28	27	26	25	24
				EIDM[28:24]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				1	1	1	1	1
Bit	23	22	21	20	19	18	17	16
	EIDM[23:16]							
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
	EIDM[15:8]							
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
	EIDM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

#### Bits 28:0 – EIDM[28:0]: Extended ID Mask

For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

### 35.8.24. High Priority Message Status

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

**Name:** HPMS  
**Offset:** 0x94  
**Reset:** 0x00000000  
**Property:** Read-only

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
	FLST		FIDX[6:0]					
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
	MSI[1:0]		BIDX[5:0]					
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bit 15 – FLST: Filter List

Indicates the filter list of the matching filter element.

Value	Description
0	Standard Filter List.
1	Extended Filter List.

#### Bits 14:8 – FIDX[6:0]: Filter Index

Index of matching filter element. Range is 0 to SIDFC.LSS - 1 (standard) or XIDFC.LSE - 1 (extended).

#### Bits 7:6 – MSI[1:0]: Message Storage Indicator

This field defines the message storage information to a FIFO.

Value	Name	Description
0x0	NONE	No FIFO selected.
0x1	LOST	FIFO message lost.



Value	Name	Description
0x2	FIFO0	Message stored in FIFO 0.
0x3	FIFO1	Message stored in FIFO 1.

**Bits 5:0 – BIDX[5:0]: Buffer Index**

Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

### 35.8.25. New Data 1

**Name:** NDAT1  
**Offset:** 0x98  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24
Access	R/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
	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
Access	R/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
	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8
Access	R/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
	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
Access	R/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 – NDn: New Data n [n = 0..31]

The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.

### 35.8.26. New Data 2

**Name:** NDAT2  
**Offset:** 0x9C  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24
Access	R/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
	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
Access	R/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
	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8
Access	R/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
	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
Access	R/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 – NDn: New Data [n = 32..64]

The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.

### 35.8.27. Rx FIFO 0 Configuration

**Name:** RXF0C  
**Offset:** 0xA0  
**Reset:** 0x00000000  
**Property:** Write-restricted

Bit	31	30	29	28	27	26	25	24
	F0OM		F0WM[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	23	22	21	20	19	18	17	16
		F0S[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
	F0SA[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
	F0SA[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 31 – F0OM: FIFO 0 Operation Mode

FIFO 0 can be operated in blocking or in overwrite mode.

Value	Description
0	FIFO 0 blocking mode.
1	FIFO 0 overwrite mode.

#### Bits 30:24 – F0WM[6:0]: Rx FIFO 0 Watermark

Value	Description
0	Watermark interrupt disabled.
1 - 64	Level for Rx FIFO 0 watermark interrupt (IR.RF0W).
>64	Watermark interrupt disabled.

#### Bits 22:16 – F0S[6:0]: Rx FIFO 0 Size

The Rx FIFO 0 elements are indexed from 0 to F0S - 1.

Value	Description
0	No Rx FIFO 0
1 - 64	Number of Rx FIFO 0 elements.
>64	Values greater than 64 are interpreted as 64.

**Bits 15:0 – F0SA[15:0]: Rx FIFO 0 Start Address**

Start address of Rx FIFO 0 in Message RAM. When the CAN module addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored. Bits 1 to 0 will always be read back as “00”.

### 35.8.28. Rx FIFO 0 Status

**Name:** RXF0S  
**Offset:** 0xA4  
**Reset:** 0x00000000  
**Property:** Read-only

Bit	31	30	29	28	27	26	25	24
							RF0L	F0F
Access							R	R
Reset							0	0
Bit	23	22	21	20	19	18	17	16
			F0PI[5:0]					
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
			F0GI[5:0]					
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	F0FL[6:0]							
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

#### Bit 25 – RF0L: Rx FIFO 0 Message Lost

This bit is a copy of interrupt flag IR.RF0L. When IR.RF0L is reset, this bit is also reset.

Overwriting the oldest message when RXF0C.F0OM = '1' will not set this flag.

Value	Description
0	No Rx FIFO 0 message lost.
1	Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero.

#### Bit 24 – F0F: Rx FIFO 0 Full

Value	Description
0	Rx FIFO 0 not full.
1	Rx FIFO 0 full.

#### Bits 21:16 – F0PI[5:0]: Rx FIFO 0 Put Index

Rx FIFO 0 write index pointer, range 0 to 63.

#### Bits 13:8 – F0GI[5:0]: Rx FIFO 0 Get Index

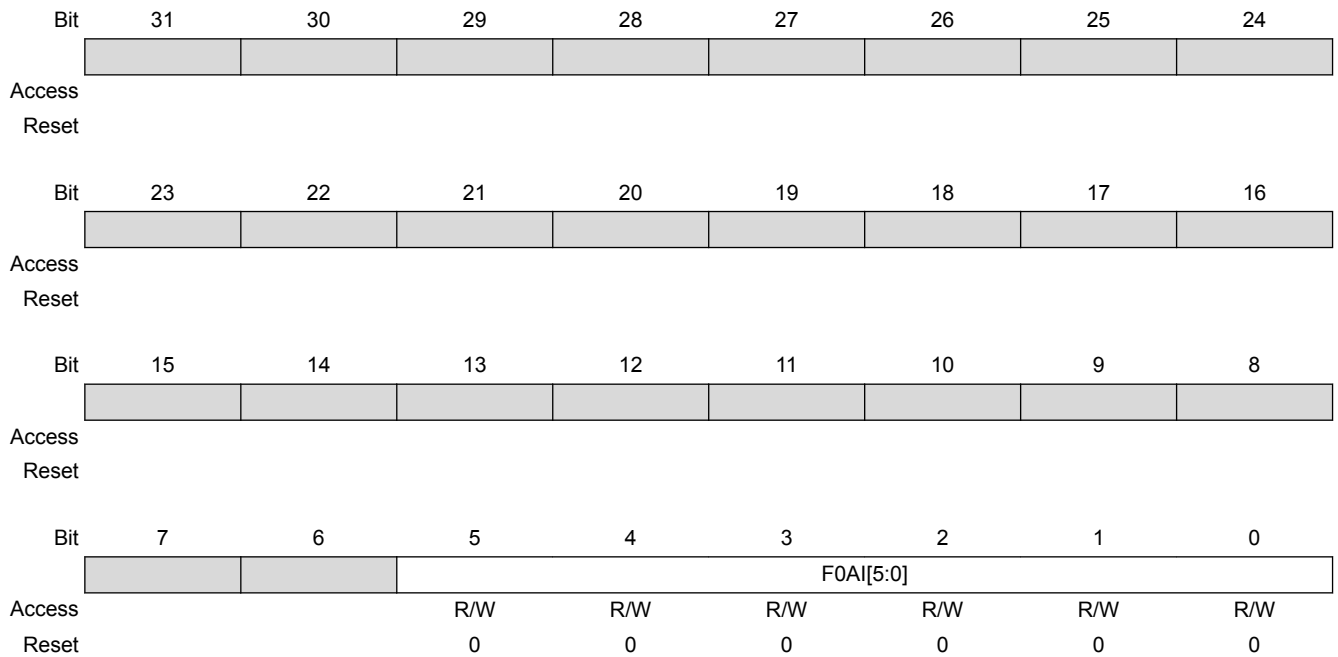
Rx FIFO 0 read index pointer, range 0 to 63.

**Bits 6:0 – F0FL[6:0]: Rx FIFO 0 Fill Level**

Number of elements stored in Rx FIFO 0, range 0 to 64.

### 35.8.29. Rx FIFO 0 Acknowledge

**Name:** RXF0A  
**Offset:** 0xA8  
**Reset:** 0x00000000  
**Property:** -



#### Bits 5:0 – F0AI[5:0]: Rx FIFO 0 Acknowledge Index

After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index RXF0S.F0GI to F0AI + 1 and update the FIFO 0 Fill Level RXF0S.F0FL.



### 35.8.30. Rx Buffer Configuration

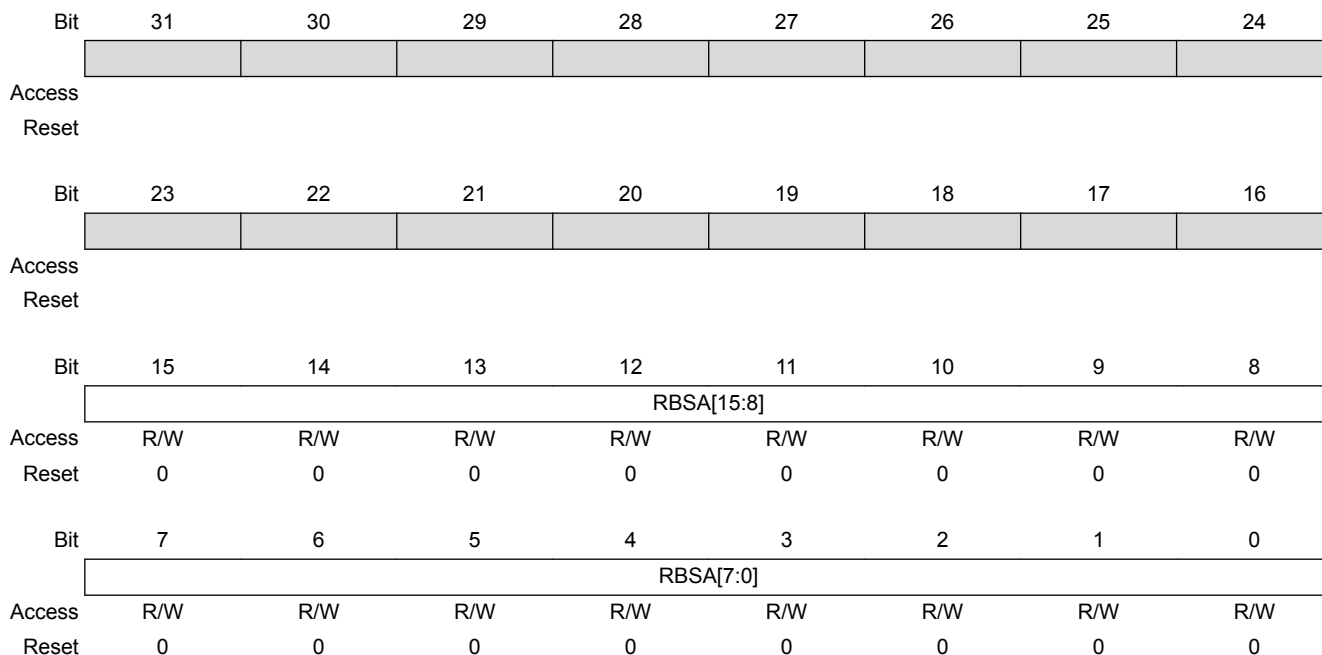
This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

**Name:** RXBC

**Offset:** 0xAC

**Reset:** 0x00000000

**Property:** Write-restricted



#### Bits 15:0 – RBSA[15:0]: Rx Buffer Start Address

Configures the start address of the Rx Buffers section in the Message RAM. Also used to reference debug message A,B,C. When the CAN module addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored. Bits 1 to 0 will always be read back as “00”.

### 35.8.31. Rx FIFO 1 Configuration

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

**Name:** RXF1C

**Offset:** 0xB0

**Reset:** 0x00000000

**Property:** Write-restricted

Bit	31	30	29	28	27	26	25	24
	F1OM		F1WM[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	23	22	21	20	19	18	17	16
		F1S[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
	F1SA[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
	F1SA[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 31 – F1OM: FIFO 1 Operation Mode

FIFO 1 can be operated in blocking or in overwrite mode.

Value	Description
0	FIFO 1 blocking mode.
1	FIFO 1 overwrite mode.

#### Bits 30:24 – F1WM[6:0]: Rx FIFO 1 Watermark

Value	Description
0	Watermark interrupt disabled.
1 - 64	Level for Rx FIFO 1 watermark interrupt (IR.RF1W).
>64	Watermark interrupt disabled.

#### Bits 22:16 – F1S[6:0]: Rx FIFO 1 Size

The Rx FIFO 1 elements are indexed from 0 to F1S - 1.

Value	Description
0	No Rx FIFO 1
1 - 64	Number of Rx FIFO 1 elements.
>64	Values greater than 64 are interpreted as 64.

**Bits 15:0 – F1SA[15:0]: Rx FIFO 1 Start Address**

Start address of Rx FIFO 1 in Message RAM. When the CAN module addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored. Bits 1 to 0 will always be read back as “00”.

### 35.8.32. Rx FIFO 1 Status

**Name:** RXF1S  
**Offset:** 0xB4  
**Reset:** 0x00000000  
**Property:** Read-only

Bit	31	30	29	28	27	26	25	24
	DMS[1:0]						RF1L	F1F
Access	R	R					R	R
Reset	0	0					0	0
Bit	23	22	21	20	19	18	17	16
			F1PI[5:0]					
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
			F1GI[5:0]					
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
		F1FL[6:0]						
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

#### Bits 31:30 – DMS[1:0]: Debug Message Status

This field defines the debug message status.

Value	Name	Description
0x0	IDLE	Idle state, wait for reception of debug messages, DMA request is cleared.
0x1	DBGA	Debug message A received.
0x2	DBGB	Debug message A, B received.
0x3	DBGC	Debug message A, B, C received, DMA request is set.

#### Bit 25 – RF1L: Rx FIFO 1 Message Lost

This bit is a copy of interrupt flag IR.RF1L. When IR.RF1L is reset, this bit is also reset.

Overwriting the oldest message when RXF1C.FOOM = '1' will not set this flag.

Value	Description
0	No Rx FIFO 1 message lost.
1	Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.

#### Bit 24 – F1F: Rx FIFO 1 Full

Value	Description
0	Rx FIFO 1 not full.
1	Rx FIFO 1 full.

**Bits 21:16 – F1PI[5:0]: Rx FIFO 1 Put Index**

Rx FIFO 1 write index pointer, range 0 to 63.

**Bits 13:8 – F1GI[5:0]: Rx FIFO 1 Get Index**

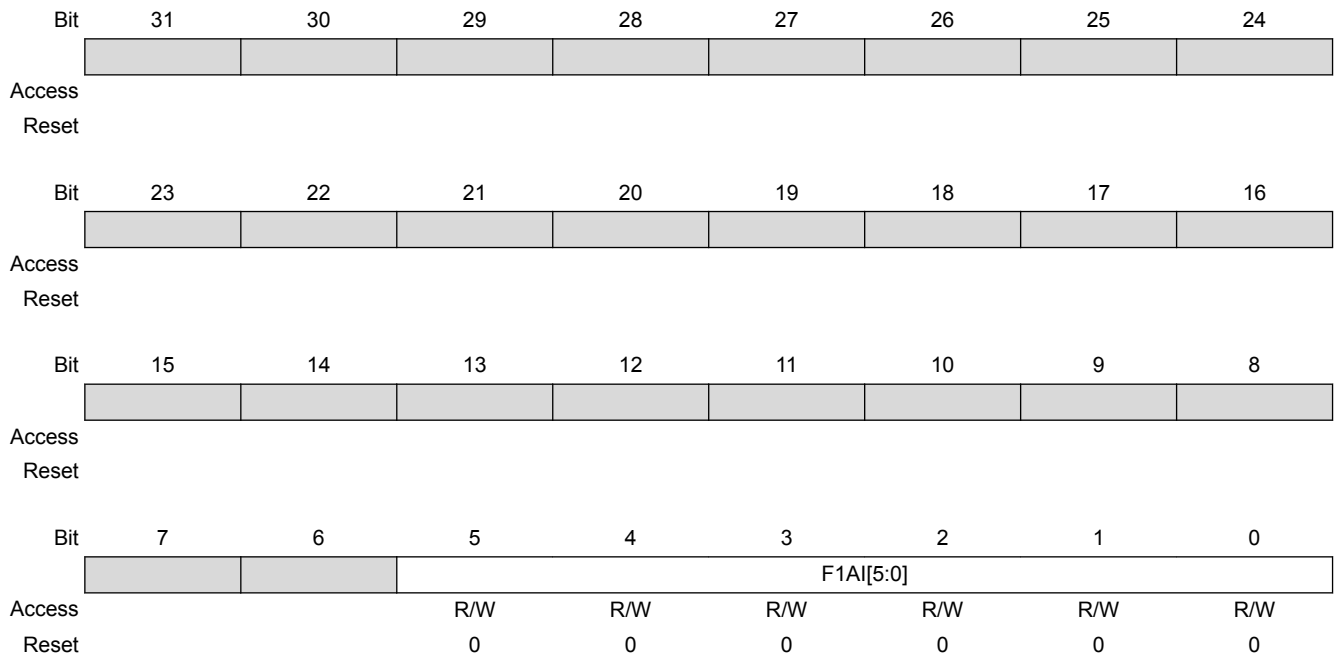
Rx FIFO 1 read index pointer, range 0 to 63.

**Bits 6:0 – F1FL[6:0]: Rx FIFO 1 Fill Level**

Number of elements stored in Rx FIFO 1, range 0 to 64.

### 35.8.33. Rx FIFO 1 Acknowledge

**Name:** RXF1A  
**Offset:** 0xB8  
**Reset:** 0x00000000  
**Property:** -



#### Bits 5:0 – F1AI[5:0]: Rx FIFO 1 Acknowledge Index

After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S.F0GI to F1AI + 1 and update the FIFO 1 Fill Level RXF1S.F1FL.

### 35.8.34. Rx Buffer / FIFO Element Size Configuration

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

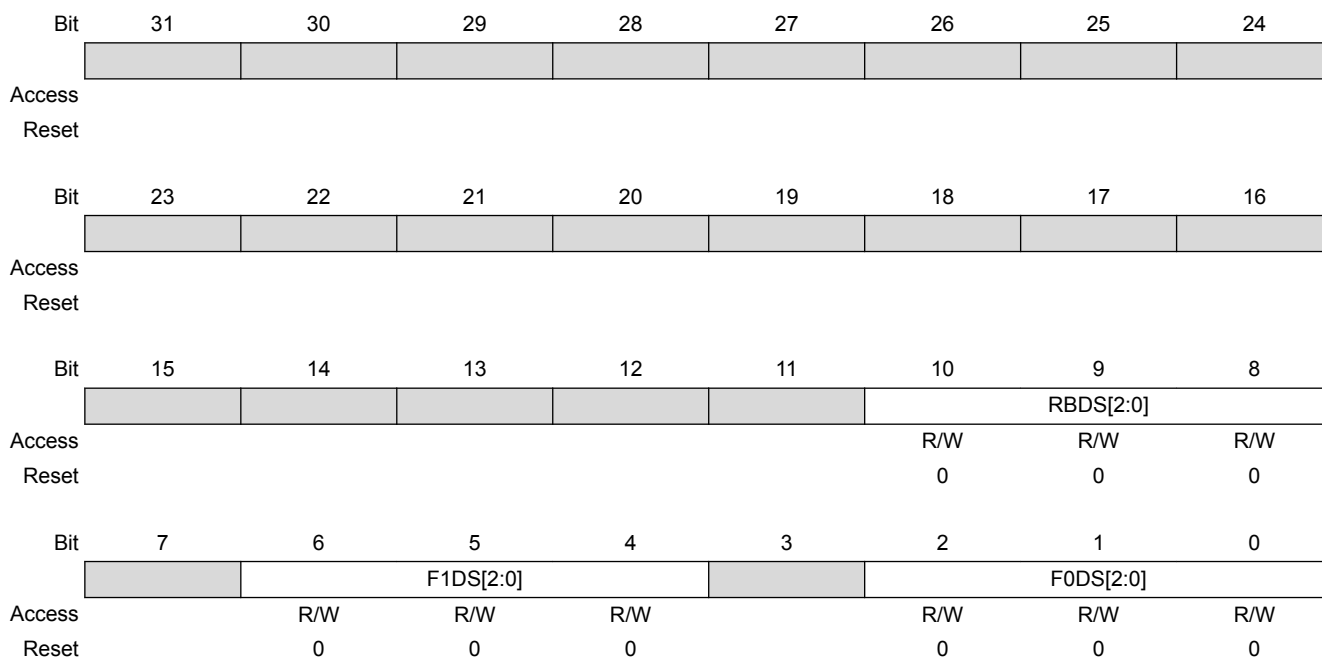
Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

**Name:** RXESC

**Offset:** 0xBC

**Reset:** 0x00000000

**Property:** Write-restricted



#### Bits 10:8 – RBDS[2:0]: Rx Buffer Data Field Size

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer, only the number of bytes as configured by RXESC are stored to the Rx Buffer element. The rest of the frame's data field is ignored.

Value	Name	Description
0x0	DATA8	8 byte data field.
0x1	DATA12	12 byte data field.
0x2	DATA16	16 byte data field.
0x3	DATA20	20 byte data field.
0x4	DATA24	24 byte data field.
0x5	DATA32	32 byte data field.
0x6	DATA48	48 byte data field.
0x7	DATA64	64 byte data field.

**Bits 6:4 – F1DS[2:0]: Rx FIFO 1 Data Field Size**

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx FIFO 1, only the number of bytes as configured by RXESC are stored to the Rx FIFO 1 element. The rest of the frame's data field is ignored.

Value	Name	Description
0x0	DATA8	8 byte data field.
0x1	DATA12	12 byte data field.
0x2	DATA16	16 byte data field.
0x3	DATA20	20 byte data field.
0x4	DATA24	24 byte data field.
0x5	DATA32	32 byte data field.
0x6	DATA48	48 byte data field.
0x7	DATA64	64 byte data field.

**Bits 2:0 – F0DS[2:0]: Rx FIFO 0 Data Field Size**

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx FIFO 0, only the number of bytes as configured by RXESC are stored to the Rx FIFO 0 element. The rest of the frame's data field is ignored.

Value	Name	Description
0x0	DATA8	8 byte data field.
0x1	DATA12	12 byte data field.
0x2	DATA16	16 byte data field.
0x3	DATA20	20 byte data field.
0x4	DATA24	24 byte data field.
0x5	DATA32	32 byte data field.
0x6	DATA48	48 byte data field.
0x7	DATA64	64 byte data field.



### 35.8.35. Tx Buffer Configuration

This register is write-restricted and only writable if bit fields `CCCR.CCE = 1` and `CCCR.INIT = 1`.

**Note:** Be aware that the sum of `TFQS` and `NDTB` may not be greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

**Name:** TXBC

**Offset:** 0xC0

**Reset:** 0x00000000

**Property:** Write-restricted

Bit	31	30	29	28	27	26	25	24
		TFQM	TFQS[5:0]					
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
			NDTB[5:0]					
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
	TBSA[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
	TBSA[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 30 – TFQM: Tx FIFO/Queue Mode

Value	Description
0	Tx FIFO operation.
1	Tx Queue operation.

#### Bits 29:24 – TFQS[5:0]: Transmit FIFO/Queue Size

Value	Description
0	No Tx FIFO/Queue.
1 - 32	Number of Tx Buffers used for Tx FIFO/Queue.
>32	Values greater than 32 are interpreted as 32.

#### Bits 21:16 – NDTB[5:0]: Number of Dedicated Transmit Buffers

Value	Description
0	No Tx FIFO/Queue.
1 - 32	Number of Tx Buffers used for Tx FIFO/Queue.
>32	Values greater than 32 are interpreted as 32.

**Bits 15:0 – TBSA[15:0]: Tx Buffers Start Address**

Start address of Tx Buffers section in Message RAM. When the CAN module addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored. Bits 1 to 0 will always be read back as “00”.

### 35.8.36. Tx FIFO/Queue Status

**Note:** In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indexes indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

**Name:** TXFQS

**Offset:** 0xC4

**Reset:** 0x00000000

**Property:** Read-only

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

#### Bit 21 – TFQF: Tx FIFO/Queue Full

Value	Description
0	Tx FIFO/Queue not full.
1	Tx FIFO/Queue full.

#### Bits 20:16 – TFQPI[4:0]: Tx FIFO/Queue Put Index

Tx FIFO/Queue write index pointer, range 0 to 31.

#### Bits 12:8 – TFGI[4:0]: Tx FIFO/Queue Get Index

Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC.TFQM = '1').

#### Bits 5:0 – TFFL[5:0]: Tx FIFO Free Level

Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC.TFQM = '1').

### 35.8.37. Tx Buffer Element Size Configuration

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

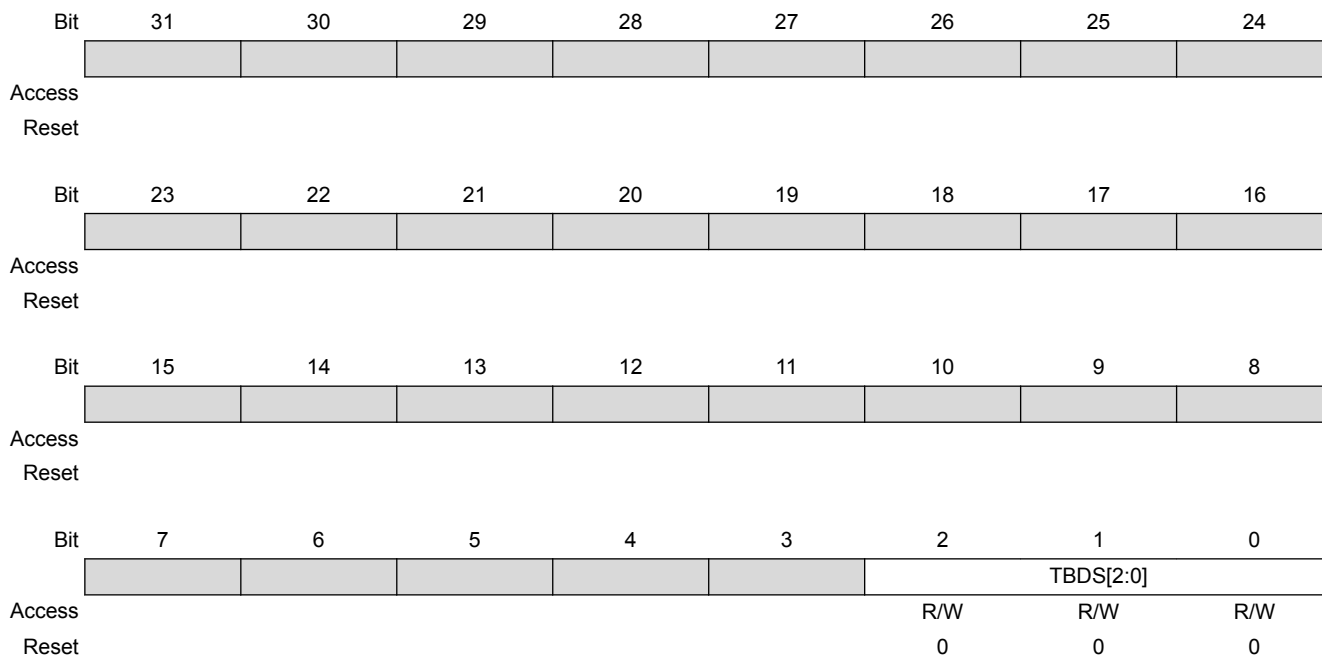
Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes >8 bytes are intended for CAN FD operation only.

**Name:** TXESC

**Offset:** 0xC8

**Reset:** 0x00000000

**Property:** Write-restricted



#### Bits 2:0 – TBDS[2:0]: Tx Buffer Data Field Size

In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as “0xCC” (padding bytes).

Value	Name	Description
0x0	DATA8	8 byte data field.
0x1	DATA12	12 byte data field.
0x2	DATA16	16 byte data field.
0x3	DATA20	20 byte data field.
0x4	DATA24	24 byte data field.
0x5	DATA32	32 byte data field.
0x6	DATA48	48 byte data field.
0x7	DATA64	64 byte data field.

### 35.8.38. Tx Buffer Request Pending

**Note:** TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is canceled immediately, the corresponding TXBRP bit is reset.

**Name:** TXBRP  
**Offset:** 0xCC  
**Reset:** 0x00000000  
**Property:** Read-only

Bit	31	30	29	28	27	26	25	24
	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24
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
	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
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
	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8
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
	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 31:0 – TRPn: Transmission Request Pending

Each Tx Buffer has its own Transmission Request Pending bit.

The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR.

TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID).

A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled via TXBCF

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions.

Value	Description
0	No transmission request pending.
1	Transmission request pending.

### 35.8.39. Tx Buffer Add Request

**Note:** If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit is already set), this add request is ignored.

**Name:** TXBAR

**Offset:** 0xD0

**Reset:** 0x00000000

**Property:** -

Bit	31	30	29	28	27	26	25	24
	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24
Access	R/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
	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Access	R/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
	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8
Access	R/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
	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
Access	R/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 – ARn: Add Request

Each Tx Buffer has its own Add Request bit.

Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

### 35.8.40. Tx Buffer Cancellation Request

**Name:** TXBCR  
**Offset:** 0xD4  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24
Access	R/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
	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
Access	R/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
	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8
Access	R/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
	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Access	R/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 – CRn: Cancellation Request

Each Tx Buffer has its own Cancellation Request bit.

Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset.

Value	Description
0	No cancellation pending.
1	Cancellation pending.



### 35.8.41. Tx Buffer Transmission Occurred

**Name:** TXBTO  
**Offset:** 0xD8  
**Reset:** 0x00000000  
**Property:** Read-only

Bit	31	30	29	28	27	26	25	24
	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24
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
	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
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
	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8
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
	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 31:0 – TOn: Transmission Occurred

Each Tx Buffer has its own Transmission Occurred bit.

The bits are set when the corresponding TXBRP bit is cleared after a successful transmission.

The bits are reset when a new transmission is requested by writing '1' to the corresponding bit of register TXBAR.

### 35.8.42. Tx Buffer Cancellation Finished

**Name:** TXBCF  
**Offset:** 0xDC  
**Reset:** 0x00000000  
**Property:** Read-only

Bit	31	30	29	28	27	26	25	24
	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24
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
	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
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
	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8
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
	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 31:0 – CFn: Cancellation Finished

Each Tx Buffer has its own Cancellation Finished bit.

The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately.

The bits are reset when a new transmission is requested by writing '1' to the corresponding bit of register TXBAR.

### 35.8.43. Tx Buffer Transmission Interrupt Enable

**Name:** TXBTIE  
**Offset:** 0xE0  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24
Access	R/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
	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
Access	R/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
	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8
Access	R/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
	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
Access	R/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 – TIE<sub>n</sub>: Transmission Interrupt Enable

Each Tx Buffer has its own Transmission Interrupt Enable bit.

Value	Description
0	Transmission interrupt disabled.
1	Transmission interrupt enabled.

### 35.8.44. Tx Buffer Cancellation Finished Interrupt Enable

**Name:** TXBCIE  
**Offset:** 0xE4  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24
Access	R/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
	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
Access	R/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
	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8
Access	R/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
	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
Access	R/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 – CFIE<sub>n</sub>: Cancellation Finished Interrupt Enable

Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit.

Value	Description
0	Cancellation finished interrupt disabled.
1	Cancellation finished interrupt enabled.

### 35.8.45. Tx Event FIFO Configuration

This register is write-restricted and only writable if bit fields CCCR.CCE = 1 and CCCR.INIT = 1.

**Name:** TXEFC

**Offset:** 0xF0

**Reset:** 0x00000000

**Property:** Write-restricted

Bit	31	30	29	28	27	26	25	24
	EFWM[5:0]							
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
	EFS[5:0]							
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	EFSA[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
	EFSA[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 29:24 – EFWM[5:0]: Event FIFO Watermark

Value	Description
0	Watermark interrupt disabled.
1 - 32	Level for Tx Event FIFO watermark interrupt (IR.TEFW).
>32	Watermark interrupt disabled.

#### Bits 21:16 – EFS[5:0]: Event FIFO Size

The Tx Event FIFO elements are indexed from 0 to EFS - 1.

Value	Description
0	Tx Event FIFO disabled
1 - 32	Number of Tx Event FIFO elements.
>32	Values greater than 32 are interpreted as 32.

#### Bits 15:0 – EFSA[15:0]: Event FIFO Start Address

Start address of Tx Event FIFO in Message RAM. When the CAN module addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, i.e.

only bits 15 to 2 are evaluated, the two least significant bits are ignored. Bits 1 to 0 will always be read back as "00".

### 35.8.46. Tx Event FIFO Status

**Name:** TXEFS  
**Offset:** 0xF4  
**Reset:** 0x00000000  
**Property:** Read-only

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

#### Bit 25 – TEFL: Tx Event FIFO Element Lost

This bit is a copy of interrupt flag IR.TEFL. When IR.TEFL is reset, this bit is also reset.

Value	Description
0	No Tx Event FIFO element lost.
1	Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.

#### Bit 24 – EFF: Event FIFO Full

Value	Description
0	Tx Event FIFO not full.
1	Tx Event FIFO full.

#### Bits 20:16 – EFP[4:0]: Event FIFO Put Index

Tx Event FIFO write index pointer, range 0 to 31.

#### Bits 12:8 – EFGI[4:0]: Event FIFO Get Index

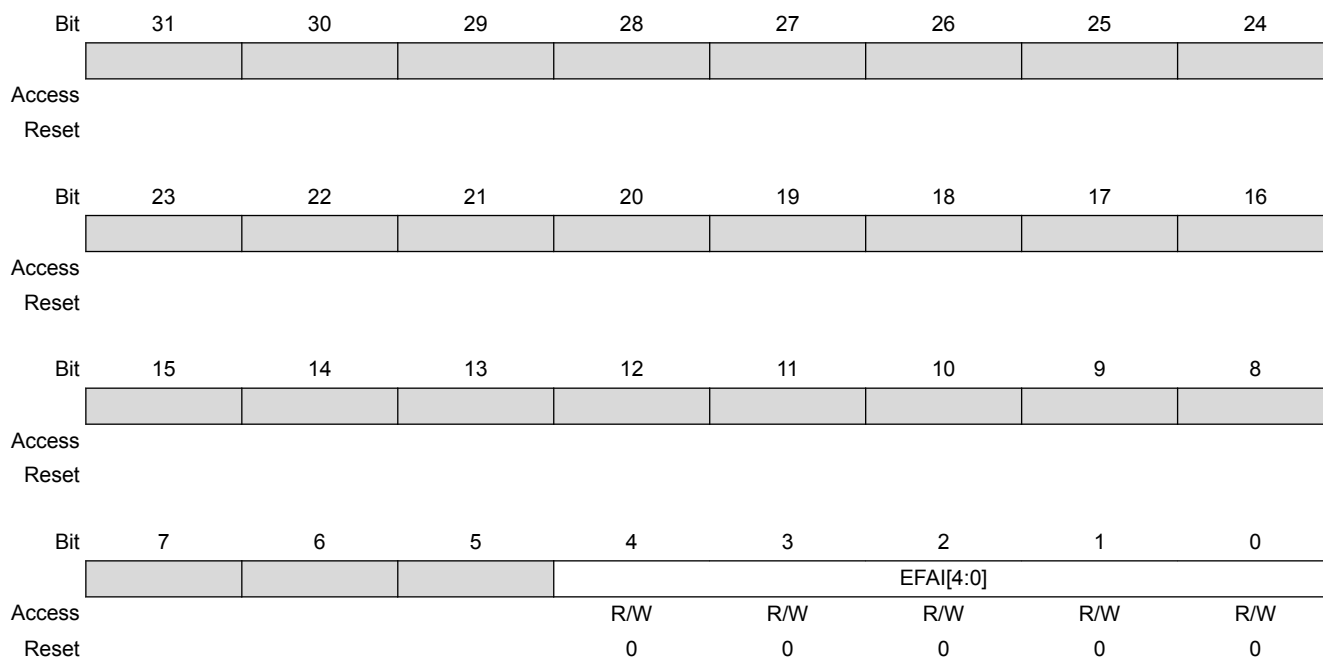
Tx Event FIFO read index pointer, range 0 to 31.

#### Bits 4:0 – EFFI[4:0]: Event FIFO Fill Level

Number of elements stored in Tx Event FIFO, range 0 to 32.

### 35.8.47. Tx Event FIFO Acknowledge

**Name:** TXEFA  
**Offset:** 0xF8  
**Reset:** 0x00000000  
**Property:** -



#### Bits 4:0 – EFAI[4:0]: Event FIFO Acknowledge Index

After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS.EFGI to EFAI + 1 and update the FIFO 0 Fill Level TXEFS.EFFL.

## 35.9. Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the CAN module.

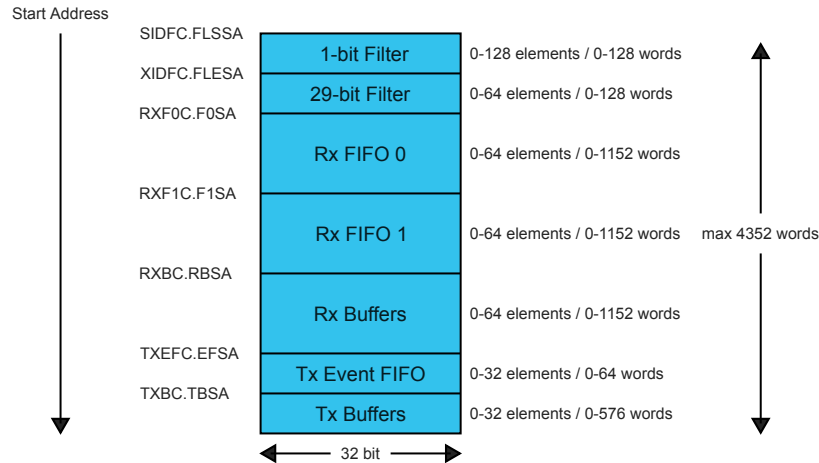
### 35.9.1. Message RAM Configuration

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The CAN module can be configured to allocate up to 4352 words in the Message RAM. It is not necessary to configure each of the sections listed in the figure below, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO 0, Rx FIFO 1, Rx Buffers, and Tx Buffers via RXESC.F0DS, RXESC.F1DS, RXESC.RBDS, and TXESC.TBDS.



**Figure 35-12. Message RAM Configuration**



When the CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses (i.e. only bits 15 to 2 are evaluated and the two LSBs are ignored).



**Warning:** The CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

### 35.9.2. Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the table below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

**Table 35-8. Rx Buffer and FIFO Element**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R0	E	X	R	ID[28:0]																												
R1	A	FIDX[6:0]									F	B	DLC[3:0]			RXTS[15:0]																
				DB3[7:0]								DB2[7:0]			DB1[7:0]						DB0[7:0]											
R3				DB7[7:0]								DB6[7:0]			DB5[7:0]						DB4[7:0]											
...				...								...			...						...											
Rn				DBm[7:0]								DBm-1[7:0]			DBm-2[7:0]						DBm-3[7:0]											

- R0 Bit 31 - ESI: Error State Indicator  
0 : Transmitting node is error active.

- 1 : Transmitting node is error passive.
- R0 Bit 30 - XTD: Extended Identifier  
Signals to the Host whether the received frame has a standard or extended identifier.  
0 : 11-bit standard identifier.  
1 : 29-bit extended identifier.
- R0 Bit 29 - RTR: Remote Transmission Request  
Signals to the Host whether the received frame is a data frame or a remote frame.  
0 : Received frame is a data frame.  
1 : Received frame is a remote frame.  
**Note:** There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = '1'), bit RTR reflects the state of the reserved bit r1.
- R0 Bits 28:0 - ID[28:0]: Identifier  
Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
- R1 Bit 31 - ANMF: Accepted Non-matching Frame  
Acceptance of non-matching frames may be enabled via GFC.ANFS and GFC.ANFE.  
0 : Received frame matching filter index FIDX.  
1 : Received frame did not match any Rx filter element.
- R1 Bits 30:24 - FIDX[6:0]: Filter Index  
0-127 : Index of matching Rx acceptance filter element (invalid if ANMF = '1').  
**Note:** Range is 0 to SIDFC.LSS-1 for standard and 0 to XIDFC.LSE-1 for extended.
- R1 Bits 23:22 - Reserved
- R1 Bit 21 - FDF: FD Format  
0 : Standard frame format.  
1 : CAN FD frame format (new DLC-coding and CRC).
- R1 Bit 20 - BRS: Bit Rate Search  
0 : Frame received without bit rate switching.  
1 : Frame received with bit rate switching.
- R1 Bits 19:16 - DLC[3:0]: Data Length Code  
0-8 : CAN + CAN FD: received frame has 0-8 data bytes.  
9-15 : CAN: received frame has 8 data bytes.  
9-15 : CAN FD: received frame has 12/16/20/24/32/48/64 data bytes.
- R1 Bits 15:0 - RXTS[15:0]: Rx Timestamp  
Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC.TCP.
- R2 Bits 31:24 - DB3[7:0]: Data Byte 3
- R2 Bits 23:16 - DB2[7:0]: Data Byte 2

- R2 Bits 15:8 - DB1[7:0]: Data Byte 1
- R2 Bits 7:0 - DB0[7:0]: Data Byte 0
- R3 Bits 31:24 - DB7[7:0]: Data Byte 7
- R3 Bits 23:16 - DB6[7:0]: Data Byte 6
- R3 Bits 15:8 - DB5[7:0]: Data Byte 5
- R3 Bits 7:0 - DB4[7:0]: Data Byte 4
- ...
- Rn Bits 31:24 - DBm[7:0]: Data Byte m
- Rn Bits 23:16 - DBm-1[7:0]: Data Byte m-1
- Rn Bits 15:8 - DBm-2[7:0]: Data Byte m-2
- Rn Bits 7:0 - DBm-3[7:0]: Data Byte m-3



**Warning:** Depending on the configuration of RXESC, between two and sixteen 32-bit words (Rn = 3 ... 17) are used for storage of a CAN message's data field.

### 35.9.3. Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC.TFQS and TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

**Table 35-9. Tx Buffer Element**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T0	E	X	R	ID[28:0]																												
T1	MM[7:0]			E	F	B	DLC[3:0]																									
				F	D	R																										
				C	F	S																										
T2	DB3[7:0]			DB2[7:0]			DB1[7:0]			DB0[7:0]																						
T3	DB7[7:0]			DB6[7:0]			DB5[7:0]			DB4[7:0]																						
...	...			...			...			...																						
Tn	DBm[7:0]			DBm-1[7:0]			DBm-2[7:0]			DBm-3[7:0]																						

- T0 Bit 31 - ESI: Error State Indicator  
 0 : ESI bit in CAN FD format depends only on error passive flag.  
 1 : ESI bit in CAN FD format transmitted recessive.

**Note:** The ESI bit of the transmit buffer is OR'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive.

- T0 Bit 30 - XTD: Extended Identifier
  - 0 : 11-bit standard identifier.
  - 1 : 29-bit extended identifier.
- T0 Bit 29 - RTR: Remote Transmission Request
  - 0 : Transmit data frame.
  - 1 : Transmit remote frame.

**Note:** When RTR = '1', the CAN transmits a remote frame according to ISO 11898-1, even if CCCR.CME enables the transmission in CAN FD format.
- T0 Bits 28:0 - ID[28:0]: Identifier
 

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
- T1 Bits 31:24 - MM[7:0]: Message Marker
 

Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
- T1 Bit 23 - EFC: Event FIFO Control
  - 0 : Don't store Tx events.
  - 1 : Store Tx events.
- T1 Bit 22 - Reserved
- TR1 Bit 21 - FDF: FD Format
  - 0 : Frame transmitted in Classic CAN format.
  - 1 : Frame transmitted in CAN FD format.
- T1 Bit 20 - BRS: Bit Rate Search
  - 0 : CAN FD frames transmitted without bit rate switching.
  - 1 : CAN FD frames transmitted with bit rate switching.

**Note:** Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled CCCR.FDOE = '1'. Bit BRS is only evaluated when in addition CCCR.BRSE = '1'.
- T1 Bits 19:16 - DLC[3:0]: Data Length Code
  - 0-8 : CAN + CAN FD: received frame has 0-8 data bytes.
  - 9-15 : CAN: received frame has 8 data bytes.
  - 9-15 : CAN FD: received frame has 12/16/20/24/32/48/64 data bytes.
- T1 Bits 15:0 - Reserved
- T2 Bits 31:24 - DB3[7:0]: Data Byte 3
- T2 Bits 23:16 - DB2[7:0]: Data Byte 2
- T2 Bits 15:8 - DB1[7:0]: Data Byte 1
- T2 Bits 7:0 - DB0[7:0]: Data Byte 0
- T3 Bits 31:24 - DB7[7:0]: Data Byte 7
- T3 Bits 23:16 - DB6[7:0]: Data Byte 6
- T3 Bits 15:8 - DB5[7:0]: Data Byte 5
- T3 Bits 7:0 - DB4[7:0]: Data Byte 4
- ...

- Tn Bits 31:24 - DBm[7:0]: Data Byte m
- Tn Bits 23:16 - DBm-1[7:0]: Data Byte m-1
- Tn Bits 15:8 - DBm-2[7:0]: Data Byte m-2
- Tn Bits 7:0 - DBm-3[7:0]: Data Byte m-3

**Note:** Depending on the configuration of TXESC, between two and sixteen 32-bit words (Tn = 3 ... 17) are used for storage of a CAN message's data field.

#### 35.9.4. Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

**Table 35-10. Tx Event FIFO Element**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E0	E S I	X T D	R T R	ID[28:0]																												
E1	MM[7:0]							ET [1:0]	F D F	B R S	DLC[3:0]			TXTS[15:0]																		

- E0 Bit 31 - ESI: Error State Indicator  
0 : Transmitting node is error active.  
1 : Transmitting node is error passive.
- E0 Bit 30 - XTD: Extended Identifier  
0 : 11-bit standard identifier.  
1 : 29-bit extended identifier.
- E0 Bit 29 - RTR: Remote Transmission Request  
0 : Received frame is a data frame.  
1 : Received frame is a remote frame.
- E0 Bits 28:0 - ID[28:0]: Identifier  
Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
- E1 Bits 31:24 - MM[7:0]: Message Marker  
Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.
- E1 Bits 23:22 - ET[1:0]: Event Type  
This field defines the event type.

**Table 35-11. Event Type**

Value	Name	Description
0x0 or 0x3	RES	Reserved
0x1	TXE	Tx event
0x2	TXC	Transmission in spite of cancellation (always set for transmission in DAR mode)

- E1 Bit 21 - FDF: FD Format  
0 : Standard frame format.  
1 : CAN FD frame format (new DLC-coding and CRC).
- E1 Bit 20 - BRS: Bit Rate Search  
0 : Frame received without bit rate switching.  
1 : Frame received with bit rate switching.
- E1 Bits 19:16 - DLC[3:0]: Data Length Code  
0-8 : CAN + CAN FD: received frame has 0-8 data bytes.  
9-15 : CAN: received frame has 8 data bytes.  
9-15 : CAN FD: received frame has 12/16/20/24/32/48/64 data bytes.
- E1 Bits 15:0 - TXTS[15:0]: Tx Timestamp  
Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC.TCP.

### 35.9.5. Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC.FLSSA plus the index of the filter element (0 ... 127).

**Table 35-12. Standard Message ID Filter Element**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S0	SFT [1:0]		SFEC [2:0]		SFID1[10:0]										SFID2[10:0]																	

- Bits 31:30 - SFT[1:0]: Standard Filter Type  
This field defines the standard filter type.

**Table 35-13. Standard Filter Type**

Value	Name	Description
0x0	RANGE	Range filter from SFID1 to SFID2 (SFID2 >= SFID1)
0x1	DUAL	Dual ID filter for SFID1 or SFID2
0x2	CLASSIC	Classic filter: SFID1 = filter, SFID2 = mask
0x3	RES	Reserved

- Bits 29:27 - SFEC[2:0]: Standard Filter Element Configuration

All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match.

**Table 35-14. Standard Filter Element Configuration**

Value	Name	Description
0x0	DISABLE	Disable filter element
0x1	STF0M	Store in Rx FIFO 0 if filter matches
0x2	STF1M	Store in Rx FIFO 1 if filter matches
0x3	REJECT	Reject ID if filter matches
0x4	PRIORITY	Set priority if filter matches.
0x5	PRIF0M	Set priority and store in FIFO 0 if filter matches.
0x6	PRIF1M	Set priority and store in FIFO 1 if filter matches.
0x7	STRXBUF	Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored.

- Bits 26:16 - SFID1[10:0]: Standard Filter ID 1

First ID of standard ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

- Bits 15:11 - Reserved
- Bits 10:0 - SFID2[10:0]: Standard Filter ID 2

This bit field has a different meaning depending on the configuration of SFEC.

5.1. SFEC = “001” ... “110”: Second ID of standard ID filter element.

5.2. SFEC = “111”: Filter for Rx Buffers or for debug messages.

SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

00 = Store message into an Rx Buffer

01 = Debug Message A

10 = Debug Message B

11 = Debug Message C

SFID2[8:6] is used to control the filter event pins at the Extension Interface. A ‘1’ at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one CLK\_CAN\_APB period in case the filter matches.

SFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

### 35.9.6. Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC.FLESA plus two times the index of the filter element (0...63).

**Table 35-15. Extended Message ID Filter Element**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0	EFEC [2:0]		EFID1[28:0]																													
F1	EFT [1:0]		EFID2[28:0]																													

- F0 Bits 31:29 - EFEC[2:0]: Extended Filter Element Configuration

All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110” a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match.

**Table 35-16. Extended Filter Element Configuration**

Value	Name	Description
0x0	DISABLE	Disable filter element.
0x1	STF0M	Store in Rx FIFO 0 if filter matches.
0x2	STF1M	Store in Rx FIFO 1 if filter matches.
0x3	REJECT	Reject ID if filter matches.
0x4	PRIORITY	Set priority if filter matches.
0x5	PRIF0M	Set priority and store in FIFO 0 if filter matches.
0x6	PRIF1M	Set priority and store in FIFO 1 if filter matches.
0x7	STRXBUF	Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored.

- F0 Bits 28:0 - EFID1[28:0]: Extended Filter ID 1

First ID of extended ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of a extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism is used.

- F1 Bits 31:30 - EFT[1:0]: Extended Filter Type

This field defines the extended filter type.

**Table 35-17. Extended Filter Type**

Value	Name	Description
0x0	RANGEM	Range filter from EFID1 to EFID2 (EFID2 >= EFID1).
0x1	DUAL	Dual ID filter for EFID1 or EFID2.
0x2	CLASSIC	Classic filter: EFID1 = filter, EFID2 = mask.
0x3	RANGE	Range filter from EFID1 to EFID2 (EFID2 >= EFID1), XIDAM mask not applied.

- F1 Bits 28:0 - EFID2[28:0]: Extended Filter ID 2



This bit field has a different meaning depending on the configuration of EFEC.

1) EFEC = "001" ... "110" Second ID of standard ID filter element.

2) EFEC = "111" Filter for Rx Buffers or for debug messages.

EFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

00 = Store message into an Rx Buffer

01 = Debug Message A

10 = Debug Message B

11 = Debug Message C

EFID2[8:6] is used to control the filter event pins at the Extension Interface. A '1' at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one CLK\_CAN\_APB period in case the filter matches.

EFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

## 36. TC – Timer/Counter

### 36.1. Overview

There are up to five TC peripheral instances.

Each TC 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 timestamp input events or IO pin edges, allowing for capturing of frequency and/or pulse width.

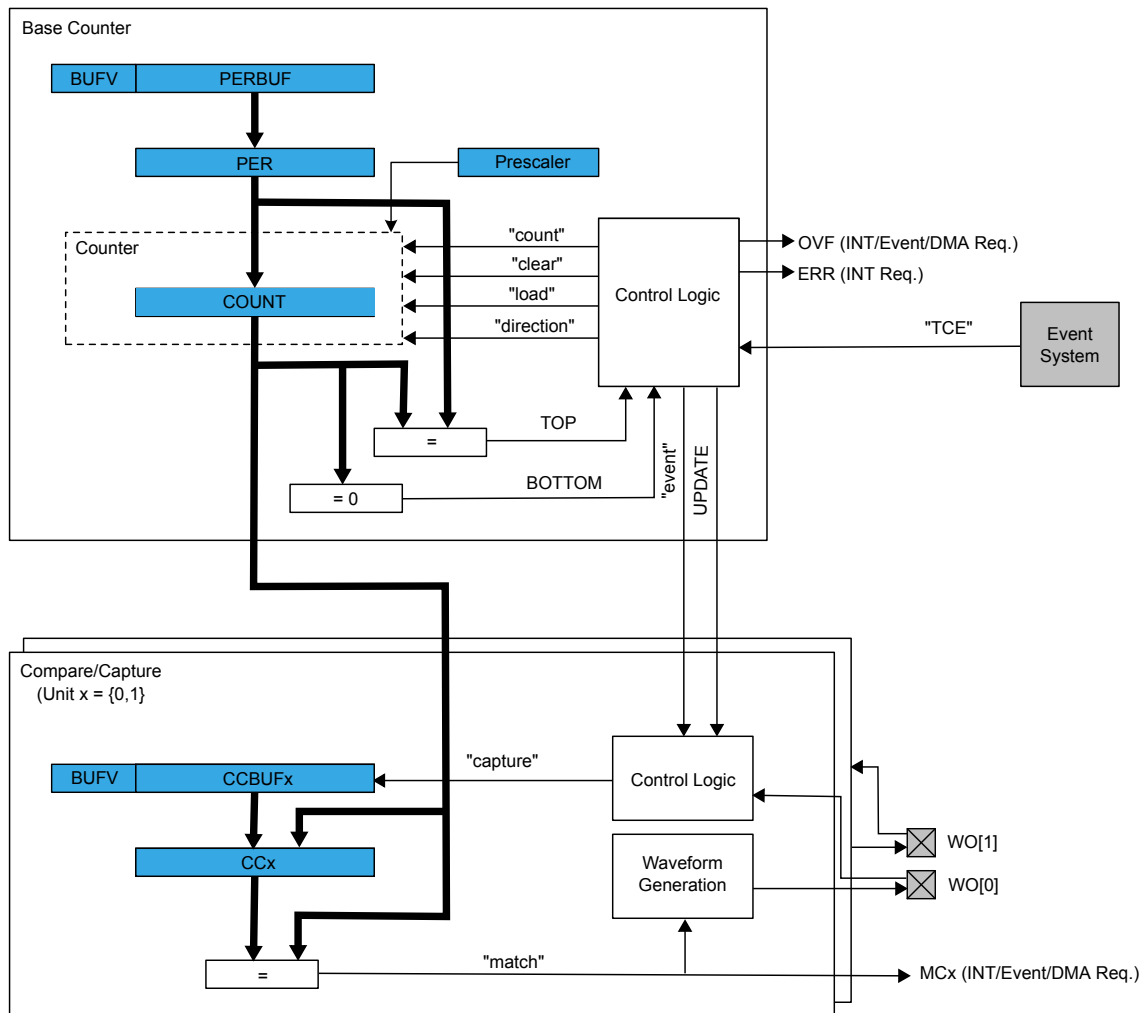
A TC can also perform waveform generation, such as frequency generation and pulse-width modulation.

### 36.2. Features

- Selectable configuration
  - 8-, 16- or 32-bit TC operation, with compare/capture channels
- 2 compare/capture channels (CC) with:
  - Double buffered timer period setting (in 8-bit mode only)
  - Double buffered compare channel
- Waveform generation
  - Frequency generation
  - Single-slope pulse-width modulation
- Input capture
  - Event / IO pin edge capture
  - Frequency capture
  - Pulse-width capture
  - Time-stamp capture
- One input event
- Interrupts/output events on:
  - Counter overflow/underflow
  - Compare match or capture
- Internal prescaler
- DMA support

### 36.3. Block Diagram

Figure 36-1. Timer/Counter Block Diagram



### 36.4. Signal Description

Table 36-1. Signal Description for TC.

Signal Name	Type	Description
WO[1:0]	Digital output	Waveform output
	Digital input	Capture input

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

## 36.5. Product Dependencies

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

### 36.5.1. 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). Refer to *PORT - I/O Pin Controller* for details.

#### Related Links

[PORT: IO Pin Controller](#) on page 444

### 36.5.2. 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. Refer to *PM – Power Manager* for details on the different sleep modes.

#### Related Links

[PM – Power Manager](#) on page 162

### 36.5.3. Clocks

The TC bus clocks (CLK\_TCx\_APB) can be enabled and disabled in the Power Manager. The default state of CLK\_TCx\_APB can be found in the *Peripheral Clock Masking*.

The generic clocks (GCLK\_TCx) are 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.

Note that TC0 and TC1 share a peripheral clock channel, as do TC2 and TC3. For this reason they cannot be set to different clock frequencies.

#### Related Links

[Peripheral Clock Masking](#) on page 135

### 36.5.4. 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](#) on page 334

### 36.5.5. 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](#) on page 41

### 36.5.6. Events

The events of this peripheral are connected to the Event System.

## Related Links

[EVSYS – Event System](#) on page 474

### 36.5.7. 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.

### 36.5.8. Register Access Protection

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

- Interrupt Flag Status and Clear register (INTFLAG)
- Status register (STATUS)
- Count register (COUNT)
- Period and Period Buffer registers (PER, PERBUF)
- Compare/Capture Value registers and Compare/Capture Value Buffer registers (CCx, CCBUFx)

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

When the CPU is halted in debug mode, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger.

### 36.5.9. Analog Connections

Not applicable.

## 36.6. Functional Description

### 36.6.1. Principle of Operation

The following definitions are used throughout the documentation:

**Table 36-2. 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> on page 765.
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.
Timer	The timer/counter clock control is handled by an internal source

Name	Description
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 TC instance has up to two compare/capture channels (CC0 and CC1).

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 Counter register (COUNT), compare and capture registers with buffers (CCx and CCBUFx) can be configured as 8-, 16- or 32-bit registers, with according MAX values. Mode settings determine the maximum range of the counter. Each buffer register has a buffer valid (BUFV) flag that indicates when the buffer contains a new value.

In 8-bit mode, Period Value (PER) and Period Buffer Value (PERBUF) registers are 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.

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 IO pin or internal event from Event System.

## 36.6.2. Basic Operation

### 36.6.2.1. 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 Enable (ENABLE) and Software Reset (SWRST) bits
- Drive Control register (DRVCTRL)
- Wave register (WAVE)
- 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 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 WAVE register (WAVE.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. If desired, 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 A register (CTRLA.CAPTEN).
8. If desired, enable inversion of the waveform output or IO pin input signal for individual channels via the Invert Enable bit group in the Drive Control register (DRVCTRL.INVEN).

### 36.6.2.2. 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 '1' 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. Refer to the [CTRLA](#) register for details.

The TC should be disabled before the TC is reset in order to avoid undefined behavior.

### 36.6.2.3. 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.

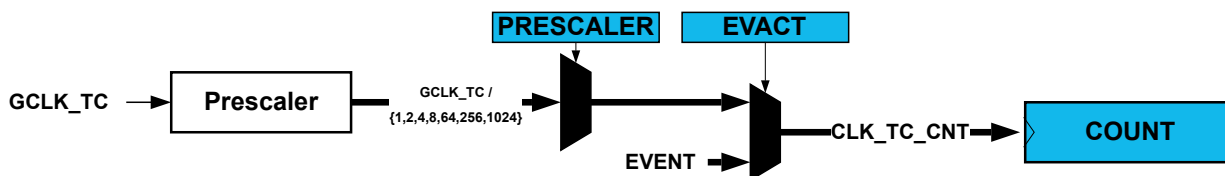
If the prescaler value is higher than one, the counter update condition can be optionally executed on the next GCLK\_TCx clock pulse or the next prescaled clock pulse. For further details, refer to Prescaler (CTRLA.PRESCALER) and Counter Synchronization (CTRLA.PRESYNC) description.

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 36-2. Prescaler



### 36.6.2.4. 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 Value and Period Buffer Value registers (PER and PERBUF).

- 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. TC0 is paired with TC1, and TC2 is paired with TC3. TC4 does not support 32-bit resolution. When paired, the TC peripherals are configured using the registers of the even-numbered TC (TC0 or TC2 respectively). The odd-numbered partner (TC1 or TC3 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.

### 36.6.2.5. Counter Operations

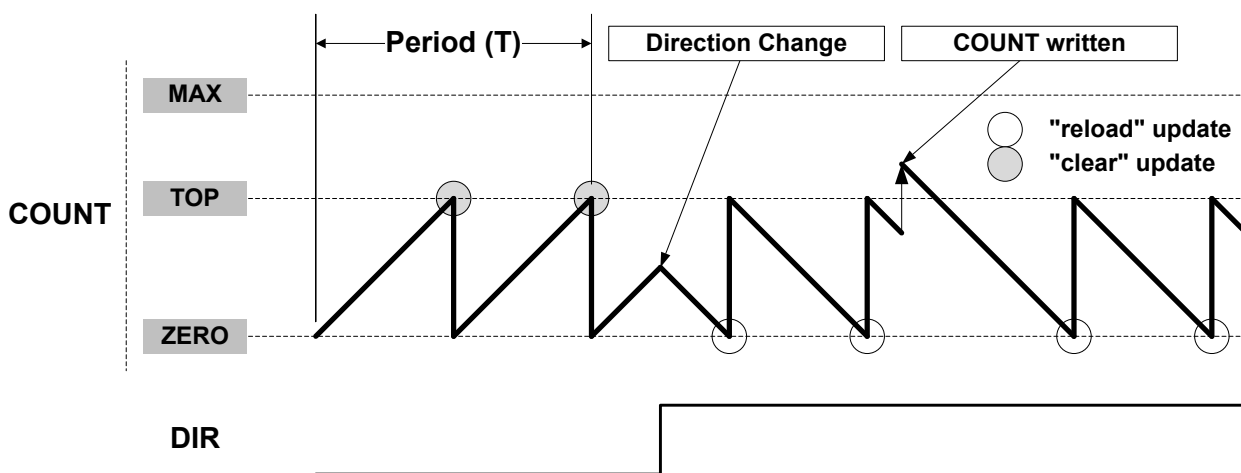
Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each TC clock input (CLK\_TC\_CNT). A counter clear or reload marks the end of the current counter cycle and the start of a new one.

The counting direction is set by the Direction bit in the Control B register (CTRLB.DIR). If 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. When it is counting down, 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).

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 36-3. Counter Operation



Due to asynchronous clock domains, the internal counter settings are written when the synchronization is complete. Normal operation must be used when using the counter as timer base for the capture channels.

#### 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 be loaded with the starting value (ZERO or TOP, depending on direction set by CTRLBSET.DIR or



CTRLBCLR.DIR). 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).

### 36.6.2.6. 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.

The Channel x Compare Buffer (CCBUFx) 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 forced update command (CTRLBSET.CMD=UPDATE). For further details, refer to [Double Buffering](#) on page 768. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses and ensures glitch-free output.

#### 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 (WAVE.WAVEGEN).
2. Optionally invert the waveform output WO[x] by writing the corresponding Output Waveform x Invert Enable 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\_TC\_CNT (see Normal Frequency Operation). An interrupt/and or event can be generated on comparison match if enabled. The same condition generates a DMA request.

There are four 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:

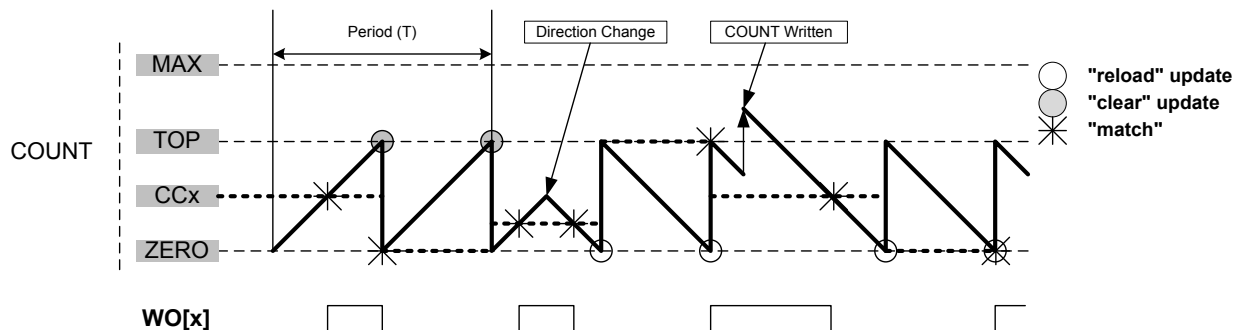
- Normal frequency (NFRQ)
- Match frequency (MFRQ)
- Normal pulse-width modulation (NPWM)
- Match pulse-width modulation (MPWM)

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.

### 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.

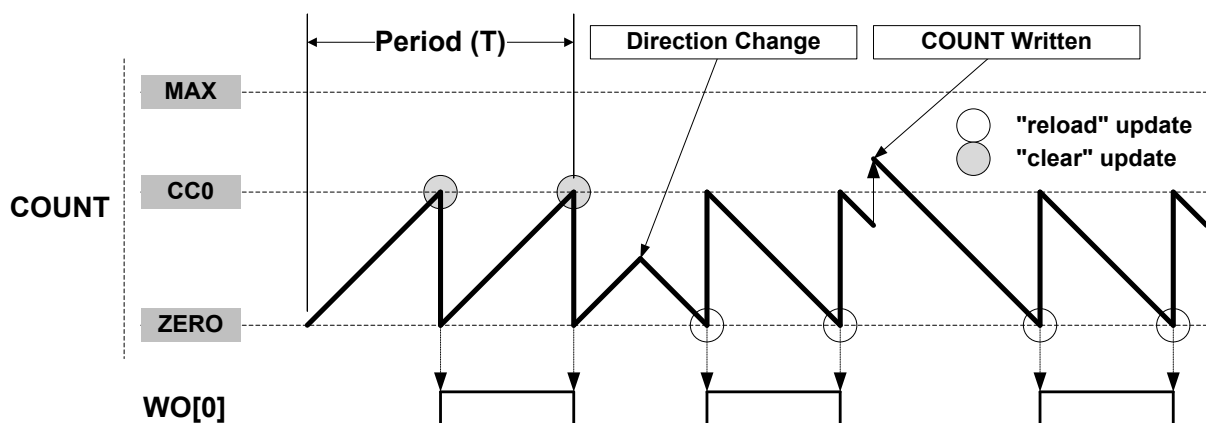
Figure 36-4. Normal Frequency Operation



### 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 36-5. Match Frequency 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 ( $f_{GCLK\_TCC}$ ), and can be calculated by the following equation:

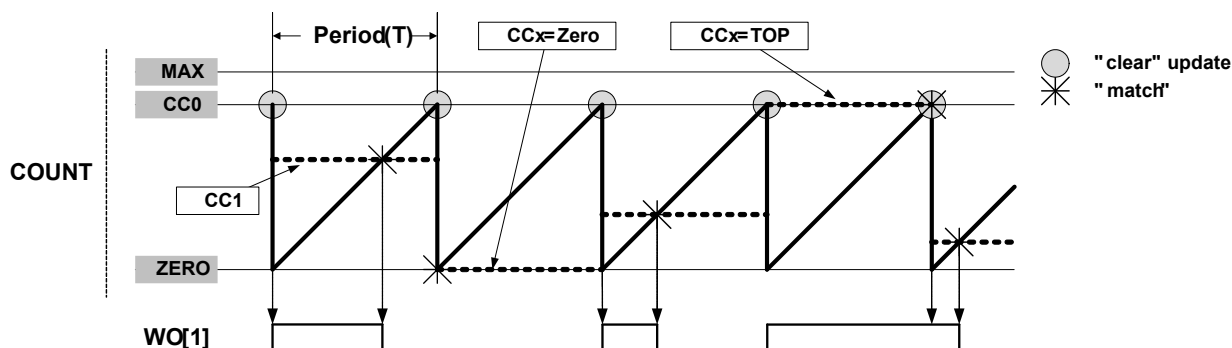
$$f_{PWM\_SS} = \frac{f_{GCLK\_TC}}{N(TOP+1)}$$

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 36-6. Match PWM Operation



The table below shows the update counter and overflow event/interrupt generation conditions in different operation modes.

Table 36-3. 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

### Related Links

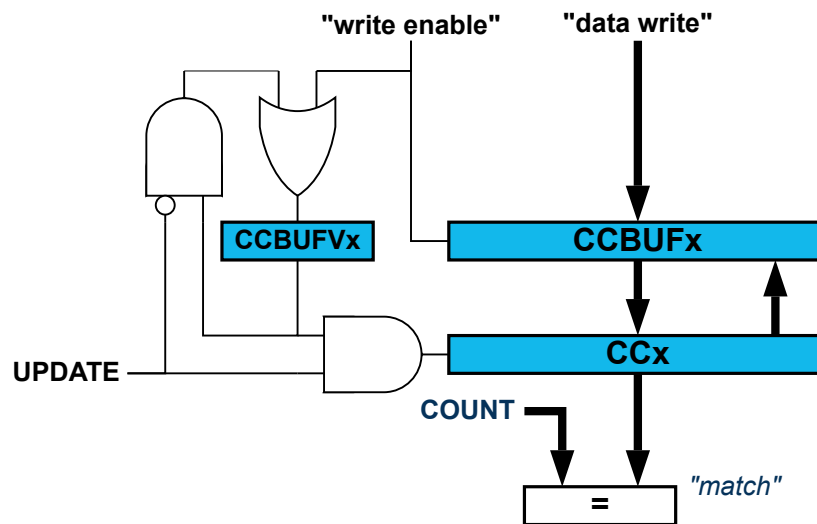
[PORT: IO Pin Controller](#) on page 444

### 36.6.2.7. Double Buffering

The Compare Channels (CCx) registers, and the Period (PER) register in 8-bit mode, are double buffered. Each buffer register has a buffer valid (CCBUFVx or PERBUFV) bit in the STATUS register, which indicates that the buffer register contains a new valid value that can be copied into the corresponding register. As long as the buffer valid status flag (PERBUFV or CCBUFVx) is set to '1', a write to the corresponding PER or CCx register will generate a Capture Overflow Error (ERR).

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' by writing CTRLBCLR.LUPD to '1', double buffering is enabled: the data from buffer registers will be copied into the corresponding register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command. Then the buffer valid flags bit in the STATUS register are automatically cleared by hardware or software. Double buffering is not applied for capture mode. A compare register is double buffered as in the following figure.

Figure 36-7. Compare Channel Double Buffering



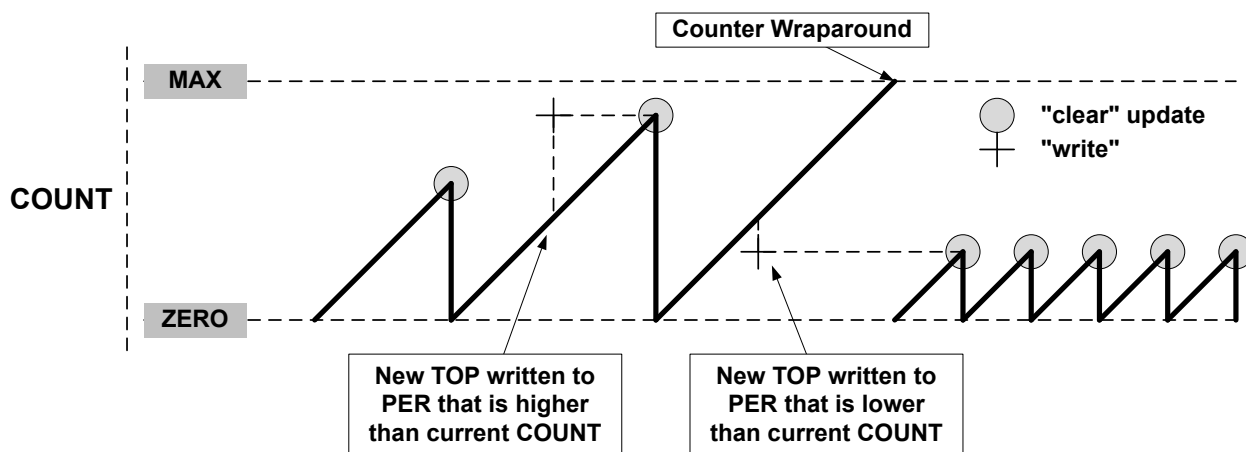
Both the registers (PER/CCx) and corresponding buffer registers (PERBUF/CCBUFx) are available in the I/O register map, and the double buffering feature is not mandatory. The double buffering is disabled by writing a '1' to CTRLBSET.LUPD. This allows initialization and bypassing of the buffer register and the double buffering feature.

Note: In NFRQ, MFRQ or PWM down-counting counter mode (CTRLBSET.DIR=1), the PER register and the PERBUF register are written simultaneously if double buffering is active or as soon as double buffering is activated (CTRLBCLR.LUPD=0).

#### Changing the Period

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 double buffering is off (CTRLBSET.LUPD=1), any period update is effective after the synchronization delay.

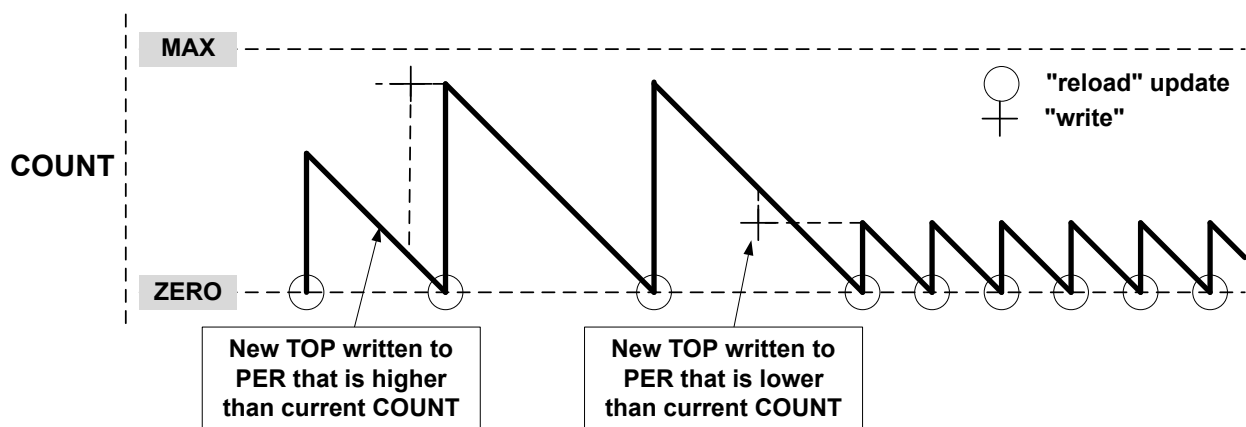
Figure 36-8. Unbuffered Single-Slope Up-Counting Operation



A counter wraparound can occur in any operation mode when up-counting without buffering, see [Figure 36-8 Unbuffered Single-Slope Up-Counting Operation](#) on page 769.

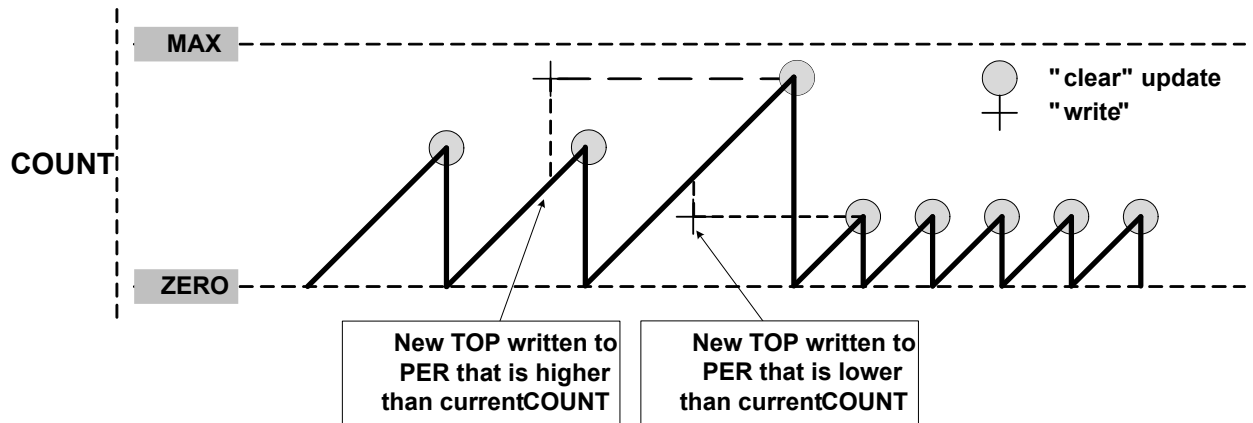
COUNT and TOP are continuously compared, so when a new TOP value that is lower than current COUNT is written to TOP, COUNT will wrap before a compare match.

Figure 36-9. Unbuffered Single-Slope Down-Counting 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 36-10 Changing the Period Using Buffering](#) on page 770. This prevents wraparound and the generation of odd waveforms.

Figure 36-10. Changing the Period Using Buffering



### 36.6.2.8. Capture Operations

To enable and use capture operations, the corresponding Capture Channel x Enable bit in the Control A register (CTRLA.CAPTENx) must be written to '1'.

A capture trigger can be provided by asynchronous IO pin WO[x] for each capture channel or by a TC event. To enable the capture from the IO pin, the Capture On Pin x Enable bit in CTRLA register (CTRLA.COPENx) must be written to '1'.

**Note:** The RETRIGGER, COUNT and START event actions are available only on an event from the Event System.

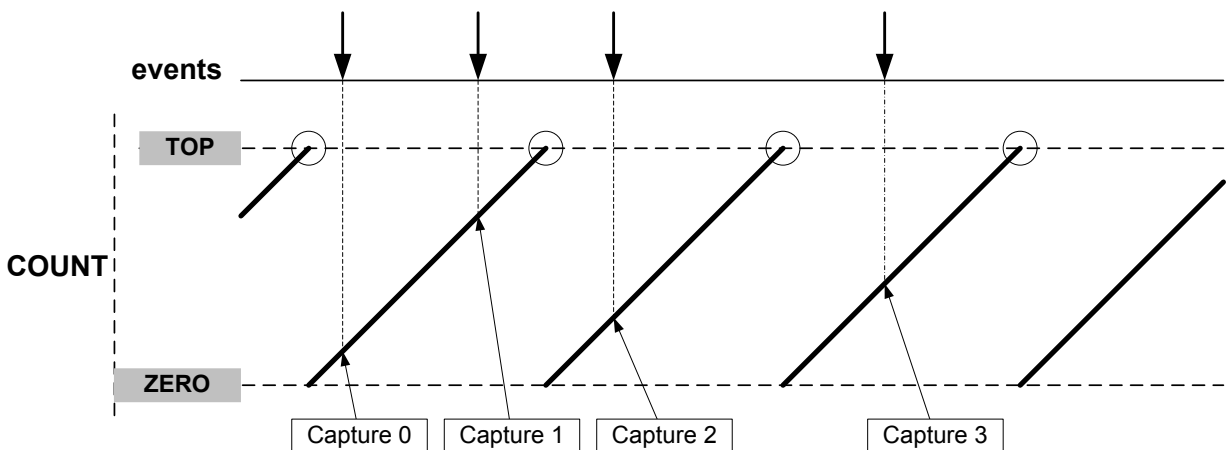
By default, a capture operation is done when a rising edge is detected on the input signal. Capture on falling edge is available, its activation is depending on the input source:

- When the channel is used with a IO pin, write a '1' to the corresponding Invert Enable bit in the Drive Control register (DRVCTRL.INVENx).
- When the channel is counting events from the Event System, write a '1' to the TC Event Input Invert Enable bit in Event Control register (EVCTRL.TCINV).

#### Event Capture Action

The compare/capture channels can be used as input capture channels to capture events from the Event System or from the corresponding IO pin, and give them a timestamp. The following figure shows four capture events for one capture channel.

Figure 36-11. Input Capture Timing



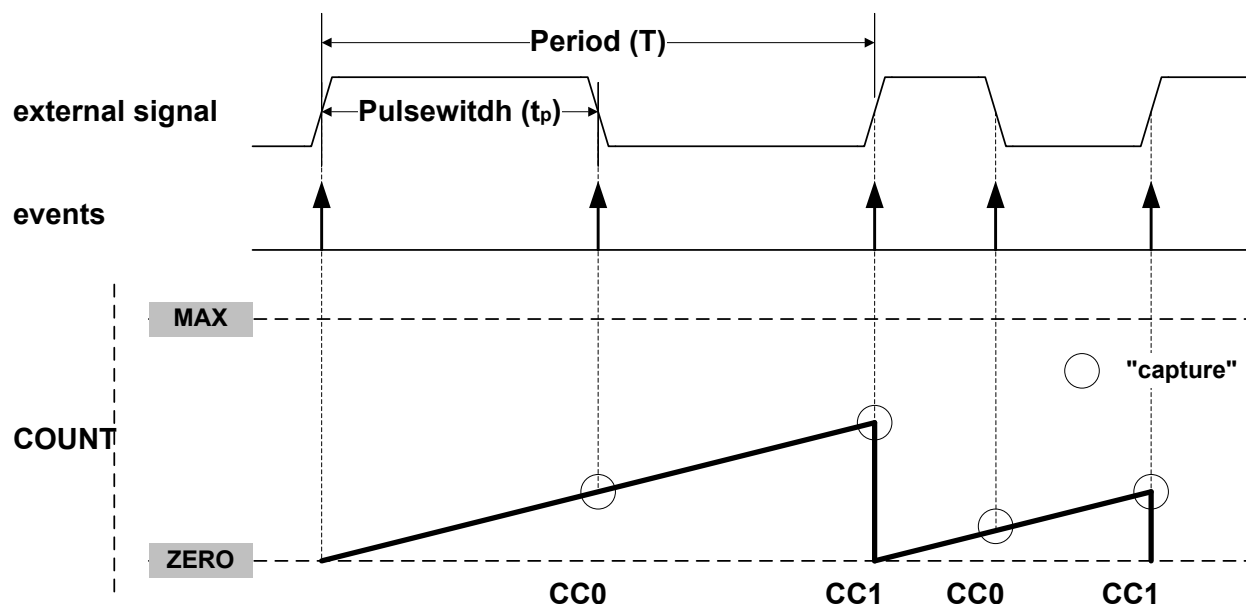
The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

#### 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:

$$f = \frac{1}{T} \qquad \text{dutyCycle} = \frac{t_p}{T}$$

Figure 36-12. PWP Capture



Selecting PWP 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. This also be for DRVCTRL.INVENx if pin capture is enabled.

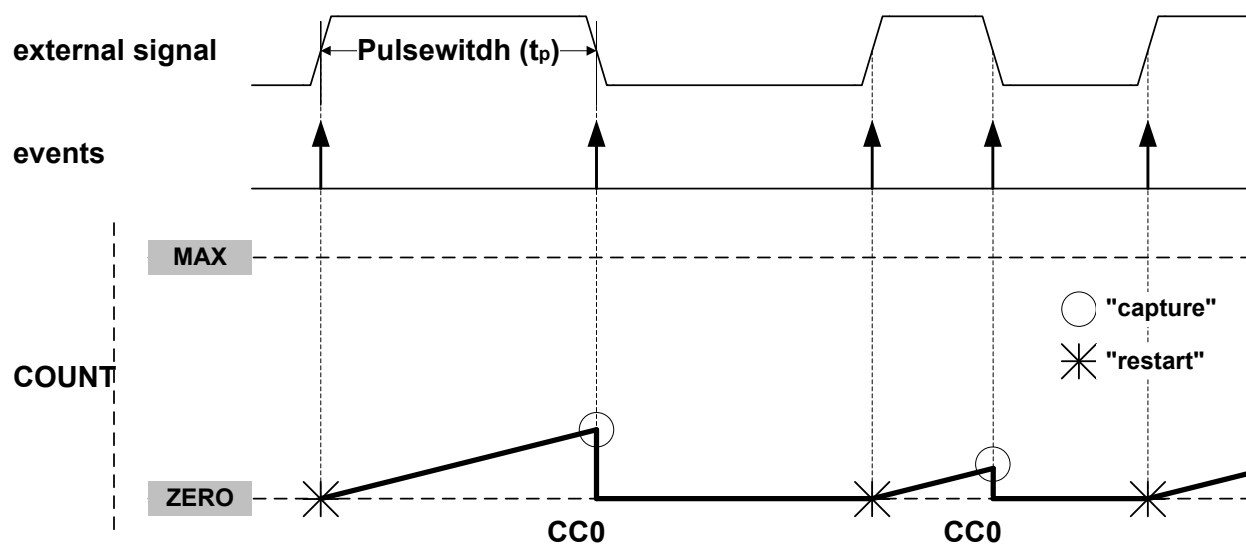
The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

**Note:** The corresponding capture is working only if the channel is enabled in capture mode (CTRLA.CAPTENx=1). If not, the capture action is ignored and the channel is enabled in compare mode of operation. Consequently, both channels must be enabled in order to fully characterize the input.

#### Pulse-Width Capture Action

The TC performs the input capture on the falling edge of the input signal. When the edge is detected, the counter value is cleared and the TC stops counting. When a rising edge is detected on the input signal, the counter restarts the counting operation. To enable the operation on opposite edges, the input signal to capture must be inverted (refer to DRVCTRL.INVEN or EVCTRL.TCEINV).

Figure 36-13. Pulse-Width Capture on Channel 0



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.

### 36.6.3. Additional Features

#### 36.6.3.1. 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 '1' 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.

#### 36.6.3.2. Time-Stamp Capture

This feature is enabled when the Capture Time Stamp (STAMP) Event Action in Event Control register (EVCTRL.EVACT) is selected. The counter TOP value must be smaller than MAX.

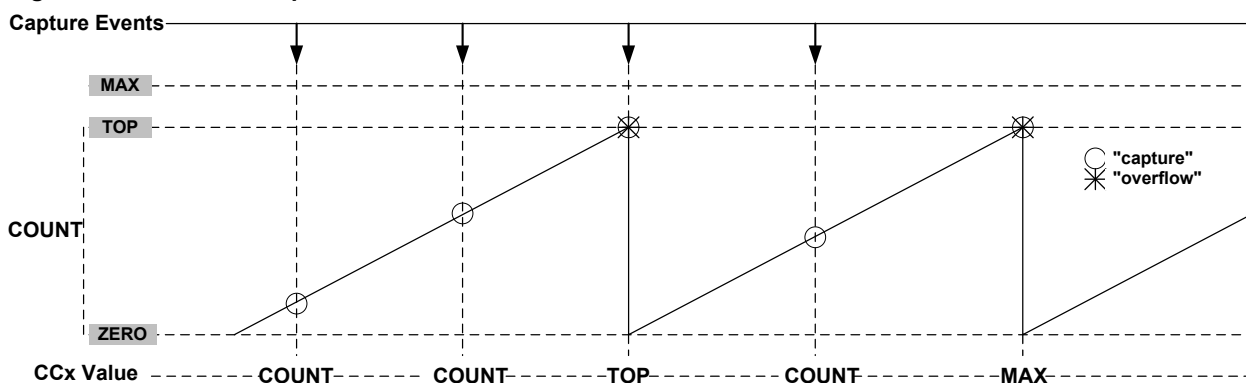
When a capture event is detected, the COUNT value is copied into the corresponding Channel x Compare/Capture Value (CCx) register. In case of an overflow, the MAX value is copied into the corresponding CCx register.

When a valid captured value is present in the capture channel register, the corresponding Capture Channel x Interrupt Flag (INTFLAG.MCx) is set.

The timer/counter can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Channel interrupt flag (INTFLAG.MCx) is still set, the new time-stamp will not be stored and INTFLAG.ERR will be set.



Figure 36-14. Time-Stamp



#### 36.6.4. DMA Operation

The TC can generate the following DMA requests:

- Overflow (OVF): the request is set when an update condition (overflow, underflow or re-trigger) is detected, the request is cleared by hardware on DMA acknowledge.
- Match or Capture Channel x (MCx): for a compare channel, the request is set on each compare match detection, the request is cleared by hardware on DMA acknowledge. For a capture channel, the request is set when valid data is present in the CCx register, and cleared when CCx register is read.

#### 36.6.5. Interrupts

The TC has the following interrupt sources:

- Overflow/Underflow (OVF)
- Match or Capture Channel x (MCx)
- Capture Overflow Error (ERR)

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. See [INTFLAG](#) for details on how to clear interrupt flags.

The TC has one common interrupt request line for all the interrupt sources. The user must read the INTFLAG register to determine which interrupt condition is present.

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

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

#### 36.6.6. Events

The TC can generate the following output events:

- Overflow/Underflow (OVF)
- Match or Capture Channel x (MCx)

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):

- Disable event action (OFF)
- Start TC (START)
- Re-trigger TC (RETRIGGER)
- Count on event (COUNT)
- Capture time stamp (STAMP)
- Capture Period (PPW and PWP)
- Capture Pulse Width (PW)

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](#) on page 474

### 36.6.7. 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 '1'. This peripheral can wake up the device from any sleep mode using interrupts or perform actions through the Event System.

If the On Demand bit in the Control A register (CTRLA.ONDEMAND) is written to '1', the module stops requesting its peripheral clock when the STOP bit in STATUS register (STATUS.STOP) is set to '1'. When a re-trigger or start condition is detected, the TC requests the clock before the operation starts.

### 36.6.8. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

The following bits are synchronized when written:

- Software Reset and Enable bits in Control A register (CTRLA.SWRST and CTRLA.ENABLE)
- Capture Channel Buffer Valid bit in STATUS register (STATUS.CCBUFVx)

The following registers are synchronized when written:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Count Value register (COUNT)
- Period Value register (PER)
- Channel x Compare/Capture Value registers (CCx)

The following registers are synchronized when read:

- Count Value register (COUNT): synchronization is done on demand through READSYNC command (CTRLBSET.CMD).

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.

## 36.7. Register Summary

**Table 36-4. Register Summary – 8-bit Mode**

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
0x01		15:8					ALOCK	PRESCALER[2:0]		
0x02		23:16			COPEN1	COPEN0			CAPTEN1	CAPTEN0
0x03		31:24								
0x04	CTRLBCLR	7:0	CMD[2:0]				ONESHOT	LUPD	DIR	
0x05	CTRLBSET	7:0	CMD[2:0]				ONESHOT	LUPD	DIR	
0x06	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
0x07		15:8			MCEO1	MCEO0			OVFEO	
0x08	INTENCLR	7:0			MC1	MC0		ERR	OVF	
0x09	INTENSET	7:0			MC1	MC0		ERR	OVF	
0x0A	INTFLAG	7:0			MC1	MC0		ERR	OVF	
0x0B	STATUS	7:0			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
0x0C	WAVE	7:0						WAVEGEN[1:0]		
0x0D	DRVCTRL	7:0						INVEN1	INVEN0	
0x0E	Reserved									
0x0F	DBGCTRL	7:0								DBGRUN
0x10	SYNCBUSY	7:0	CC1	CC0	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
0x11		15:8								
0x12		23:16								
0x13		31:24								
0x14	COUNT	7:0	COUNT[7:0]							
0x15	Reserved									
0x16	Reserved									
0x17	Reserved									
0x18	Reserved									
0x19	Reserved									
0x1A	Reserved									
0x1B	PER	7:0	PER[7:0]							
0x1C	CC0	7:0	CC[7:0]							
0x1D	CC1	7:0	CC[7:0]							
0x1E	Reserved									
0x1F	Reserved									
0x20	Reserved									
0x21	Reserved									
0x22	Reserved									
0x23	Reserved									
0x24	Reserved									
0x25	Reserved									

Offset	Name	Bit Pos.							
0x26	Reserved								
0x27	Reserved								
0x28	Reserved								
0x29	Reserved								
0x2A	Reserved								
0x2B	Reserved								
0x2C	Reserved								
0x2D	Reserved								
0x2E	Reserved								
0x2F	PERBUF	7:0						PERBUF[7:0]	
0x30	CCBUF0	7:0						CCBUF[7:0]	
0x31	CCBUF1	7:0						CCBUF[7:0]	
0x32	Reserved								
0x33	Reserved								

**Table 36-5. Register Summary – 16-bit Mode**

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]	ENABLE	SWRST	
0x01		15:8					ALOCK	PRESCALER[2:0]		
0x02		23:16			COPEN1	COPEN0		CAPTEN1	CAPTEN0	
0x03		31:24								
0x04	CTRLBCLR	7:0	CMD[2:0]				ONESHOT	LUPD	DIR	
0x05	CTRLBSET	7:0	CMD[2:0]				ONESHOT	LUPD	DIR	
0x06	EVCTRL	7:0			TCEI	TCINV	EVACT[2:0]			
0x07		15:8			MCEO1	MCEO0			OVFEO	
0x08	INTENCLR	7:0			MC1	MC0		ERR	OVF	
0x09	INTENSET	7:0			MC1	MC0		ERR	OVF	
0x0A	INTFLAG	7:0			MC1	MC0		ERR	OVF	
0x0B	STATUS	7:0			CCBUFV1	CCBUFV0		SLAVE	STOP	
0x0C	WAVE							WAVEGEN[1:0]		
0x0D	DRVCTRL	7:0						INVEN1	INVEN0	
0x0E	Reserved									
0x0F	DBGCTRL	7:0							DBGRUN	
0x10	SYNDBUSY	7:0	CC1	CC0		COUNT	STATUS	CTRLB	ENABLE	SWRST
0x11		15:8								
0x12		23:16								
0x13		31:24								
0x14	COUNT	7:0	COUNT[7:0]							
0x15		15:8	COUNT[15:8]							
0x16	Reserved									
0x17	Reserved									
0x18	Reserved									
0x19	Reserved									
0x1A	Reserved									
0x1B	Reserved									
0x1C	CC0	7:0	CC[7:0]							
0x1D		15:8	CC[15:8]							

Offset	Name	Bit Pos.							
0x1E	CC1	7:0	CC[7:0]						
0x1F		15:8	CC[5:8]						
0x20	Reserved								
0x21	Reserved								
0x22	Reserved								
0x23	Reserved								
0x24	Reserved								
0x25	Reserved								
0x26	Reserved								
0x27	Reserved								
0x28	Reserved								
0x29	Reserved								
0x2A	Reserved								
0x2B	Reserved								
0x2C	Reserved								
0x2D	Reserved								
0x2E	Reserved								
0x2F	Reserved								
0x30	CCBUF0	7:0	CCBUF[7:0]						
0x31		15:8	CCBUF[15:8]						
0x32	CCBUF1	7:0	CCBUF[7:0]						
0x33		15:8	CCBUF[5:8]						
0x34	Reserved								
0x35	Reserved								
0x36	Reserved								
0x37	Reserved								

**Table 36-6. Register Summary – 32-bit Mode**

Offset	Name	Bit Pos.							
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]	ENABLE	SWRST
0x01		15:8					ALOCK	PRESCALER[2:0]	
0x02		23:16			COPEN1	COPEN0		CAPTEN1	CAPTEN0
0x03		31:24							
0x04	CTRLBCLR	7:0	CMD[2:0]				ONESHOT	LUPD	DIR
0x05	CTRLBSET	7:0	CMD[2:0]				ONESHOT	LUPD	DIR
0x06	EVCTRL	7:0		TCEI	TCINV	EVACT[2:0]			
0x07		15:8		MCEO1	MCEO0				OVFEO
0x08	INTENCLR	7:0		MC1	MC0		ERR	OVF	
0x09	INTENSET	7:0		MC1	MC0		ERR	OVF	
0x0A	INTFLAG	7:0		MC1	MC0		ERR	OVF	
0x0B	STATUS	7:0		CCBUFV1	CCBUFV0		SLAVE	STOP	
0x0C	WAVE	7:0					WAVEGEN[1:0]		
0x0D	DRVCTRL	7:0					INVEN1	INVEN0	
0x0E	Reserved								
0x0F	DBGCTRL	7:0						DBGRUN	

Offset	Name	Bit Pos.								
0x10	SYNCBUSY	7:0	CC1	CC0		COUNT	STATUS	CTRLB	ENABLE	SWRST
0x11		15:8								
0x12		23:16								
0x13		31:24								
0x14	COUNT	7:0	COUNT[7:0]							
0x15		15:8	COUNT[15:8]							
0x16		23:16	COUNT[23:16]							
0x17		31:24	COUNT[31:24]							
0x18	Reserved									
0x19	Reserved									
0x1A	Reserved									
0x1B	Reserved									
0x1C	CC0	7:0	CC[7:0]							
0x1D		15:8	CC[15:8]							
0x1E		23:16	CC[23:16]							
0x1F		31:24	CC[31:24]							
0x20	CC1	7:0	CC[7:0]							
0x21		15:8	CC[15:8]							
0x22		23:16	CC[23:16]							
0x23		31:24	CC[31:24]							
0x24	Reserved									
0x25	Reserved									
0x26	Reserved									
0x27	Reserved									
0x28	Reserved									
0x29	Reserved									
0x2A	Reserved									
0x2B	Reserved									
0x2C	Reserved									
0x2D	Reserved									
0x2E	Reserved									
0x2F	Reserved									
0x30	CCBUF0	7:0	CCBUF[7:0]							
0x31		15:8	CCBUF[15:8]							
0x32		23:16	CCBUF[23:16]							
0x33		31:24	CCBUF[31:24]							
0x34	CCBUF1	7:0	CCBUF[7:0]							
0x35		15:8	CCBUF[15:8]							
0x36		23:16	CCBUF[23:16]							
0x37		31:24	CCBUF[31:24]							
0x38	Reserved									
0x39	Reserved									
0x3A	Reserved									
0x3B	Reserved									
0x3C	Reserved									
0x3D	Reserved									

Offset	Name	Bit Pos.								
0x3E	Reserved									
0x3F	Reserved									

### 36.8. 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.

### 36.8.1. Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access			COPEN1	COPEN0			CAPTEN1	CAPTEN0
Reset			0	0			0	0
Bit	15	14	13	12	11	10	9	8
Access					ALOCK	PRESCALER[2:0]		
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
Reset	0	0	0	0	0	0	0	0

#### Bit 11 – ALOCK: Auto Lock

When this bit is set, Lock bit update (LUPD) is set to '1' on each overflow/underflow or re-trigger event.

This bit is not synchronized.

Value	Description
0	The LUPD bit is not affected on overflow/underflow, and re-trigger event.
1	The LUPD bit is set on each overflow/underflow or re-trigger event.

#### 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



Value	Name	Description
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

#### Bit 7 – ONDEMAND: Clock On Demand

This bit selects the clock requirements when the TC is stopped.

In standby mode, if the Run in Standby bit (CTRLA.RUNSTDBY) is '0', ONDEMAND is forced to '0'.

This bit is not synchronized.

Value	Description
0	The On Demand is disabled. If On Demand is disabled, the TC will continue to request the clock when its operation is stopped (STATUS.STOP=1).
1	The On Demand is enabled. When On Demand is enabled, the stopped TC will not request the clock. The clock is requested when a software re-trigger command is applied or when an event with start/re-trigger action is detected.

#### Bit 6 – 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 5:4 – 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

#### 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

Value	Name	Description
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.

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.

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

#### Bits 21,20 – COPENx: Capture On Pin x Enable [x = 1..0]

This bit selects the trigger source for capture operation, either events or I/O pin input.

Value	Description
0	Event from Event System is selected as trigger source for capture operation on channel x.
1	I/O pin is selected as trigger source for capture operation on channel x.

#### Bits 17,16 – CAPTENx: Capture Channel x Enable [x = 1..0]

These bits are used to select whether channel x is a capture or a compare channel.

These bits are not synchronized.

Value	Description
0	CAPTENx disables capture on channel x.
1	CAPTENx enables capture on channel x.

### 36.8.2. Control B Clear

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).

**Name:** CTRLBCLR

**Offset:** 0x04

**Reset:** 0x00

**Property:** PAC Write-Protection, Read-Synchronized, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

#### Bits 7:5 – CMD[2:0]: 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.

#### 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 1 – LUPD: Lock Update

This bit controls the update operation of the TC buffered registers.

When CTRLB.LUPD is set, no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update ensures that all buffer registers are valid before an update is performed.

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 clear the LUPD bit.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on counter update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers.

**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).

### 36.8.3. Control B Set

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).

**Name:** CTRLBSET

**Offset:** 0x05

**Reset:** 0x00

**Property:** PAC Write-Protection, Read-synchronized, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

#### Bits 7:5 – CMD[2:0]: 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 value different from 0x0 to these bits will issue a command for execution.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT

#### 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 enable 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 1 – LUPD: Lock Update

This bit controls the update operation of the TC buffered registers.

When CTRLB.LUPD is set, no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update ensures that all buffer registers are valid before an update is performed.

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

Writing a '1' to this bit will set the LUPD bit.

This bit has no effect when input capture operation is enabled.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on counter update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers.

**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).

### 36.8.4. Event Control

**Name:** EVCTRL  
**Offset:** 0x06  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEO1	MCEO0				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

#### 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

Value	Name	Description
0x3	START	Start TC on event
0x4	STAMP	Time stamp capture
0x5	PPW	Period captured in CC0, pulse width in CC1
0x6	PWP	Period captured in CC1, pulse width in CC0
0x7	PW	Pulse width capture

**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.



### 36.8.5. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x08

**Reset:** 0x00

**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### 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.

#### 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.

### 36.8.6. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x09

**Reset:** 0x00

**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### 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.

#### 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.

### 36.8.7. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** -

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

#### Bit 1 – ERR: Error

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

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.

#### 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.

### 36.8.8. Status

**Name:** STATUS  
**Offset:** 0x0B  
**Reset:** 0x01  
**Property:** Write-Synchronized

Bit	7	6	5	4	3	2	1	0
			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
Access			R/W	R/W	R/W		R	R
Reset			0	0	0		0	1

#### Bit 3 – PERBUFV: Period Buffer Valid

This bit is set when a new value is written to the PERBUF register. The bit is cleared by writing '1' to the corresponding location, or automatically cleared by hardware on UPDATE condition. This bit is available only in 8-bit mode and will always read zero in 16- and 32-bit modes.

#### Bit 1 – 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 0 – 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.

#### Bits 5,4 – CCBUFVx: Channel x Compare or Capture Buffer Valid [x = 1..0]

For a compare channel x, the bit x is set when a new value is written to the corresponding CCBUFx register.

The bit x is cleared by writing a '1' to it, or it is cleared automatically by hardware on UPDATE condition.

For a capture channel x, the bit x is set when a valid capture value is stored in the CCBUFx register. The bit x is cleared automatically when the CCx register is read.

### 36.8.9. Waveform Generation Control

**Name:** WAVE  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected



#### Bits 1:0 – WAVEGEN[1:0]: Waveform Generation Mode

These bits select the waveform generation operation. They affect the top value, as shown in [Waveform Output Operations](#) on page 765. They also control whether frequency or PWM waveform generation should be used. The waveform generation operations are explained in [Waveform Output Operations](#) on page 765.

These bits are not synchronized.

Value	Name	Operation	Top Value	Output Waveform on Match	Output Waveform 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	Set	Clear
0x3	MPWM	Match PWM	CC0	Set	Clear

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 respective MAX value.

### 36.8.10. Driver Control

**Name:** DRVCTRL  
**Offset:** 0x0D  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
							INVEN1	INVEN0
Access							R/W	R/W
Reset							0	0

#### Bits 1,0 – INVENx: Output Waveform x Invert Enable [x = 1..0]

These bits are used to select inversion of the output or capture trigger input of channel x.

Value	Description
0	Disable inversion of the WO[x] output and IO input pin.
1	Enable inversion of the WO[x] output and IO input pin.

### 36.8.11. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x0F  
**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 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.

### 36.8.12. Synchronization Busy

**Name:** SYNCBUSY  
**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								
Reset								
Bit	15	14	13	12	11	10	9	8
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
	CC1	CC0	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST

#### Bit 5 – PER: PER Synchronization Busy

This bit is cleared when the synchronization of PER between the clock domains is complete.

This bit is set when the synchronization of PER between clock domains is started.

This bit is also set when the PER is written, and cleared on update condition. The bit is automatically cleared when the STATUS.PERBUF bit is cleared.

#### Bit 4 – COUNT: COUNT Synchronization Busy

This bit is cleared when the synchronization of COUNT between the clock domains is complete.

This bit is set when the synchronization of COUNT between clock domains is started.

#### Bit 3 – STATUS: STATUS Synchronization Busy

This bit is cleared when the synchronization of STATUS between the clock domains is complete.

This bit is set when a '1' is written to the Capture Channel Buffer Valid status flags (STATUS.CCBUFVx) and the synchronization of STATUS between clock domains is started.

#### Bit 2 – CTRLB: CTRLB Synchronization Busy

This bit is cleared when the synchronization of CTRLB between the clock domains is complete.

This bit is set when the synchronization of CTRLB 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.

**Bits 7,6 – CCx: Compare/Capture Channel x Synchronization Busy**

For details on CC channels number, refer to each TC feature list.

This bit is set when the synchronization of CCx between clock domains is started.

This bit is also set when the CCBUFx is written, and cleared on update condition. The bit is automatically cleared when the STATUS.CCBUFx bit is cleared.

### 36.8.13. Counter Value, 8-bit Mode

**Name:** COUNT

**Offset:** 0x14

**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.

### 36.8.14. Counter Value, 16-bit Mode

**Name:** COUNT

**Offset:** 0x14

**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.

### 36.8.15. Counter Value, 32-bit Mode

**Name:** COUNT

**Offset:** 0x14

**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.

### 36.8.16. Period Value, 8-bit Mode

**Name:** PER  
**Offset:** 0x1B  
**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.

### 36.8.17. Channel x Compare/Capture Value, 8-bit Mode

**Name:** CCx  
**Offset:** 0x1C+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 (WAVE.WAVEGEN), the CC0 register is used as a period register.

### 36.8.18. Channel x Compare/Capture Value, 16-bit Mode

**Name:** CCx  
**Offset:** 0x1C+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 (WAVE.WAVEGEN), the CC0 register is used as a period register.

### 36.8.19. Channel x Compare/Capture Value, 32-bit Mode

**Name:** CCx  
**Offset:** 0x1C+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 (WAVE.WAVEGEN), the CC0 register is used as a period register.



### 36.8.20. Period Buffer Value, 8-bit Mode

**Name:** PERBUF  
**Offset:** 0x2F  
**Reset:** 0xFF  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	PERBUF[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 – PERBUF[7:0]: Period Buffer Value**

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition.

### 36.8.21. Channel x Compare Buffer Value, 8-bit Mode

**Name:** CCBUFx  
**Offset:** 0x30+i\*0x1 [i=0..1]  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	CCBUF[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 – CCBUF[7:0]: Channel x Compare Buffer Value**

These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

### 36.8.22. Channel x Compare Buffer Value, 16-bit Mode

**Name:** CCBUFx  
**Offset:** 0x30+i\*0x2 [i=0..1]  
**Reset:** 0x0000  
**Property:** -

Bit	15	14	13	12	11	10	9	8
CCBUF[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
CCBUF[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 – CCBUF[15:0]: Channel x Compare Buffer Value

These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

### 36.8.23. Channel x Compare Buffer Value, 32-bit Mode

**Name:** CCBUFx  
**Offset:** 0x30+i\*0x4 [i=0..1]  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
	CCBUF[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
	CCBUF[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
	CCBUF[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
	CCBUF[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 – CCBUF[31:0]: Channel x Compare Buffer Value

These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

## 37. TCC – Timer/Counter for Control Applications

### 37.1. 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 intended 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 37-1 [Timer/Counter for Control Applications - Block Diagram](#) on page 810 shows all features in TCC, whereas the table below lists the actual configuration of each of the TCC[2:0].

**Table 37-1. 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.

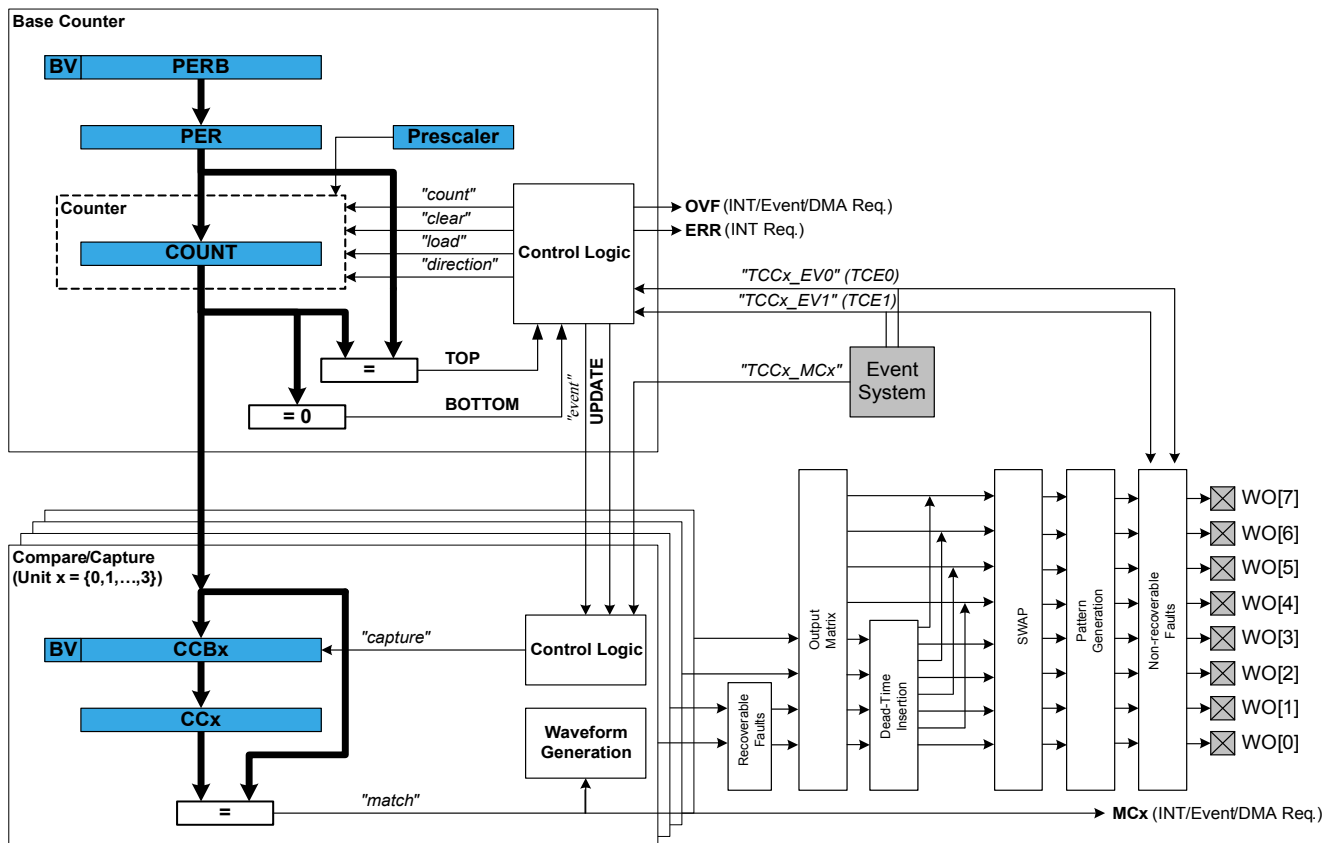
### 37.2. Features

- 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 for counter
  - One input event 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

### 37.3. Block Diagram

Figure 37-1. Timer/Counter for Control Applications - Block Diagram



## 37.4. 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](#) on page 27

## 37.5. Product Dependencies

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

### 37.5.1. 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). Refer to *PORT - I/O Pin Controller* for details.

### Related Links

[PORT: IO Pin Controller](#) on page 444

### 37.5.2. 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. Refer to *PM – Power Manager* for details on the different sleep modes.

### 37.5.3. Clocks

The TCC bus clock (CLK\_TCCx\_APB, with x instance number of the TCCx) is enabled by default, and can be enabled and disabled in the Main Clock.

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](#) on page 843 for further details.

### Related Links

[Peripheral Clock Masking](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

#### 37.5.4. 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](#) on page 334

#### 37.5.5. 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](#) on page 41

#### 37.5.6. Events

The events of this peripheral are connected to the Event System.

##### Related Links

[EVSYS – Event System](#) on page 474

#### 37.5.7. 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](#) on page 864 register for details.

#### 37.5.8. Register Access Protection

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

When the CPU is halted in debug mode, all write-protection is automatically disabled. Write-protection does not apply for accesses through an external debugger.

#### 37.5.9. Analog Connections

Not applicable.



## 37.6. Functional Description

### 37.6.1. Principle of Operation

The following definitions are used throughout the documentation:

**Table 37-2. 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> on page 818.
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 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](#) on page 830.

In addition, six optional independent and successive units primarily intended for use with different types of motor control, ballast, LED, H-bridge, power converter, and other types of power switching applications, are implemented in some of TCC instances. See also [Figure 37-1 Timer/Counter for Control Applications - Block Diagram](#) on page 810.

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 safe and pre-configured 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](#) on page 474

## 37.6.2. Basic Operation

### 37.6.2.1. 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).

### 37.6.2.2. 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 on page 848) register for details.

The TCC should be disabled before the TCC is reset to avoid undefined behavior.

### 37.6.2.3. 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.

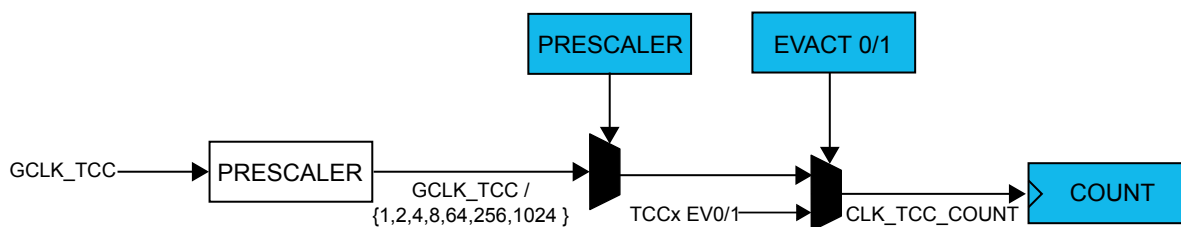
If the prescaler value is higher than one, the counter update condition can be optionally executed on the next GCLK\_TCC clock pulse or the next prescaled clock pulse. For further details, refer to the Prescaler (CTRLA.PRESCALER) and Counter Synchronization (CTRLA.PRESYNC) descriptions.

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 37-2. Prescaler



### 37.6.2.4. Counter Operation

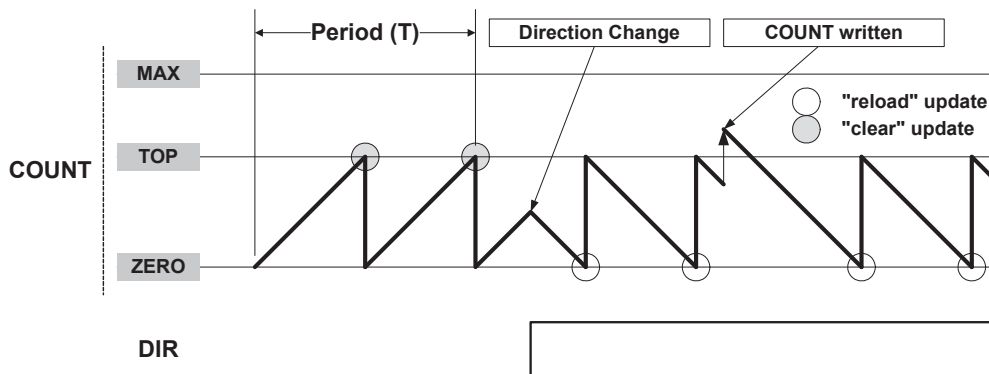
Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each TCC clock input (CLK\_TCC\_COUNT). A counter clear or reload mark the end of current counter cycle and the start of a new one.

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).

**Figure 37-3. 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 37-3 Counter Operation](#) on page 816.

### Stop Command and Event Action

A stop command can be issued from software by using TCC Command bits in Control B Set register (CTRLBSET.CMD=0x2, STOP) or when the stop event action is configured in the Input Event Action 1 bits in Event Control register (EVCTRL.EVACT1=0x3, STOP).

When a stop is detected while the counter is running, the counter will maintain its current value. If the waveform generation (WG) is used, all waveforms are set to a state defined in Non-Recoverable State x Output Enable bit and Non-Recoverable State x Output Value bit in the Driver Control register (DRVCTRL.NREx and DRVCTRL.NRVx), 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 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.

**37.6.2.5. 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](#) on page 822. 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)

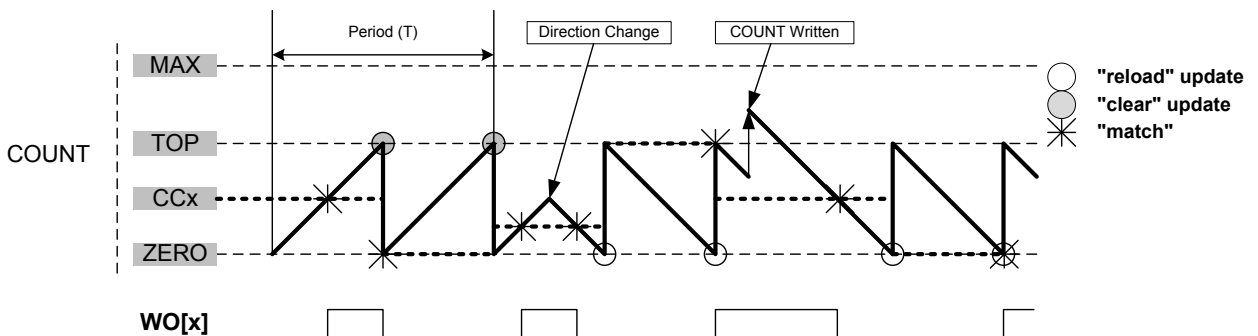
When using MFRQ configuration, the TOP value is defined by the CC0 register value. For the other waveform operations, the TOP value is defined by the Period (PER) register value.

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.

### 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.

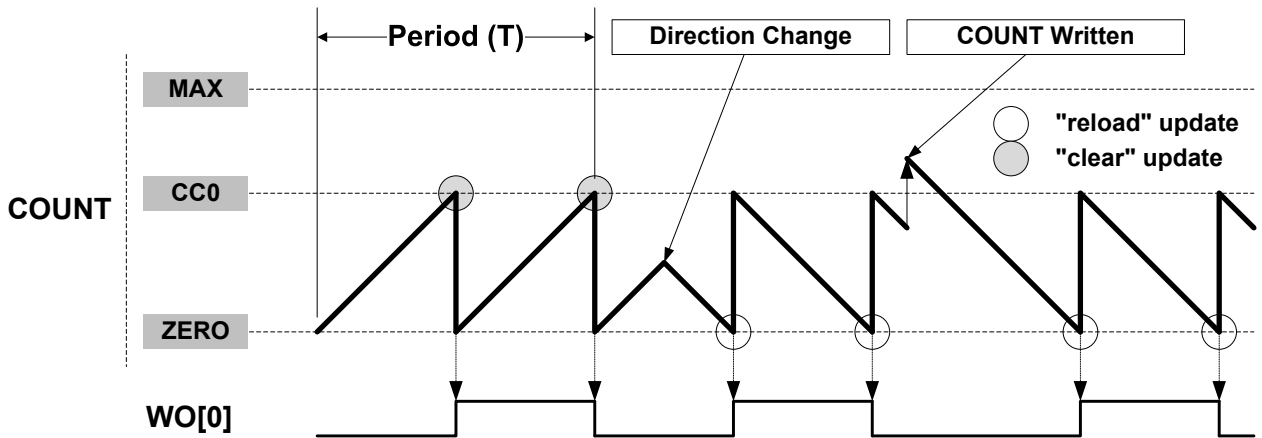
**Figure 37-4. Normal Frequency Operation**



### 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 37-5. Match Frequency Operation



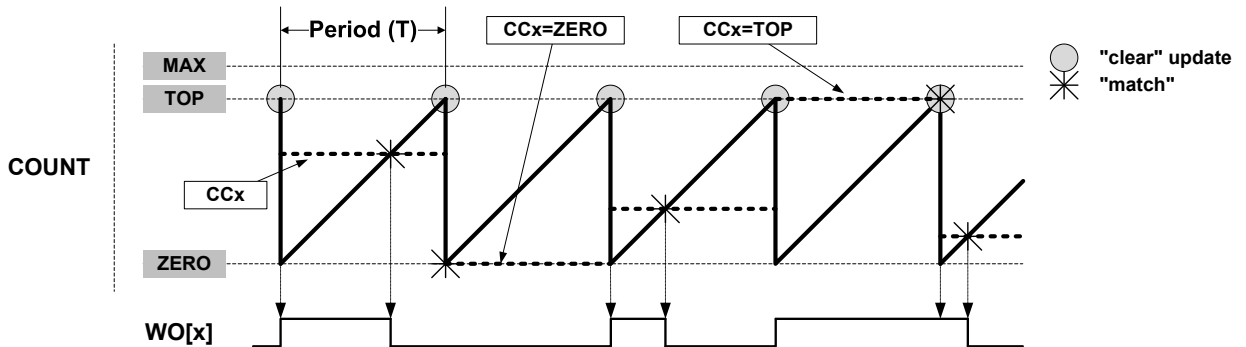
### Normal Pulse-Width Modulation (NPWM)

NPWM uses single-slope PWM generation.

#### Single-Slope PWM Generation

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 37-6. 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).

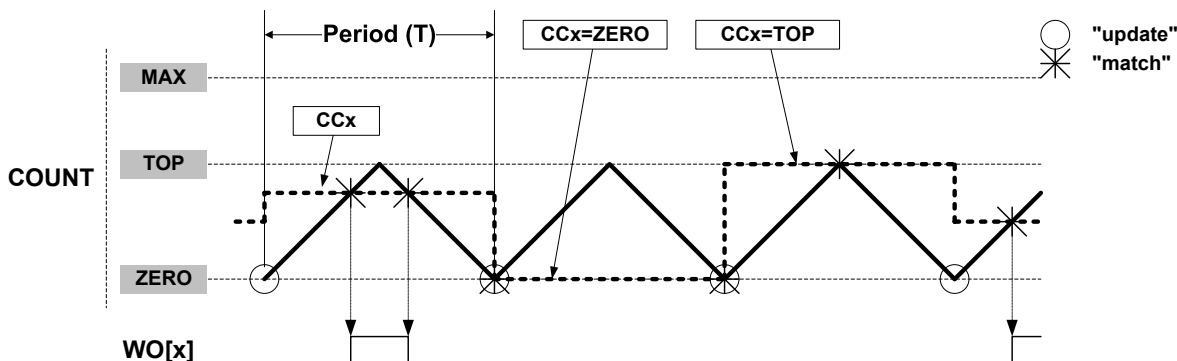
#### Dual-Slope PWM Generation

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/event is generated on TOP and/or ZERO, depend of Dual slope. In DSBOTH operation, a second update time occurs on TOP.

**Figure 37-7. 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 ( $R_{PWM\_DS}$ ):

$$R_{PWM\_DS} = \frac{\log(PER+1)}{\log(2)}$$

The PWM frequency  $f_{PWM\_DS}$  depends on the period setting (TOP) and the peripheral clock frequency  $f_{GCLK\_TCC}$ , and can be calculated by the following equation:

$$f_{PWM\_DS} = \frac{f_{GCLK\_TCC}}{2N \cdot PER}$$

$N$  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 ( $P_{PWM\_DS}$ ) depends on the compare channel (CCx) register value and the peripheral clock frequency ( $f_{GCLK\_TCC}$ ), and can be calculated by the following equation:

$$P_{PWM\_DS} = \frac{2N \cdot (TOP - CCx)}{f_{GCLK\_TCC}}$$

$N$  represents the prescaler divider used.

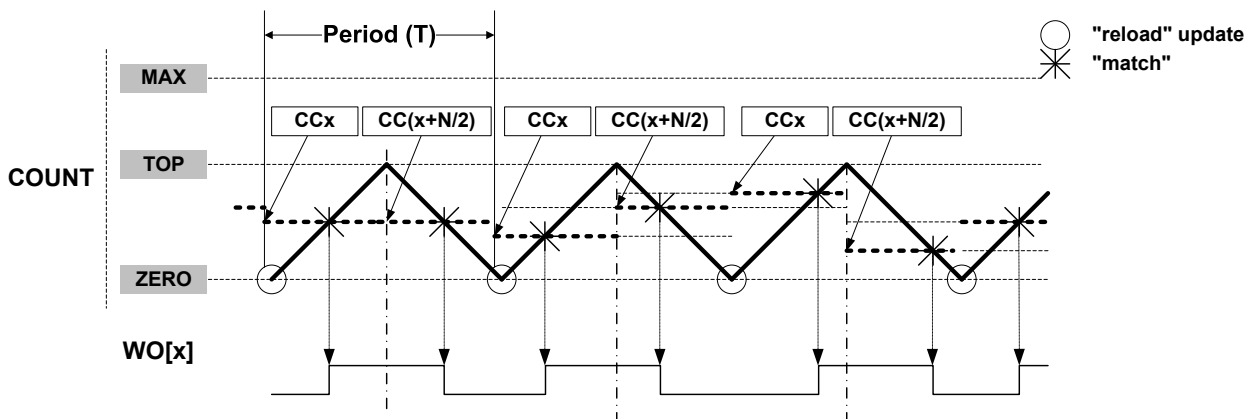
**Note:** In DSTOP, DSBOTTOM and DSBOTH operation, when TOP is lower than MAX/2, the CCx MSB bit defines the ramp on which the CCx Match interrupt or event is generated. (Rising if CCx[MSB]=0, falling if CCx[MSB]=1.)

#### 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 37-8. Dual-Slope Critical Pulse Width Modulation (N=CC\_NUM)



The table below shows the update counter and overflow event/interrupt generation conditions in different operation modes.

Table 37-3. 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 & ZERO			TOP	ZERO
DSTOP	Dual-slope PWM	PER	ZERO			TOP	-

### 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 37-4. 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.

**Related Links**

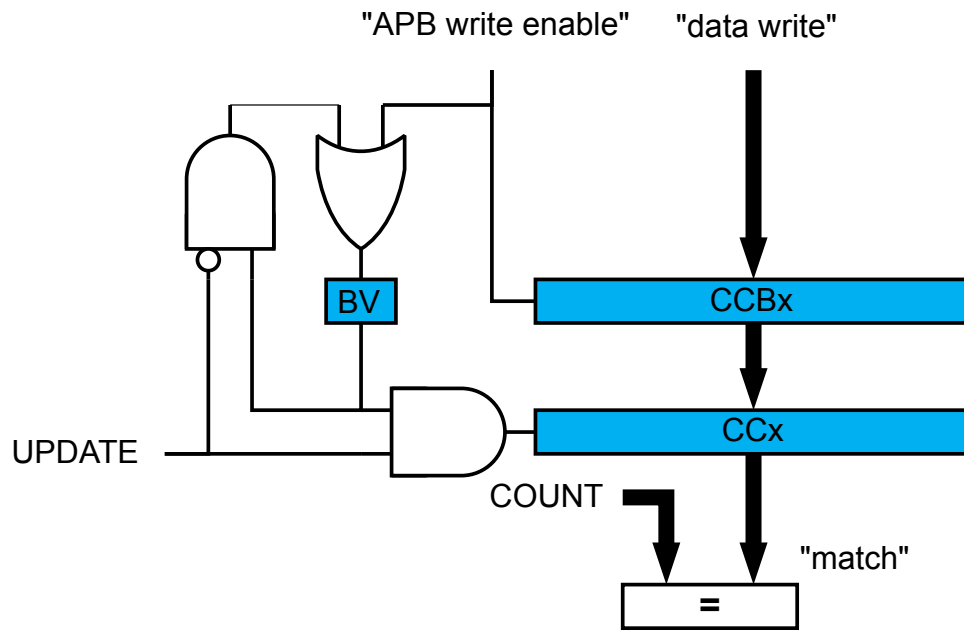
[PORT: IO Pin Controller](#) on page 444

**37.6.2.6. Double Buffering**

The Pattern (PATT), Period (PER), and Compare Channels (CCx) registers are all double buffered. Each buffer register has a buffer valid (PATTBUFV, PERBUFV or CCBUFVx) 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' the and write a '0' to CTRLBCLR.LUPD to enable double buffering, the data from the buffer register will be copied into the corresponding register under UPDATE condition (CTRLBSET.CMD=0x3, UPDATE), including the software update command. Then the buffer valid flag bit in the STATUS register are automatically cleared by hardware or software. Double buffering is not applied for capture mode. A compare register is as the following figure.

Figure 37-9. Compare Channel Double Buffering



Both the registers (PATT/PER/CCx) and corresponding buffer registers (PATTB /PERBUF/CCBUFx) 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. This allows initialization and bypassing of the buffer register and the double buffering feature.

**Note:** In NFRQ, MFRQ or PWM down-counting counter mode (CTRLBSET.DIR=1), the PER register and the PERBUF register are written simultaneously if double buffering is active or as soon as double buffering is activated (CTRLBCLR.LUPD=0).

### Changing the Period

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 double buffering is off (CTRLBSET.LUPD=1), any period update is effective after the synchronization delay.

Figure 37-10. Unbuffered Single-Slope Up-Counting Operation

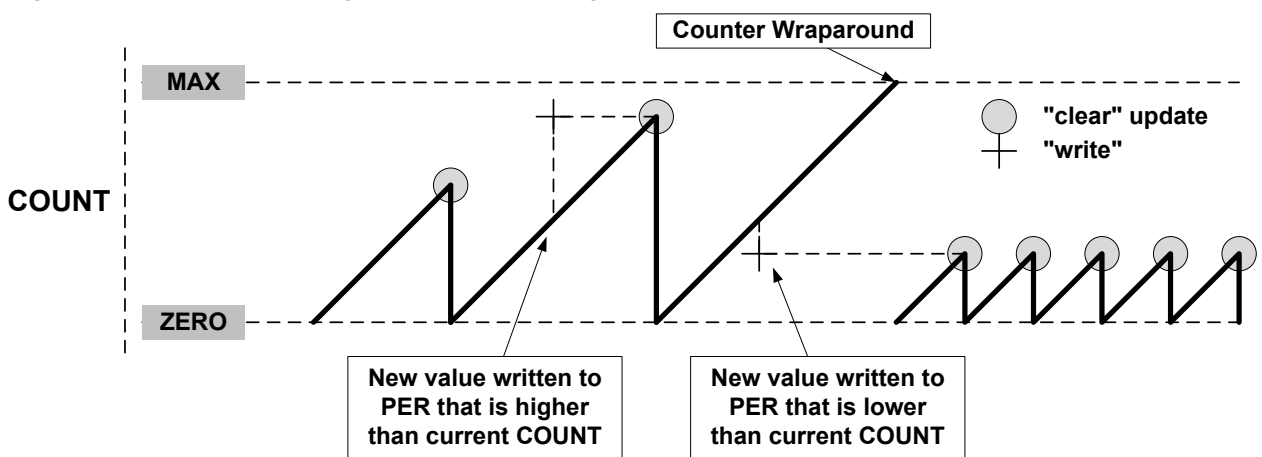
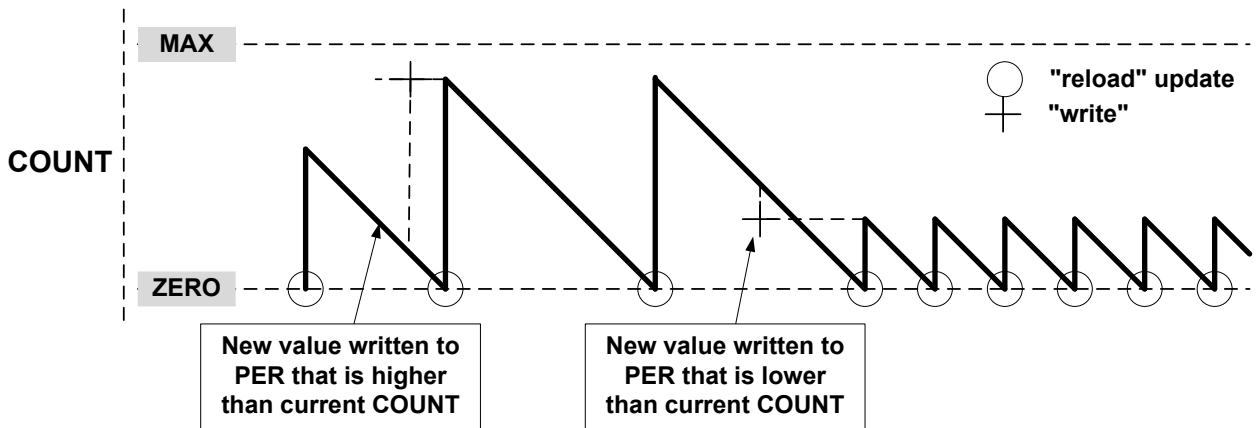
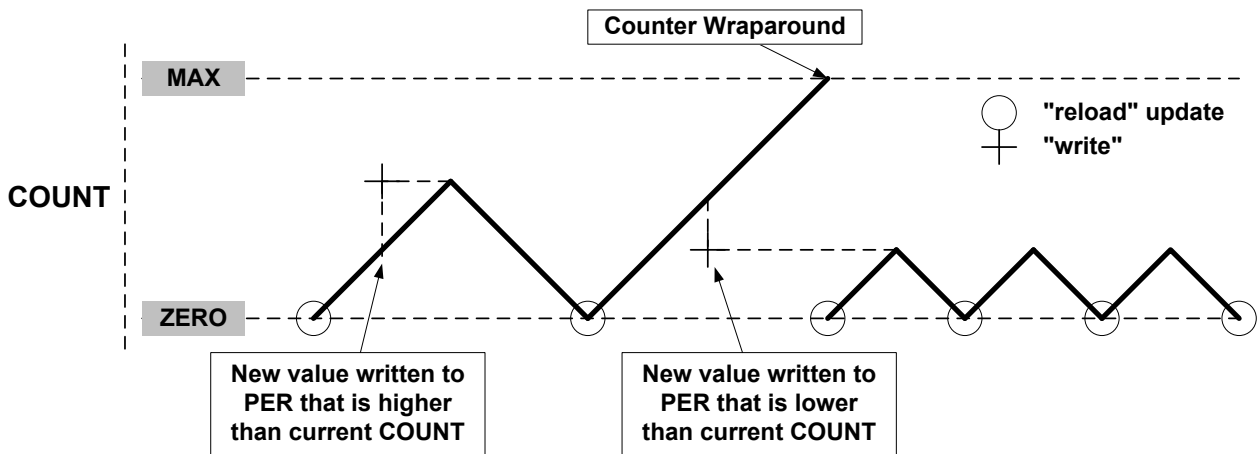


Figure 37-11. Unbuffered Single-Slope Down-Counting Operation



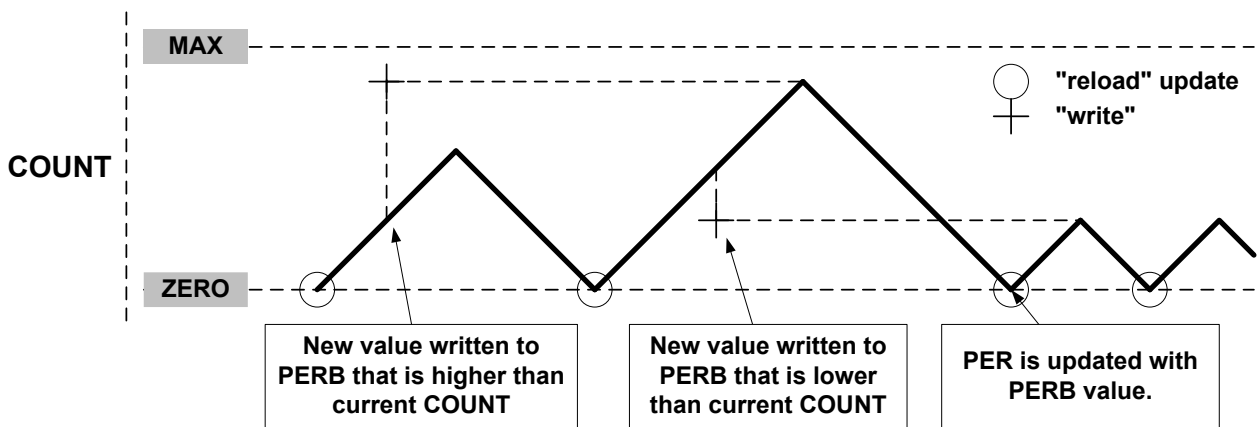
A counter wraparound can occur in any operation mode when up-counting without buffering, see [Figure 37-10 Unbuffered Single-Slope Up-Counting Operation](#) on page 823. COUNT and TOP are continuously compared, so when a new value that is lower than the current COUNT is written to TOP, COUNT will wrap before a compare match.

Figure 37-12. 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 37-13 Changing the Period Using Buffering](#) on page 824. This prevents wraparound and the generation of odd waveforms.

Figure 37-13. Changing the Period Using Buffering



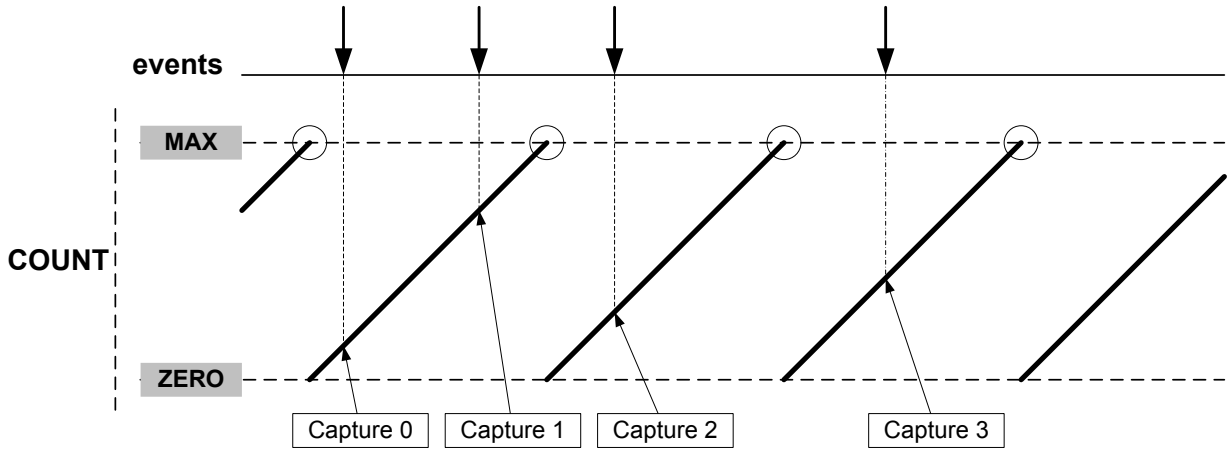
### 37.6.2.7. Capture Operations

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

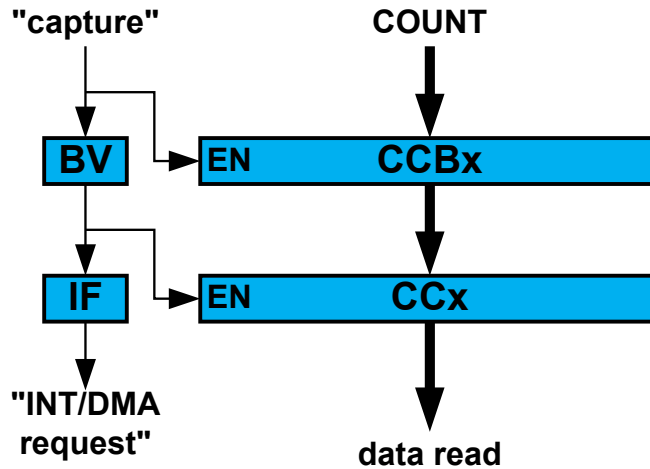
The compare/capture channels can be used as input capture channels to capture events from the Event System, and give them a timestamp. The following figure shows four capture events for one capture channel.

Figure 37-14. 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.

Figure 37-15. Capture Double Buffering



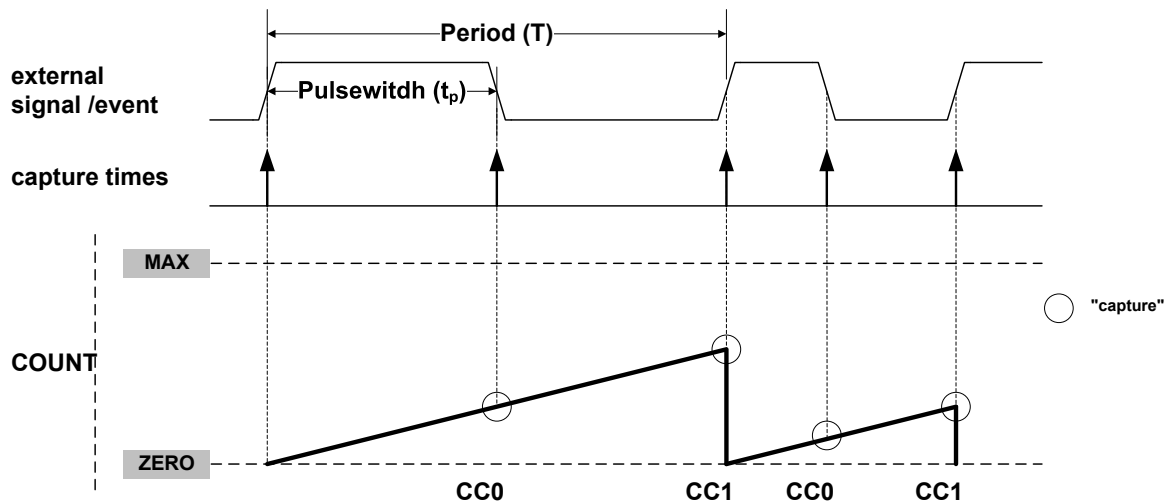
The TCC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Buffer Valid flag (STATUS.CCBV) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

#### 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:

$$f = \frac{1}{T} \quad , \quad \text{dutyCycle} = \frac{t_p}{T}$$

Figure 37-16. 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.TCEINV $x$ ) is used for event source  $x$  to select whether the wraparound should occur on the rising edge or the falling edge. If EVCTRL.TCEINV $x$ =1, the wraparound will happen on the falling edge.

The corresponding capture is done only if the channel is enabled in capture mode (CTRLA.CPTEN $x$ =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.MC $x$  is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

**Note:** In dual-slope PWM operation, and when TOP is lower than MAX/2, the CC $x$  MSB captures the CTRLB.DIR state to identify the ramp on which the capture has been done. For rising ramps CC $x$ [MSB] is zero, for falling ramps CC $x$ [MSB]=1.

### 37.6.3. Additional Features

#### 37.6.3.1. 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.NRE $x$  and DRVCTRL.NRV $x$ .

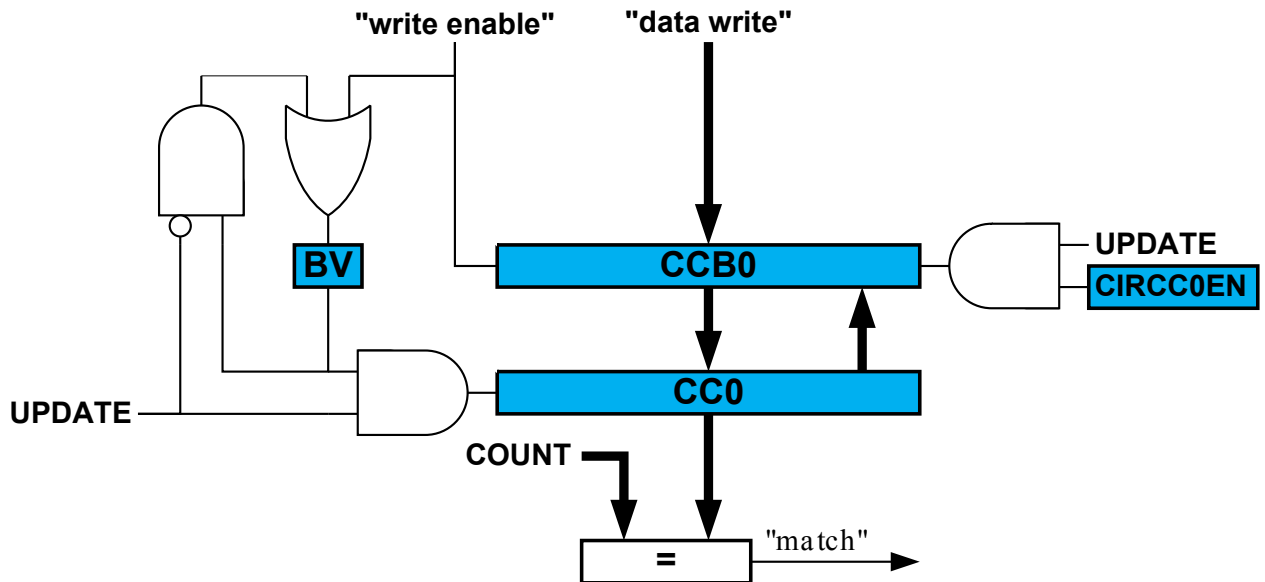
One-shot operation can be enabled by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) and disabled by writing a '1' to CTRLBCLR.ONESHOT. When enabled, the TCC will count until an overflow or underflow occurs and stop counting. The one-shot operation can be

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.

### 37.6.3.2. 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 DSBOTB operations.

Figure 37-17. Circular Buffer on Channel 0



### 37.6.3.3. 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.

#### 37.6.3.4. 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 37-18 RAMP2](#)

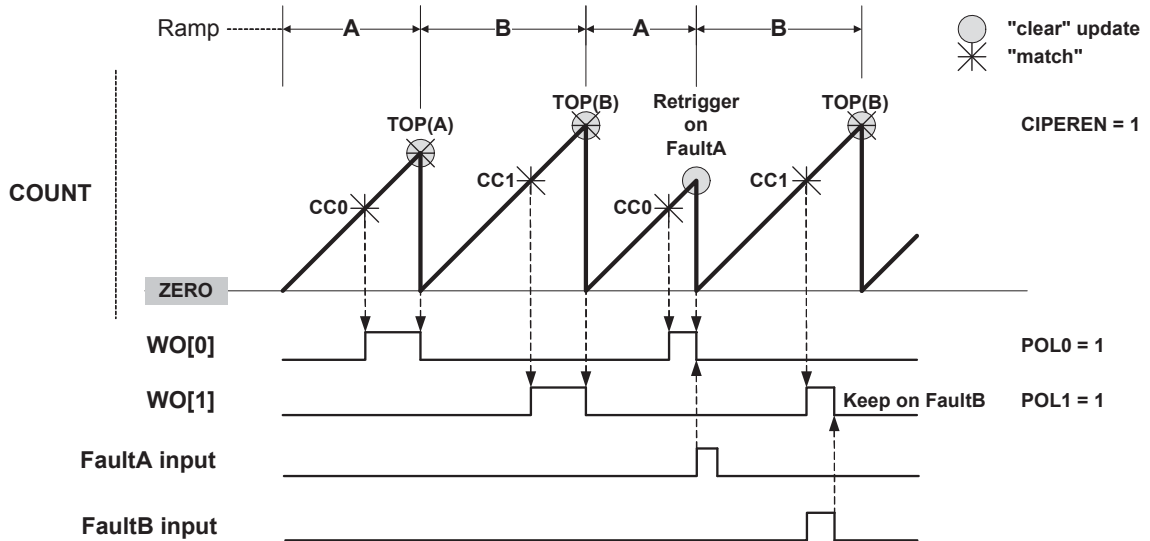


[Standard Operation](#) on page 829. In cycle A, odd channel output is disabled, and in cycle B, even channel output is disabled. The ramp index changes after each update, but can be software modified using the Ramp index command bits in Control B Set register (CTRLBSET.IDXCMD).

### 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.

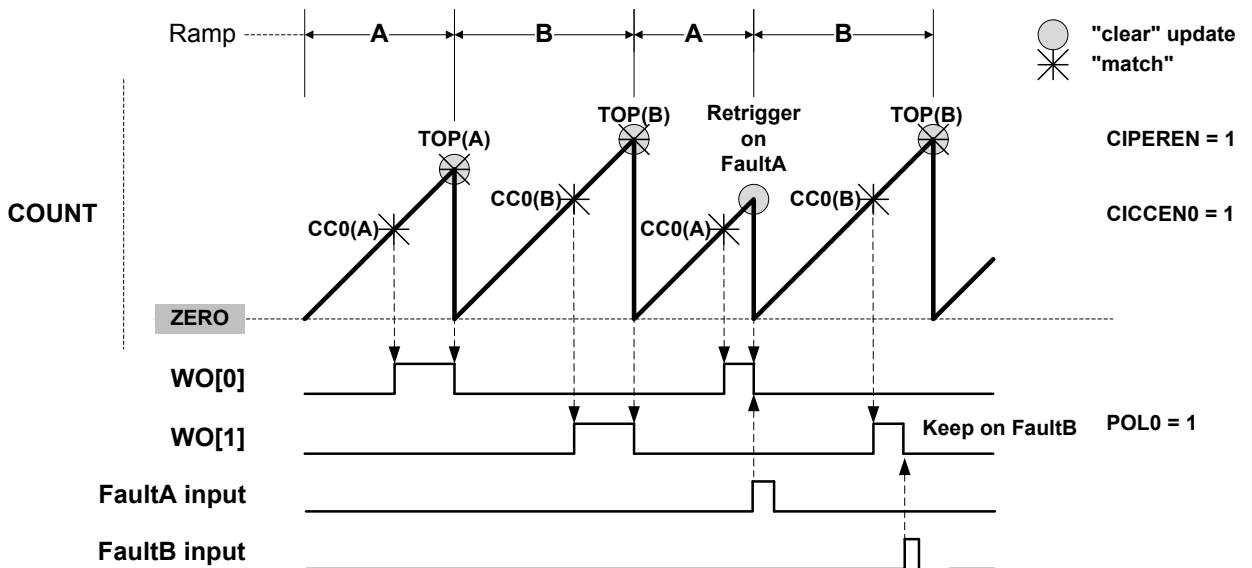
**Figure 37-18. RAMP2 Standard Operation**



### 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 37-19. RAMP2 Alternate Operation**



### Critical RAMP2 (RAMP2C) Operation

Critical RAMP2 operation provides a way to cover RAMP2 operation requirements without the update constraint associated to the use of circular buffers. In this mode, CC0 is controlling the period of ramp A and PER is controlling the period of ramp B. When using more than two channels, WO[0] output is controlled by CC2 (HIGH) and CC0 (LOW). On TCC with 2 channels, a pulse on WO[0] will last the entire period of ramp A, if WAVE.POL0=0.

#### 37.6.3.5. 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

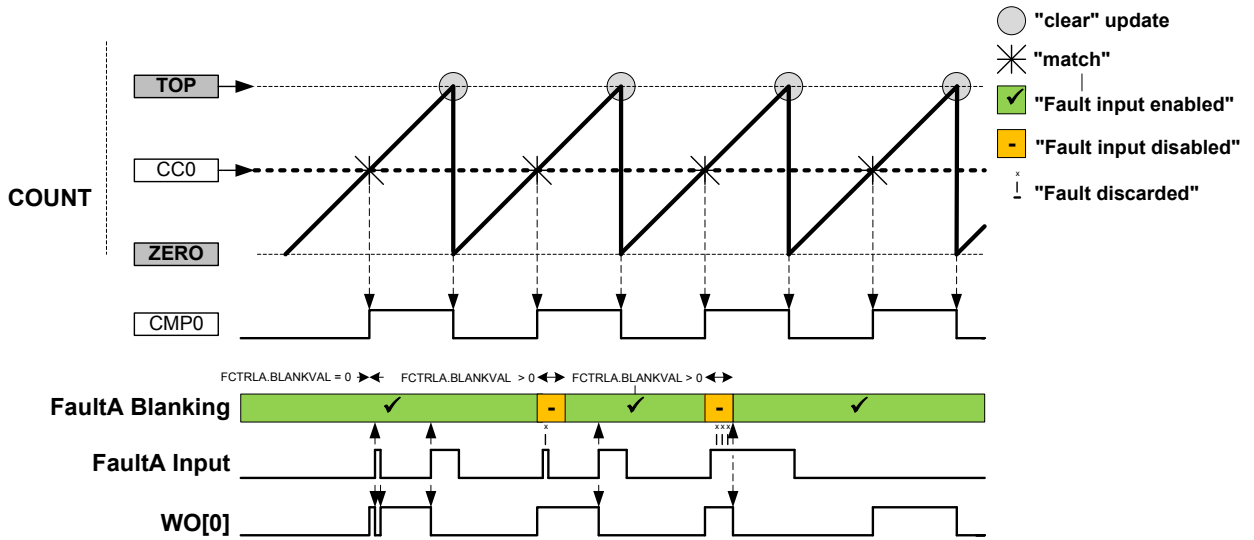
$$t_b = \frac{1 + \text{BLANKVAL}}{f_{\text{GCLK\_TCCx\_PRESC}}}$$

Here,  $f_{\text{GCLK\_TCCx\_PRESC}}$  is the frequency of the prescaled peripheral clock frequency  $f_{\text{GCLK\_TCCx}}$ .

The prescaler is enabled by writing '1' to the Fault n Blanking Prescaler bit (FCTRLn.BLANKPRESC). When disabled,  $f_{\text{GCLK\_TCCx\_PRESC}} = f_{\text{GCLK\_TCCx}}$ . When enabled,  $f_{\text{GCLK\_TCCx\_PRESC}} = f_{\text{GCLK\_TCCx}}/64$ .

The maximum blanking time (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 37-20. 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 (FCTRLn.QUAL). When the recoverable fault qualification is enabled (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 37-21. Fault Qualification in RAMP1 Operation

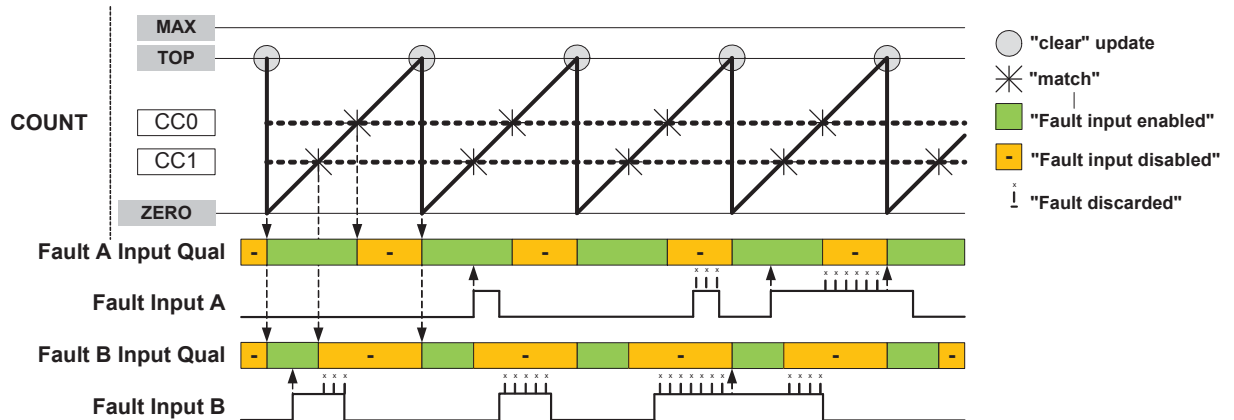
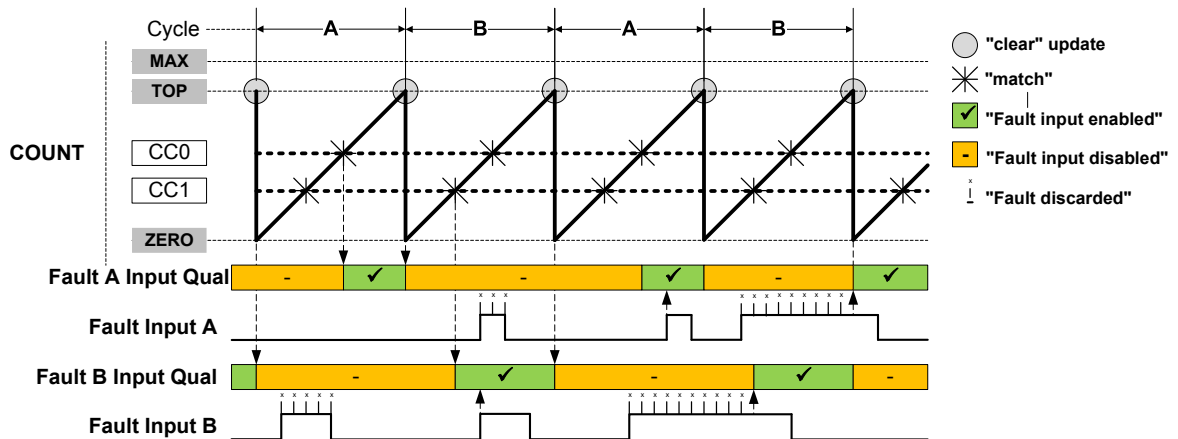


Figure 37-22. Fault Qualification in RAMP2 Operation with Inverted Polarity

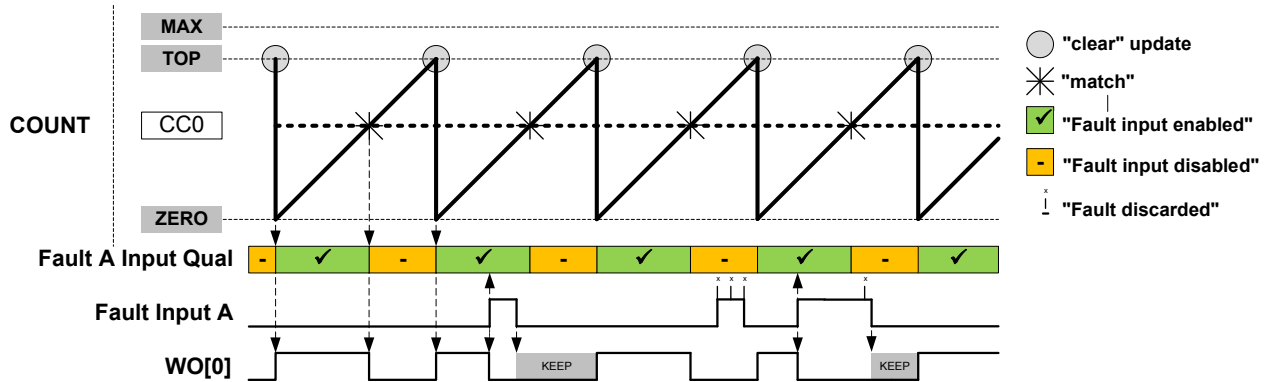


## Fault Actions

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 37-23. 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 37-24 Waveform Generation in RAMP1 mode with Restart Action](#) on page 832. 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 37-25 Waveform Generation in RAMP2 mode with Restart Action](#) on page 833. 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 37-24. Waveform Generation in RAMP1 mode with Restart Action

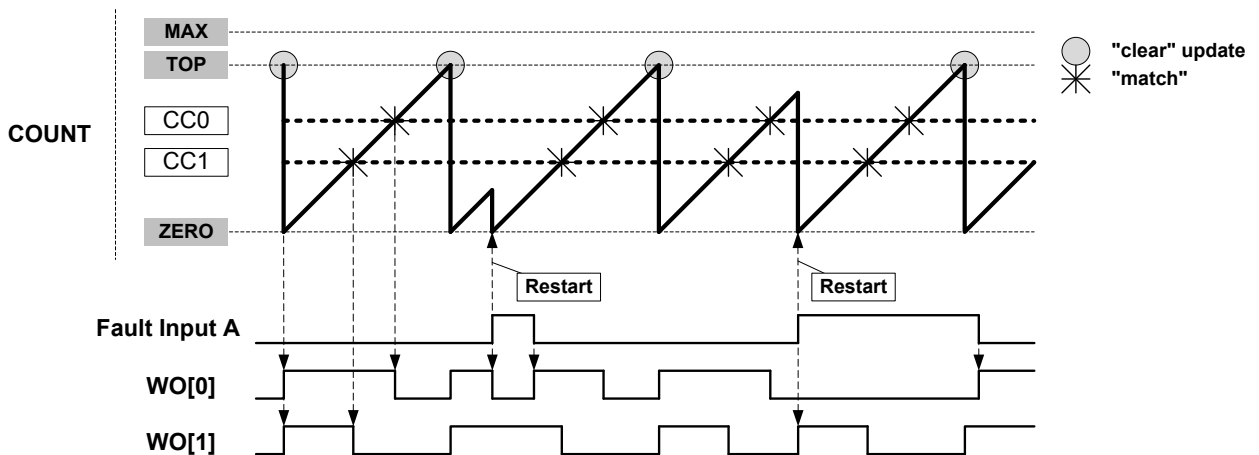
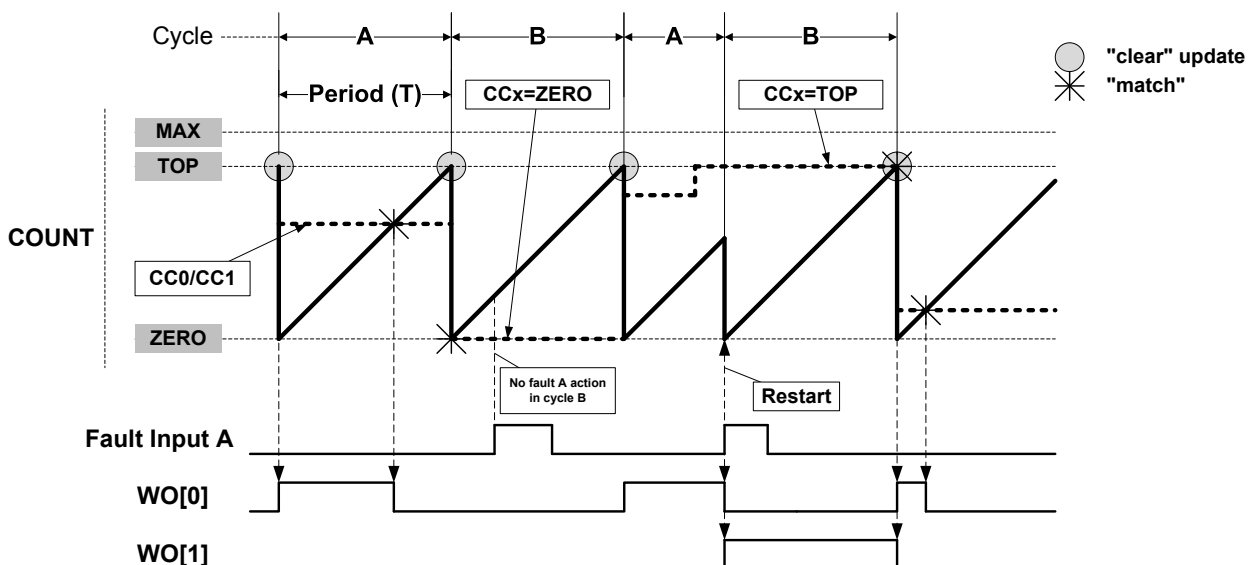


Figure 37-25. Waveform Generation in RAMP2 mode with Restart Action



**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](#) on page 825
- 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 37-26 Capture Action "CAPTMAX"](#) on page 834.
- 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 37-27 Capture Action "DERIV0"](#) on page 834.

**CCx Content:**

In CAPTMIN and CAPTMAX operations, CCx keeps the respective extremum values, see [Figure 37-26 Capture Action "CAPTMAX"](#) on page 834. In LOCMIN, LOCMAX or DERIV0 operation, CCx follows the counter value at fault time, see [Figure 37-27 Capture Action "DERIV0"](#) on page 834.

**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 lower (for LOCMIN) or higher (for LOCMAX) than the previous captured value. DERIV0 is equivalent to an OR function of (LOCMIN, LOCMAX).

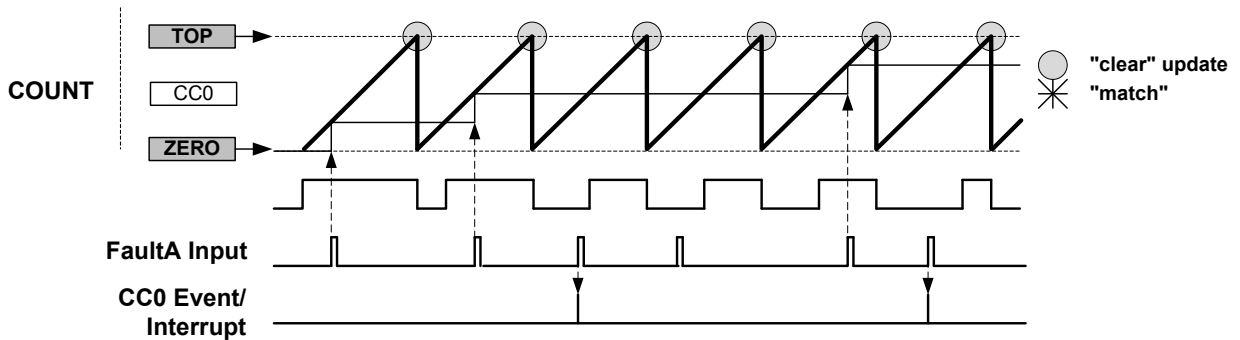
In CAPT operation, capture is performed on each capture event. The MCx interrupt flag is set on each new capture.

In CAPTMIN and CAPTMAX operation, capture is performed only when a new lower (for CAPTMIN) or new higher (for CAPMAX) value is detected. The MCx interrupt flag is set on each new capture.

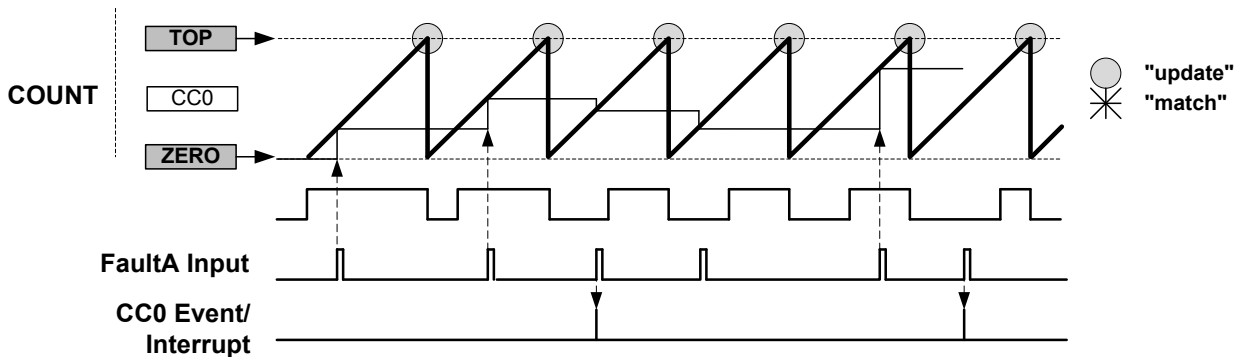
*Interrupt Generation*

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 37-26. Capture Action “CAPTMAX”**



**Figure 37-27. 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 37-28. Waveform Generation with Halt and Restart Actions

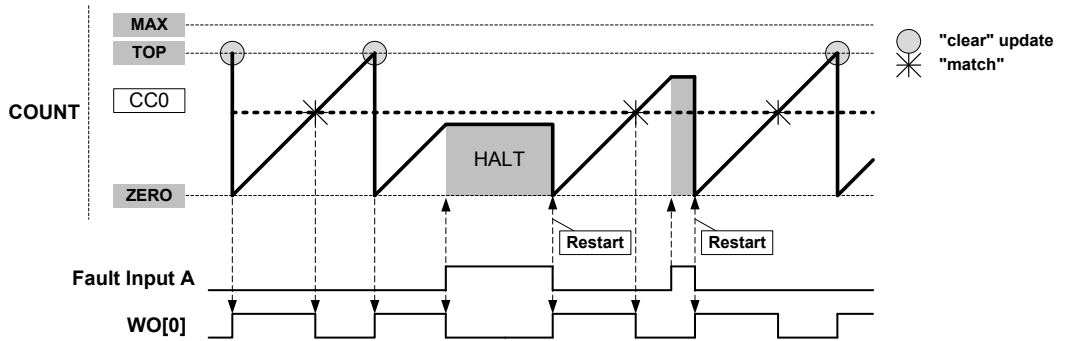
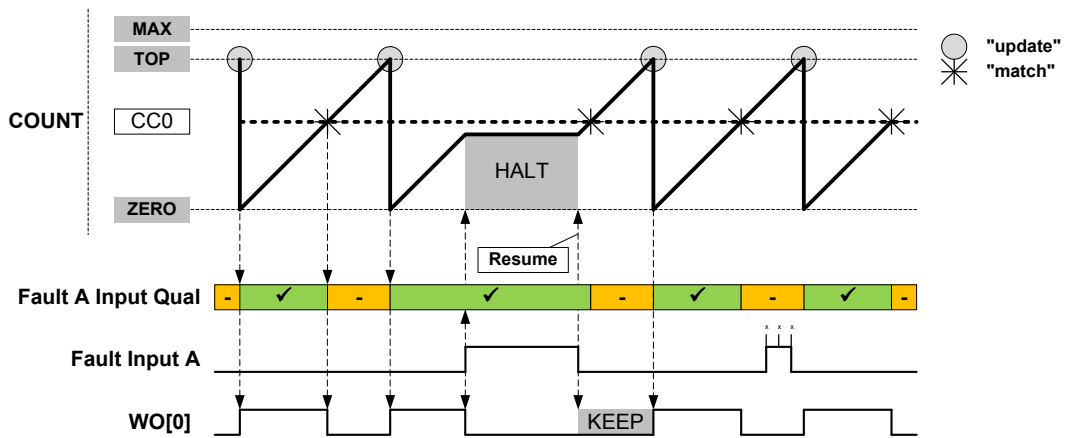


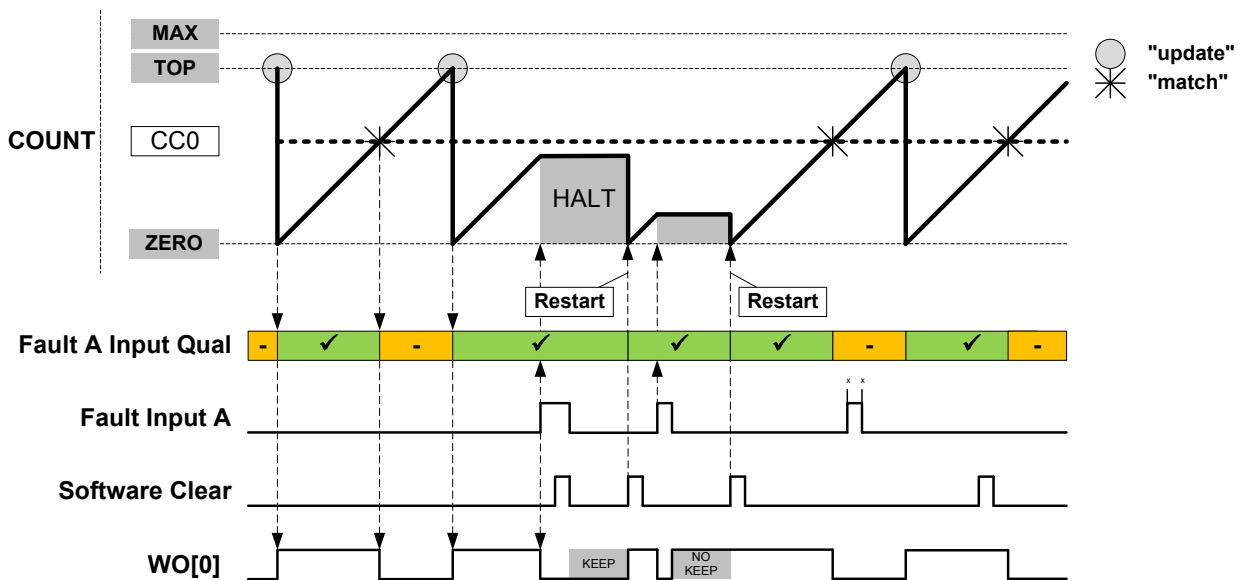
Figure 37-29. Waveform Generation with Fault Qualification, Halt, and Restart Actions



**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 37-30. Waveform Generation with Software Halt, Fault Qualification, Keep and Restart Actions



### 37.6.3.6. 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.

In RAMP2, RAMP2A, or DSBOTH operation, when the Lock Update bit in the Control B register is set by writing CTRLBSET.LUPD=1 and the ramp index or counter direction changes, a non-recoverable Update Fault State and the respective interrupt (UFS) are generated.

### 37.6.3.7. Time-Stamp Capture

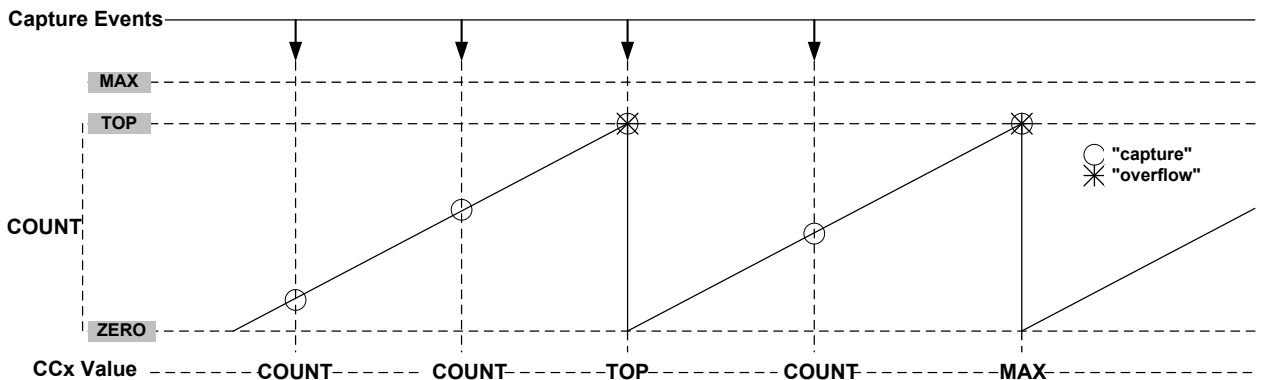
This feature is enabled when the Capture Time Stamp (STAMP) Event Action in Event Control register (EVCTRL.EVACT) is selected. The counter TOP value must be smaller than MAX.

When a capture event is detected, the COUNT value is copied into the corresponding Channel x Compare/Capture Value (CCx) register. In case of an overflow, the MAX value is copied into the corresponding CCx register.

When a valid captured value is present in the capture channel register, the corresponding Capture Channel x Interrupt Flag (INTFLAG.MCx) is set.

The timer/counter can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Channel interrupt flag (INTFLAG.MCx) is still set, the new time-stamp will not be stored and INTFLAG.ERR will be set.

**Figure 37-31. Time-Stamp**



### 37.6.3.8. Waveform Extension

Figure 37-32 [Waveform Extension Stage Details](#) on page 837 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:

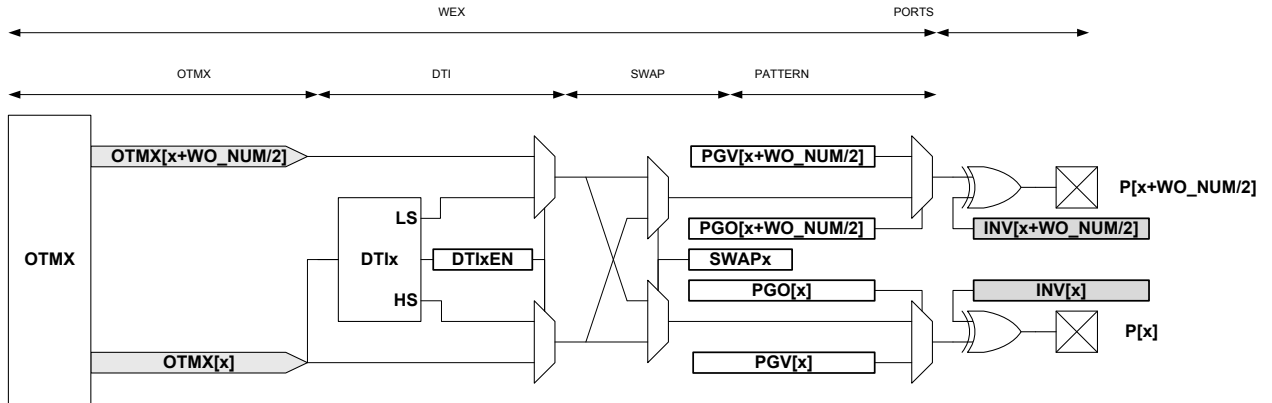
- Slice 0 DTI0 / SWAP0 acting on port pins (WO[0], WO[WO\_NUM/2 +0])
- Slice 1 DTI1 / SWAP1 acting on port pins (WO[1], WO[WO\_NUM/2 +1])

And more generally:



- Slice n DTIx / SWAPx acting on port pins (WO[x], WO[WO\_NUM/2 +x])

**Figure 37-32. Waveform Extension Stage Details**



The output matrix (OTMX) unit distributes compare channels, according to the selectable configurations in [Table 37-5 Output Matrix Channel Pin Routing Configuration](#) on page 837.

**Table 37-5. 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 37-5 Output Matrix Channel Pin Routing Configuration](#) on page 837:

- 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 37-6. Example: four compare channels on four outputs**

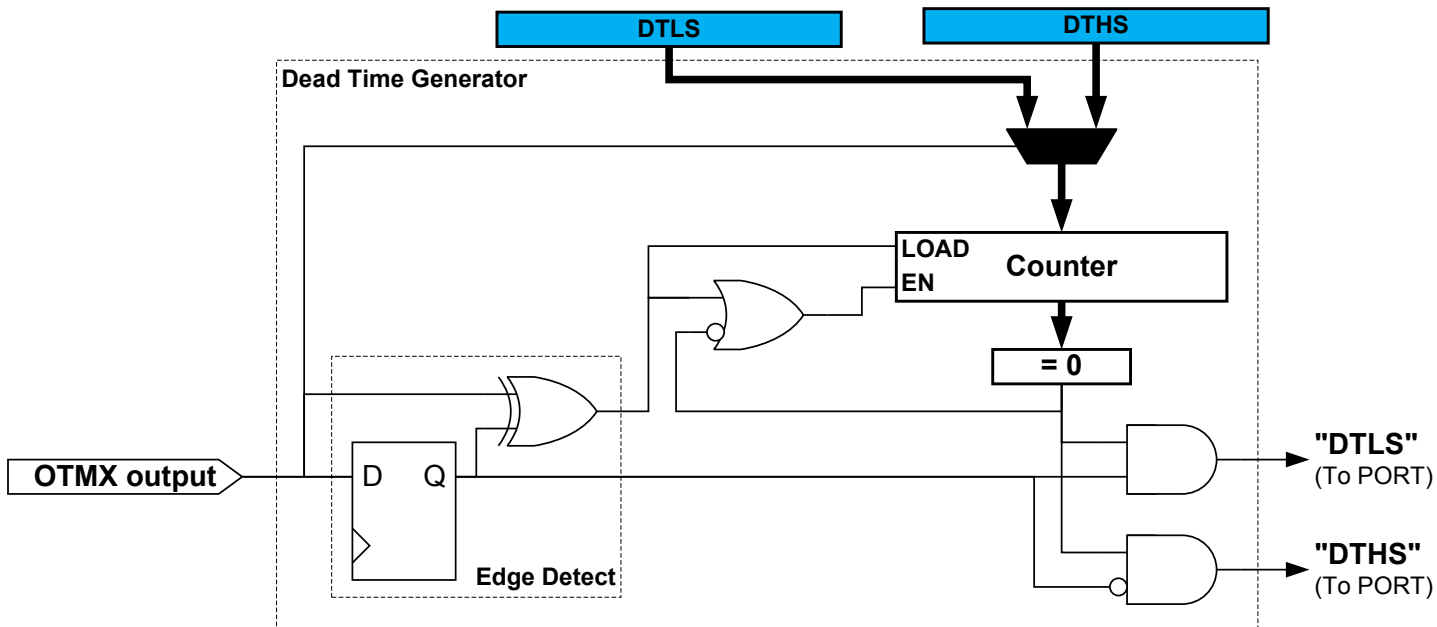
Value	OTMX[3]	OTMX[2]	OTMX[1]	OTMX[0]
0x0	CC3	CC2	CC1	CC0
0x1	CC1	CC0	CC1	CC0

Value	OTMX[3]	OTMX[2]	OTMX[1]	OTMX[0]
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.

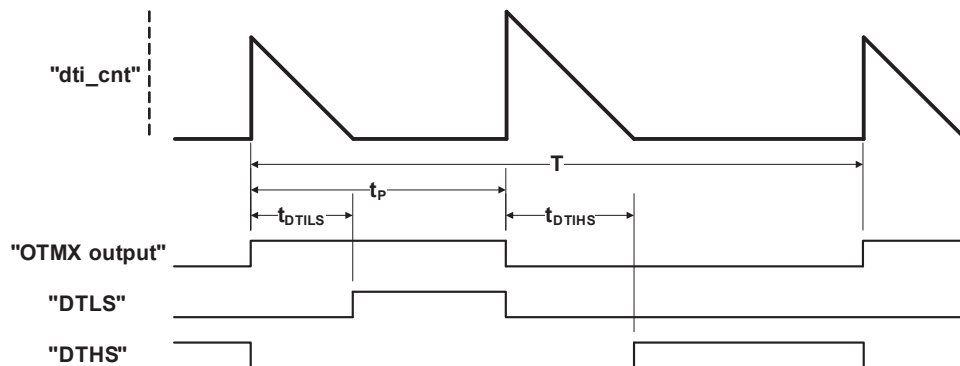
The DTI stage consists of four equal dead-time insertion generators; one for each of the first four compare channels. Figure 37-33 Dead-Time Generator Block Diagram on page 838 shows the block diagram of one DTI generator. The four channels have a common register which controls the dead time, which is independent of high side and low side setting.

Figure 37-33. Dead-Time Generator Block Diagram



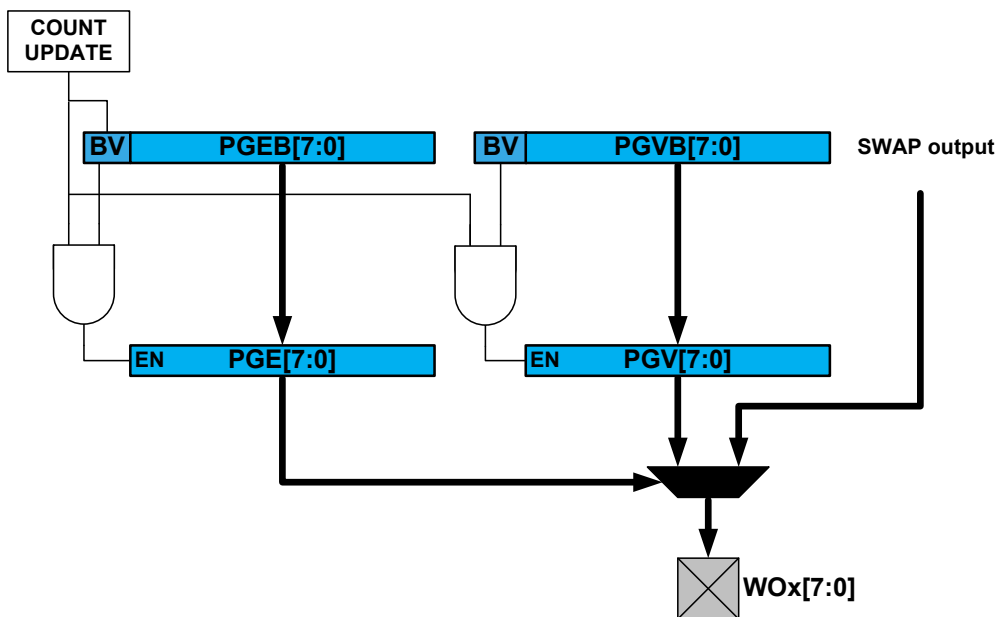
As shown in Figure 37-34 Dead-Time Generator Timing Diagram on page 838, the 8-bit dead-time counter is decremented by one for each peripheral clock cycle until it reaches zero. A non-zero counter value will force both the low side and high side outputs into their OFF state. When the output matrix (OTMX) 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 37-34. 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 37-35 Pattern Generator Block Diagram](#) on page 839.

**Figure 37-35. 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.

#### 37.6.4. Master/Slave Operation

Two TCC instances sharing the same GCLK\_TCC clock, can be linked to provide more synchronized CC channels. The operation is enabled by setting the Master Synchronization bit in Control A register (CTRLA.MSYNC) in the Slave instance. When the bit is set, the slave TCC instance will synchronize the CC channels to the Master counter.

#### Related Links

[CTRLA](#) on page 848

#### 37.6.5. DMA, Interrupts, and Events

**Table 37-7. 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 For capture channel: when CCx register is read
Retrigger	Yes	Yes			

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
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

### 37.6.5.1. 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 DSBOOTH operation only.

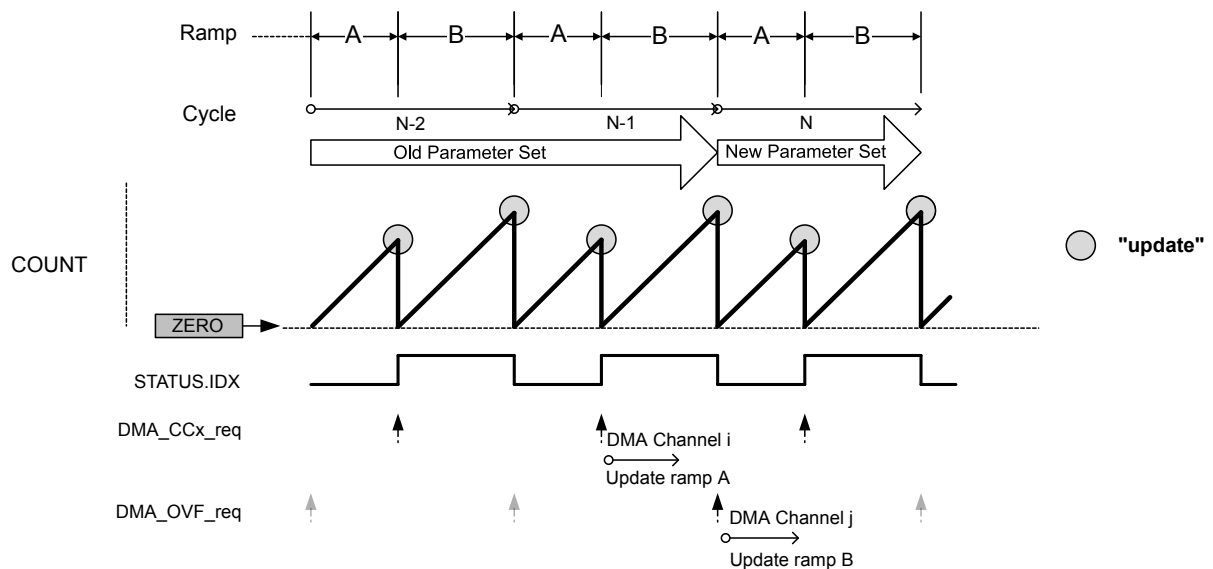
#### DMA Operation with Circular Buffer in RAMP and RAMP2A Mode

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 37-36. DMA Triggers in RAMP and RAMP2 Operation Mode and Circular Buffer Enabled**



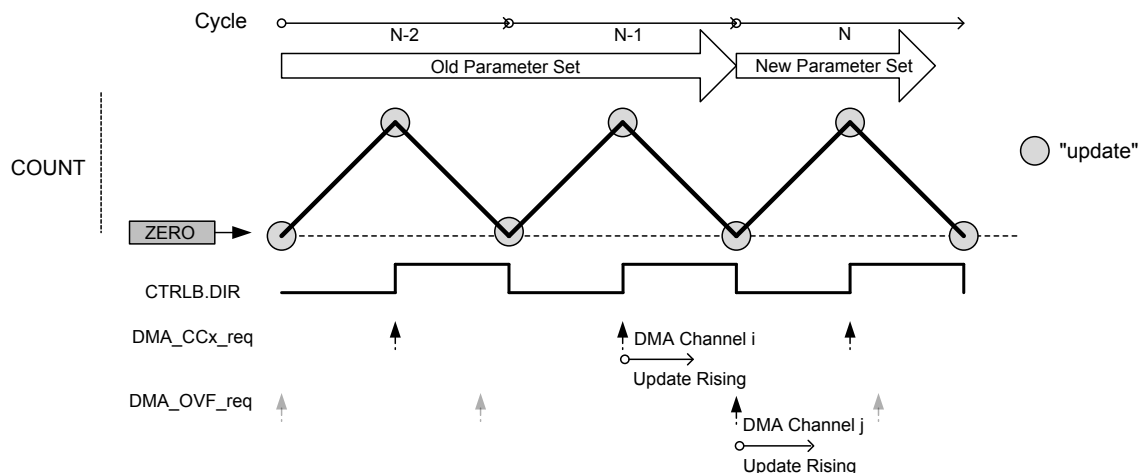
#### DMA Operation with Circular Buffer in DSBOOTH Mode

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 37-37. DMA Triggers in DSBOTH Operation Mode and Circular Buffer Enabled**



### 37.6.5.2. 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)
- Non-Recoverable Update Fault (UFS)
- 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](#) on page 874 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](#) on page 41

### 37.6.5.3. 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](#) on page 474

### 37.6.6. 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.

### 37.6.7. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the

synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

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)
- 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](#) on page 110

[Register Synchronization](#) on page 110

## 37.7. Register Summary

Offset	Name	Bit Pos.							
0x00	CTRLA	7:0		RESOLUTION[1:0]				ENABLE	SWRST
0x01		15:8	MSYNC	ALOCK	PRESCSYNC[1:0]	RUNSTDBY	PRESCALER[2:0]		
0x02		23:16							
0x03		31:24					CPTEN3	CPTEN2	CPTEN1
0x04	CTRLBCLR	7:0		CMD[2:0]		IDXCMD[1:0]	ONESHOT	LUPD	DIR
0x05	CTRLBSET	15:8		CMD[2:0]		IDXCMD[1:0]	ONESHOT	LUPD	DIR
0x06	Reserved								
0x07	Reserved								



Offset	Name	Bit Pos.									
0x08	SYNCBUSY	7:0	PER	WAVE	PATT	COUNT	STATUS	CTRLB	ENABLE	SWRST	
0x09		15:8					CC3	CC2	CC1	CC0	
0x0A		23:16		CCB3	CCB2	CCB1	CCB0	PERB	WAVEB	PATTB	
0x0B		31:24									
0x0C	FCTRLA	7:0	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]		
0x0D		15:8	BLANKPRES C	CAPTURE[2:0]			CHSEL[1:0]		HALT[1:0]		
0x0E		23:16	BLANKVAL[7:0]								
0x0F		31:24	FILTERVAL[3:0]								
0x10	FCTRLB	7:0	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]		
0x11		15:8	BLANKPRES C	CAPTURE[2:0]			CHSEL[1:0]		HALT[1:0]		
0x12		23:16	BLANKVAL[7:0]								
0x13		31:24	FILTERVAL[3:0]								
0x14	WEXCTRL	7:0							OTMX[1:0]		
0x15		15:8					DTIEN3	DTIEN2	DTIEN1	DTIEN0	
0x16		23:16	DTLS[7:0]								
0x17		31:24	DTHS[7:0]								
0x18	DRVCTRL	7:0	NRE7	NRE6	NRE5	NRE4	NRE3	NRE2	NRE1	NRE0	
0x19		15:8	NRV7	NRV6	NRV5	NRV4	NRV3	NRV2	NRV1	NRV0	
0x1A		23:16	INVEN7	INVEN6	INVEN5	INVEN4	INVEN3	INVEN2	INVEN1	INVEN0	
0x1B		31:24	FILTERVAL1[3:0]				FILTERVAL0[3:0]				
0x1C	Reserved										
0x1D	Reserved										
0x1E	DBGCTRL	7:0						FDDBD		DBGRUN	
0x1F	Reserved										
0x20	EVCTRL	7:0	CNTSEL[1:0]		EVACT1[2:0]			EVACT0[2:0]			
0x21		15:8	TCEI1	TCEI0	TCEINV1	TCEINV0		CNTEO	TRGEO	OVFEO	
0x22		23:16					MCEI3	MCEI2	MCEI1	MCEI0	
0x23		31:24					MCEO3	MCEO2	MCEO1	MCEO0	
0x24	INTENCLR	7:0					ERR	CNT	TRG	OVF	
0x25		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS			
0x26		23:16					MC3	MC2	MC1	MC0	
0x27	Reserved										
0x28	INTENSET	7:0					ERR	CNT	TRG	OVF	
0x29		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS			
0x2A		23:16					MC3	MC2	MC1	MC0	
0x2B	Reserved										
0x2C	INTFLAG	7:0					ERR	CNT	TRG	OVF	
0x2D		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS			
0x2E		23:16					MC3	MC2	MC1	MC0	
0x2F	Reserved										
0x30	STATUS	7:0	PERBV	WAVEBV	PATTBV		DFS		IDX	STOP	
0x31		15:8	FAULT1	FAULT0	FAULTB	FAULTA	FAULT1IN	FAULT0IN	FAULTBIN	FAULTAIN	
0x32		23:16					CCBV3	CCBV2	CCBV1	CCBV0	
0x33		31:24					CMP3	CMP2	CMP1	CMP0	

Offset	Name	Bit Pos.								
0x34	COUNT	7:0	COUNT[7:0]							
0x35		15:8	COUNT[15:8]							
0x36		23:16	COUNT[23:16]							
0x37		31:24								
0x38	PATT	7:0	PGE7	PGE6	PGE5	PGE4	PGE3	PGE2	PGE1	PGE0
0x39		15:8	PGV7	PGV6	PGV5	PGV4	PGV3	PGV2	PGV1	PGV0
0x3A	Reserved									
0x3B	Reserved									
0x3C	WAVE	7:0	CIPEREN	RAMP[1:0]			WAVEGEN[2:0]			
0x3D		15:8				CICCEN3	CICCEN2	CICCEN1	CICCEN0	
0x3E		23:16				POL3	POL2	POL1	POL0	
0x3F		31:24				SWAP3	SWAP2	SWAP1	SWAP0	
0x40	PER	7:0	PER[7:0]							
0x41		15:8	PER[15:8]							
0x42		23:16	PER[23:16]							
0x43		31:24								
0x44	CC0	7:0	CC0[7:0]							
0x45		15:8	CC0[15:8]							
0x46		23:16	CC0[23:16]							
0x47		31:24								
0x48	CC1	7:0	CC1[7:0]							
0x49		15:8	CC1[15:8]							
0x4A		23:16	CC1[23:16]							
0x4B		31:24								
0x4C	CC2	7:0	CC2[7:0]							
0x4D		15:8	CC2[15:8]							
0x4E		23:16	CC2[23:16]							
0x4F		31:24								
0x50	CC3	7:0	CC3[7:0]							
0x51		15:8	CC3[15:8]							
0x52		23:16	CC3[23:16]							
0x53		31:24								
0x54 ... 0x63	Reserved									
0x64	PATTB	7:0	PGE[7:0]							
0x65		15:8	PGV[7:0]							
0x66	Reserved									
0x67	Reserved									
0x68	WAVEB	7:0	CIPERENB	RAMPB[1:0]			WAVEGENB[2:0]			
0x69		15:8				CICCENB3	CICCENB2	CICCENB1	CICCENB0	
0x6A		23:16				POLB3	POLB2	POLB1	POLB0	
0x6B		31:24				SWAPB3	SWAPB2	SWAPB1	SWAPB0	
0x6C	PERB	7:0	PERB[7:0]							
0x6D		15:8	PERB[15:8]							
0x6E		23:16	PERB[23:16]							
0x6F		31:24								

Offset	Name	Bit Pos.							
0x70	CCB0	7:0							CCB0[7:0]
0x71		15:8							CCB0[15:8]
0x72		23:16							CCB0[23:16]
0x73		31:24							
0x74	CCB1	7:0							CCB1[7:0]
0x75		15:8							CCB1[15:8]
0x76		23:16							CCB1[23:16]
0x77		31:24							
0x78	CCB2	7:0							CCB2[7:0]
0x79		15:8							CCB2[15:8]
0x7A		23:16							CCB2[23:16]
0x7B		31:24							CCB2[31:24]
0x7C	CCB3	7:0							
0x7D		15:8							CCB3[15:8]
0x7E		23:16							CCB3[23:16]
0x7F		31:24							

### 37.8. 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.

### 37.8.1. 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
	MSYNC	ALOCK	PRESCYNC[1:0]		RUNSTDBY	PRESCALER[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
		RESOLUTION[1:0]					ENABLE	SWRST
Access		R/W	R/W				R/W	R/W
Reset		0	0				0	0

#### Bit 15 – MSYNC: Master Synchronization (only for TCC slave instance)

This bit must be set if the TCC counting operation must be synchronized on its Master TCC.

This bit is not synchronized.

Value	Description
0	The TCC controls its own counter.
1	The counter is controlled by its Master TCC.

#### 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 37-8. 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. 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.

**Bits 27,26,25,24 – 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.

### 37.8.2. Control B Clear

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBSET) register.

**Name:** CTRLBCLR

**Offset:** 0x04

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

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 this bit is set, no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update ensures that all buffers registers are valid before an update is performed.

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
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.

#### 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).



### 37.8.3. Control B Set

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBCLR) register.

**Name:** CTRLBSET

**Offset:** 0x05

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

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 this bit is set, no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update can be used to ensure that all buffer registers are loaded with the desired values, before an 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.
1	The CCBx, PERB, PGVB, PGOB, and SWAPBx buffer registers values are <i>not</i> copied into CCx, PER, PGV, PGO and SWAPx registers.

#### 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).

### 37.8.4. 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
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
Access					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bit 18 – PERB: PERB 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: WAVEB 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: PATTB Synchronization Busy

This bit is cleared when the synchronization of PATTERNB register between the clock domains is complete.

This bit is set when the synchronization of PATTERNB 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.

**Bits 22,21,20,19 – CCBx: 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.

**Bits 11,10,9,8 – CCx: 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.

### 37.8.5. Fault Control A and B

**Name:** FCTRLn  
**Offset:** 0x0C+4\*x (x = 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
	BLANKPRESC	CAPTURE[2:0]			CHSEL[1:0]		HALT[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
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.

#### Bit 15 – BLANKPRESC: Recoverable Fault n Blanking Value Prescaler

This bit enables a factor 64 prescaler factor on used as base frequency of the BLANKVAL value.

Value	Description
0	Blank time is BLANKVAL* prescaled GCLK_TCC.
1	Blank time is BLANKVAL* 64 * prescaled GCLK_TCC.

#### Bits 14:12 – CAPTURE[2:0]: Recoverable Fault n Capture Action

These bits select the capture and Fault n interrupt/event conditions.

**Table 37-9. 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.
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

Value	Name	Description
0x2	INVERT	Inverted MCE <sub>x</sub> (x=0,1) event input
0x3	ALTFAULT	Alternate fault (A or B) state at the end of the previous period.



### 37.8.6. 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
					DTIEN3	DTIEN2	DTIEN1	DTIEN0
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 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 37-5 Output Matrix Channel Pin Routing Configuration](#) on page 837.

**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.

### 37.8.7. 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
	INVEN7	INVEN6	INVEN5	INVEN4	INVEN3	INVEN2	INVEN1	INVEN0
Access	R/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
		NRV6	NRV5	NRV4	NRV3	NRV2	NRV1	NRV0
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
	NRE7	NRE6	NRE5	NRE4	NRE3	NRE2	NRE1	NRE0
Access	R/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 16,14,13,12,11,10,9,8 – 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.

### 37.8.8. 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 enabled. OCD break request from the OCD system will trigger non-recoverable fault. When this bit is set, OCD fault protection is disabled and OCD break request will not trigger a 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.

### 37.8.9. Event Control

**Name:** EVCTRL  
**Offset:** 0x20  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
					MCEO3	MCEO2	MCEO1	MCEO0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					MCEI3	MCEI2	MCEI1	MCEI0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	TCEI1	TCEI0	TCINV1	TCINV0		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

#### 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 – OVFEO: Overflow/Underflow Event Output Enable

This bit is used to enable the overflow/underflow event. When enabled an event will be generated when the counter 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
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 (async)	Count on active state of asynchronous event
0x6	STAMP	Capture overflow times (Max value).
0x7	FAULT	Non-recoverable Fault

**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.

### 37.8.10. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x24

**Reset:** 0x000000

**Property:** PAC Write-Protection

Bit	23	22	21	20	19	18	17	16
					MC3	MC2	MC1	MC0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
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
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### 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: Debug Fault State 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 10 – UFS: Non-Recoverable Update Fault Interrupt Enable

Writing a zero to this bit has no effect.

Writing a one to this bit will clear the Non-Recoverable Update Fault Interrupt Disable/Enable bit, which disables the Non-Recoverable Update Fault interrupt.

Value	Description
0	The Non-Recoverable Update Fault interrupt is disabled.
1	The Non-Recoverable Update Fault 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.

**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.

### 37.8.11. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x28

**Reset:** 0x000000

**Property:** PAC Write-Protection

Bit	23	22	21	20	19	18	17	16
					MC3	MC2	MC1	MC0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
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
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### 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: Debug Fault State 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 10 – UFS: Non-Recoverable Update Fault Interrupt Enable

Writing a zero to this bit has no effect.

Writing a one to this bit will set the Non-Recoverable Update Fault Interrupt Disable/Enable bit, which enables the Non-Recoverable Update Fault interrupt.

Value	Description
0	The Non-Recoverable Update Fault interrupt is disabled.
1	The Non-Recoverable Update Fault 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.

**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.

### 37.8.12. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x2C  
**Reset:** 0x000000  
**Property:** -

Bit	23	22	21	20	19	18	17	16
					MC3	MC2	MC1	MC0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
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
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### 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 A occurs.

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

Writing a '1' to this bit clears the Recoverable Fault A interrupt flag.

#### Bit 11 – DFS: 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 10 – UFS: Non-Recoverable Update Fault

This flag is set when the RAMP index changes and the Lock Update bit is set (CTRLBSET.LUPD).

Writing a zero to this bit has no effect.

Writing a one to this bit clears the Non-Recoverable Update Fault 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.

**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.

### 37.8.13. Status

**Name:** STATUS  
**Offset:** 0x30  
**Reset:** 0x00000001  
**Property:** -

Bit	31	30	29	28	27	26	25	24
					CMP3	CMP2	CMP1	CMP0
Access					R	R	R	R
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					CCBV3	CCBV2	CCBV1	CCBV0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	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	SLAVE	DFS	UFS	IDX	STOP
Access	R/W	R/W	R/W	R	R/W	R/W	R	R
Reset	0	0	0	0	0	0	0	1

#### 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 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 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 or by writing a '1' to this bit.

**Bit 4 – SLAVE: Slave**

This bit is set when TCC is set in Slave mode. This bit follows the CTRLA.MSYNC bit state.

**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 2 – UFS: Non-recoverable Update Fault State**

This bit is set by hardware when the RAMP index changes and the Lock Update bit is set (CTRLBSET.LUPD). The bit is cleared by writing a one to this bit.

When the bit is set, 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](#) on page 828.

**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.

**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 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](#) on page 853.CMD).

### 37.8.14. Counter Value, CTRLA.RESOLUTION = NONE

**Name:** COUNT

**Offset:** 0x34

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

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.

### 37.8.15. Counter Value, CTRLA.RESOLUTION = DITH4

**Name:** COUNT

**Offset:** 0x34

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	COUNT[19:12]							
Access	R/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[11:4]							
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	7	6	5	4	3	2	1	0
	COUNT[3:0]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

**Bits 23:4 – COUNT[19:0]: Counter Value**

These bits hold the value of the counter register.

### 37.8.16. Counter Value, CTRLA.RESOLUTION = DITH5

**Name:** COUNT

**Offset:** 0x34

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

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

#### Bits 23:5 – COUNT[18:0]: Counter Value

These bits hold the value of the counter register.

### 37.8.17. Counter Value, CTRLA.RESOLUTION = DITH6

**Name:** COUNT

**Offset:** 0x34

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	COUNT[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
	COUNT[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
	COUNT[1:0]							
Access	R/W	R/W						
Reset	0	0						

#### Bits 23:6 – COUNT[17:0]: Counter Value

These bits hold the value of the counter register.

### 37.8.18. Pattern

**Name:** PATT

**Offset:** 0x38

**Reset:** 0x0000

**Property:** Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
	PGV7	PGV6	PGV5	PGV4	PGV3	PGV2	PGV1	PGV0
Access	R/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
	PGE7	PGE6	PGE5	PGE4	PGE3	PGE2	PGE1	PGE0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

#### Bits 15:8 – PGVn: Pattern Generation Output Value

This register holds the values of pattern for each waveform output.

#### Bits 7:0 – PGE<sub>n</sub>: 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 PGV<sub>n</sub> value.

### 37.8.19. Waveform

**Name:** WAVE  
**Offset:** 0x3C  
**Reset:** 0x00000000  
**Property:** Write-Synchronized, Read-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
					CIRCCEN3	CIRCCEN2	CIRCCEN1	CIRCCEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	CIRPEREN		RAMP[1:0]			WAVEGEN[2:0]		
Access	R/W		R/W	R/W		R/W	R/W	R/W
Reset	0		0	0		0	0	0

#### Bit 7 – CIRPEREN: 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.

#### Bits 5:4 – RAMP[1:0]: Ramp Operation

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	RAMP2C	Critical RAMP2 operation

#### 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	Reserved	–	–	–	–	–	–	–
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	–

#### Bits 27,26,25,24 – SWAPx : 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 19,18,17,16 – POLx: 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 11,10,9,8 – CIRCCENx: 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.

### 37.8.20. Period Value, CTRLA.RESOLUTION = NONE

**Name:** PER  
**Offset:** 0x40  
**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
Access	PER[23:16]							
Reset	1	1	1	1	1	1	1	1
Bit	15	14	13	12	11	10	9	8
Access	PER[15:8]							
Reset	1	1	1	1	1	1	1	1
Bit	7	6	5	4	3	2	1	0
Access	PER[7:0]							
Reset	1	1	1	1	1	1	1	1

#### Bits 23:0 – PER[23:0]: Period Value

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition. The number of bits in this field corresponds to the size of the counter.

### 37.8.21. Period Value, CTRLA.RESOLUTION = DITH4

**Name:** PER  
**Offset:** 0x40  
**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
	PER[19:12]							
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
	PER[11:4]							
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
	PER[3:0]				DITHERCY[3:0]			
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 23:4 – PER[19:0]: Period Value

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition. The number of bits in this field corresponds to the size of the counter.

#### Bits 3:0 – DITHERCY[3:0]: Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM period every 16 PWM frames.

### 37.8.22. Period Value, CTRLA.RESOLUTION = DITH5

**Name:** PER  
**Offset:** 0x40  
**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
Access	PER[18:11]							
Reset	1	1	1	1	1	1	1	1
Bit	15	14	13	12	11	10	9	8
Access	PER[10:3]							
Reset	1	1	1	1	1	1	1	1
Bit	7	6	5	4	3	2	1	0
Access	PER[2:0]			DITHERCY[4:0]				
Reset	1	1	1	1	1	1	1	1

#### Bits 23:5 – PER[18:0]: Period Value

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition. The number of bits in this field corresponds to the size of the counter.

#### Bits 4:0 – DITHERCY[4:0]: Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM period every 32 PWM frames.

### 37.8.23. Period Value, CTRLA.RESOLUTION = DITH6

**Name:** PER  
**Offset:** 0x40  
**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
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]		DITHERCY[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. The value is copied to PER register on UPDATE condition. The number of bits in this field corresponds to the size of the counter.

#### Bits 5:0 – DITHERCY[5:0]: Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM period every 64 PWM frames.

### 37.8.24. Compare/Capture Channel x, CTRLA.RESOLUTION = NONE

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 from 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.

**Name:** CCx

**Offset:** 0x44 + 4\*x [x = 0..3]

**Reset:** 0x00000000

**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
	CC0[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
	CC0[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
	CC0[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 – CCx: Channel x Compare/Capture Value

These bits hold the value of the Channel x compare/capture register.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (CCx[23:16]) are read zero when TCC is configured as 16-bits timer/counter.

### 37.8.25. Compare/Capture Channel x, CTRLA.RESOLUTION = DITH4

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 from 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.

**Name:** CCx  
**Offset:** 0x44 + 4\*x [x = 0..3]  
**Reset:** 0x00000000  
**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
	CC0[19:12]							
Access	R/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
	CC0[11:4]							
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	7	6	5	4	3	2	1	0
	CC0[3:0]				DITHERCY[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

#### Bits 23:4 – CCx: Channel x Compare/Capture Value

These bits hold the value of the Channel x compare/capture register.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (CCx[19:12]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 3:0 – DITHERCY[3:0]: Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM pulse width every 16 PWM frames.

### 37.8.26. Compare/Capture Channel x, CTRLA.RESOLUTION = DITH5

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 from 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.

**Name:** CCx  
**Offset:** 0x44 + 4\*x [x = 0..3]  
**Reset:** 0x00000000  
**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
	CC0[18:11]							
Access	R/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
	CC0[10:3]							
Access	R/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
	CC0[2:0]			DITHERCY[4:0]				
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 23:5 – CCx: Channel x Compare/Capture Value

These bits hold the value of the Channel x compare/capture register.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (CCx[18:11]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 4:0 – DITHERCY[4:0]: Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM pulse width every 32 PWM frames.



### 37.8.27. Compare/Capture Channel x, CTRLA.RESOLUTION = DITH6

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 from 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.

**Name:** CCx  
**Offset:** 0x44 + 4\*x [x = 0..3]  
**Reset:** 0x00000000  
**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
	CC0[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
	CC0[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
	CC0[1:0]		DITHERCY[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 – CCx: Channel x Compare/Capture Value

These bits hold the value of the Channel x compare/capture register.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (CCx[17:10]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 5:0 – DITHERCY[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.

### 37.8.28. Pattern Buffer

**Name:** PATTBUF

**Offset:** 0x64

**Reset:** 0x0000

**Property:** Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
	PGVB7	PGVB6	PGVB5	PGVB4	PGVB3	PGVB2	PGVB1	PGVB0
Access	R/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
	PGEb7	PGEb6	PGEb5	PGEb4	PGEb3	PGEb2	PGEb1	PGEb0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

#### Bits 15:8 – PGVBn: 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 7:0 – PGEbn: 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.

### 37.8.29. Waveform Buffer

**Name:** WAVEBUF  
**Offset:** 0x68  
**Reset:** 0x00000000  
**Property:** Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
					SWAPB3	SWAPB2	SWAPB1	SWAPB0
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
Reset	0			0	0			0

#### 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.

#### Bits 27,26,25,24 – SWAPBx : 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 19,18,17,16 – POLBx: 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 11,10,9,8 – CICCENBx: 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.

### 37.8.30. Period Buffer Value, CTRLA.RESOLUTION=NONE

**Name:** PERBUF  
**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
	PERBUF[23:16]							
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
	PERBUF[15:8]							
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
	PERBUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

#### Bits 23:0 – PERBUF[23:0]: Period Buffer Value

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (PERBUF[23:16]) are read zero when TCC is configured as 16-bits timer/counter.

### 37.8.31. Period Buffer Value, CTRLA.RESOLUTION=DITH4

**Name:** PERBUF  
**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
Access					PERBUF[19:16]			
Reset					R/W	R/W	R/W	R/W
Reset					1	1	1	1
Bit	15	14	13	12	11	10	9	8
Access	PERBUF[15:8]							
Reset	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
Access	PERBUF[7:0]							
Reset	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 19:0 – PERBUF[19:0]: Period Buffer Value

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (PERBUF[19:12]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 3:0 – DITHERBUF[3:0]: Dithering Buffer Cycle Number

These bits represent the PER.DITHER bits buffer. When the double buffering is enable, DITHERBUF bits value is copied to the PER.DITHER bits on an UPDATE condition.

### 37.8.32. Period Buffer Value, CTRLA.RESOLUTION=DITH5

**Name:** PERBUF  
**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
Access						PERBUF[18:16]		
Reset						1	1	1
Bit	15	14	13	12	11	10	9	8
Access	PERBUF[15:8]							
Reset	1	1	1	1	1	1	1	1
Bit	7	6	5	4	3	2	1	0
Access	PERBUF[7:0]							
Reset	1	1	1	1	1	1	1	1

#### Bits 18:0 – PERBUF[18:0]: Period Buffer Value

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (PERBUF[18:11]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 4:0 – DITHERBUF[4:0]: Dithering Buffer Cycle Number

These bits represent the PER.DITHER bits buffer. When the double buffering is enable, DITHERBUF bits value is copied to the PER.DITHER bits on an UPDATE condition.

### 37.8.33. Period Buffer Value, CTRLA.RESOLUTION=DITH6

**Name:** PERBUF  
**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
Access							PERBUF[17:16]	
Reset							1	1
Bit	15	14	13	12	11	10	9	8
Access	PERBUF[15:8]							
Reset	1	1	1	1	1	1	1	1
Bit	7	6	5	4	3	2	1	0
Access	PERBUF[7:0]							
Reset	1	1	1	1	1	1	1	1

#### Bits 17:0 – PERBUF[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.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (PERBUF[17:10]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 5:0 – DITHERBUF[5:0]: Dithering Buffer Cycle Number

These bits represent the PER.DITHER bits buffer. When the double buffering is enable, DITHERBUF bits value is copied to the PER.DITHER bits on an UPDATE condition.

### 37.8.34. Channel x Compare/Capture Buffer Value, CTRLA.RESOLUTION=NONE

CCBUF<sub>x</sub> is copied into CC<sub>x</sub> at TCC update time

**Name:** CCBUF<sub>x</sub>

**Offset:** 0x70 + 4\*x [x = 0..3]

**Reset:** 0x00000000

**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
	CCBUF0[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
	CCBUF0[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
	CCBUF0[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 – CCBUF<sub>x</sub>: Channel x Compare/Capture 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 (CC<sub>x</sub>). Accessing this register using the CPU or DMA will affect the corresponding CCBV<sub>x</sub> status bit.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (CCBUF[23:16]) are read zero when TCC is configured as 16-bits timer/counter.



### 37.8.35. Channel x Compare/Capture Buffer Value, CTRLA.RESOLUTION=DITH4

CCBUFx is copied into CCx at TCC update time

**Name:** CCBUFx

**Offset:** 0x70 + 4\*x [x = 0..3]

**Reset:** 0x00000000

**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
	CCBUF0[19:12]							
Access	R/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
	CCBUF0[11:4]							
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	7	6	5	4	3	2	1	0
	CCBUF0[3:0]				DITHERBUF[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

#### Bits 23:4 – CCBUFx: 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.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (CCBUF[19:12]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 3:0 – DITHERBUF[3:0]: 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.

### 37.8.36. Channel x Compare/Capture Buffer Value, CTRLA.RESOLUTION=DITH5

CCBUFx is copied into CCx at TCC update time

**Name:** CCBUFx

**Offset:** 0x70 + 4\*x [x = 0..3]

**Reset:** 0x00000000

**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
	CCBUF0[18:11]							
Access	R/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
	CCBUF0[10:3]							
Access	R/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
	CCBUF0[2:0]			DITHERBUF[4:0]				
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 23:5 – CCBUFx: 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.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (CCBUF[18:11]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 4:0 – DITHERBUF[4:0]: 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.

### 37.8.37. Channel x Compare/Capture Buffer Value, CTRLA.RESOLUTION=DITH5

CCBUFx is copied into CCx at TCC update time

**Name:** CCBUFx

**Offset:** 0x70 + 4\*x [x = 0..3]

**Reset:** 0x00000000

**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
	CCBUF0[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
	CCBUF0[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
	CCBUF0[1:0]		DITHERBUF[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 – CCBUFx: 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.

Bit 31:24 are read zero when TCC is configured as 16- or 24-bits timer/counter.

Bit 23:16 (CCBUF[17:10]) are read zero when TCC is configured as 16-bits timer/counter.

#### Bits 5:0 – DITHERBUF[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.

## 38. CCL – Configurable Custom Logic

### 38.1. Overview

The Configurable Custom Logic (CCL) is a programmable logic peripheral which can be connected to the device pins, to events, or to other internal peripherals. This allows the user to eliminate logic gates for simple glue logic functions on the PCB.

Each Lookup Table (LUT) consists of three inputs, a truth table, and as options synchronizer, filter and edge detector. Each LUT can generate an output as a user programmable logic expression with three inputs. Inputs can be individually masked.

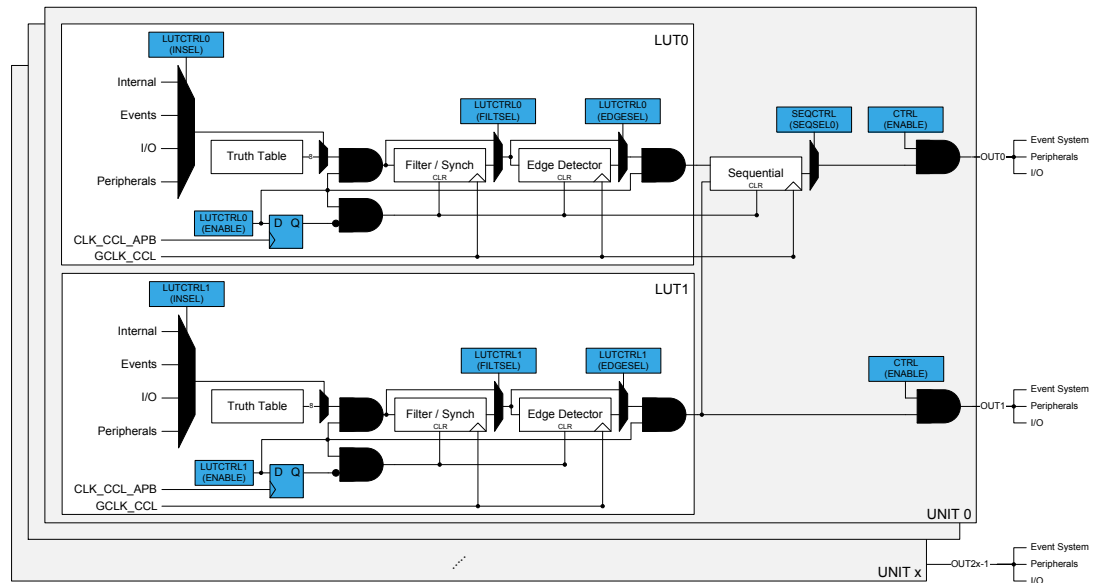
The output can be combinatorially generated from the inputs, and can be filtered to remove spikes. An optional sequential module can be enabled. The inputs of the sequential module are individually controlled by two independent, adjacent LUT (LUT0/LUT1, LUT2/LUT3 etc) outputs, enabling complex waveform generation.

### 38.2. Features

- Glue logic for general purpose PCB design
- Up to four Programmable LookUp Table (LUT)
- Combinatorial Logic Functions:  
AND, NAND, OR, NOR, XOR, XNOR, NOT
- Sequential Logic Functions:  
Gated D Flip-Flop, JK Flip-Flop, gated D Latch, RS Latch
- Flexible LookUp Table Inputs Selection:
  - I/Os
  - Events
  - Internal Peripherals
  - Subsequent LUT Output
- Output can be connected to IO pins or Event System
- Optional synchronizer, filter, or edge detector available on each LUT output

### 38.3. Block Diagram

Figure 38-1. Configurable Custom Logic



### 38.4. Signal Description

Pin Name	Type	Description
OUT[n]-OUT0	Digital output	Output from lookup table
IN[3n+2] - IN0	Digital input	Input to lookup table

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](#) on page 27

### 38.5. Product Dependencies

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

#### 38.5.1. I/O Lines

Using the CCL I/O lines requires the I/O pins to be configured. Refer to *PORT - I/O Pin Controller* for details.

#### Related Links

[PORT: IO Pin Controller](#) on page 444

#### 38.5.2. Power Management

This peripheral can continue to operate in any sleep mode where its source clock is running. Events connected to the event system can trigger other operations in the system without exiting sleep modes. Refer to *PM – Power Manager* for details on the different sleep modes.

#### Related Links

[PM – Power Manager](#) on page 162

### 38.5.3. Clocks

The CCL bus clock (CLK\_CCL\_APB) can be enabled and disabled in the power manager, and the default state of CLK\_CCL\_APB can be found in the *Peripheral Clock Masking*.

A generic clock (GCLK\_CCL) is optionally required to clock the CCL. This clock must be configured and enabled in the Generic Clock Controller (GCLK) before using the sequential sub-module of CCL. GCLK\_CCL is required when input events, a filter, an edge detector, or a sequential sub-module is enabled. Refer to *GCLK - Generic Clock Controller* for details.

This generic clock is asynchronous to the user interface clock (CLK\_CCL\_APB).

#### Related Links

[Peripheral Clock Masking](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

### 38.5.4. DMA

Not applicable.

### 38.5.5. Interrupts

Not applicable.

### 38.5.6. 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](#) on page 474

### 38.5.7. Debug Operation

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

### 38.5.8. Register Access Protection

All registers with write-access can be write-protected optionally by the peripheral access controller (PAC). Refer to *PAC - Peripheral Access Controller* for details.

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](#) on page 46

### 38.5.9. Analog Connections

Not applicable.

## 38.6. Functional Description

### 38.6.1. Principle of Operation

Configurable Custom Logic (CCL) is a programmable logic block that can use the device port pins and the internal Event System as both input and output channels. The CCL can serve as glue logic between the device and external devices. This increases the reliability of the PCB by reducing its complexity, and enables more powerful functions.

### 38.6.2. Basic Operation

#### 38.6.2.1. Initialization

The following bits are enable-protected, meaning that they can only be written when the corresponding even LUT is disabled ( $LUTCTRLx.ENABLE=0$ ):

- Sequential Selection in Sequential Control x register ( $SEQCTRLx.SEQSEL$ )

The following registers are enable-protected, meaning that they can only be written when the corresponding LUT is disabled ( $LUTCTRLx.ENABLE=0$ ):

- LUT Control x register, except ENABLE bit ( $LUTCTRLx$ )

Enable-protected bits in the  $LUTCTRLx$  registers can be written at the same time as  $LUTCTRLx.ENABLE$  is written to '1', but not at the same time as  $LUTCTRLx.ENABLE$  is written to '0'.

Enable-protection is denoted by the Enable-Protected property in the register description.

#### 38.6.2.2. Enabling, Disabling, and Resetting

The CCL is enabled by writing a '1' to the Enable bit in the Control register ( $CTRL.ENABLE$ ). The CCL is disabled by writing a '0' to  $CTRL.ENABLE$ .

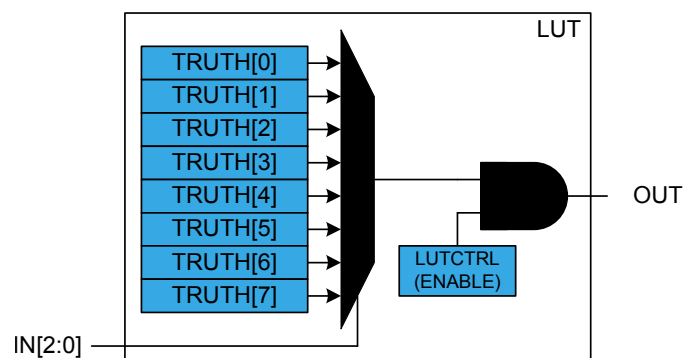
Each LUT is enabled by writing a '1' to the Enable bit in the LUT Control x register ( $LUTCTRLx.ENABLE$ ). Each LUT is disabled by writing a '0' to  $LUTCTRLx.ENABLE$ .

The CCL is reset by writing a '1' to the Software Reset bit in the Control register ( $CTRL.SWRST$ ). All registers in the CCL will be reset to their initial state, and the CCL will be disabled. Refer to [CTRL](#) on page 919 for details.

#### 38.6.2.3. Lookup Table Logic

The lookup table in each LUT unit can generate any logic expression  $OUT$  as a function of three inputs ( $IN[2:0]$ ), as shown in [Figure 38-2 Truth Table Output Value Selection](#) on page 907. One or more inputs can be masked. The truth table for the expression is defined by  $TRUTH$  bits in LUT Control x register ( $LUTCTRLx.TRUTH$ ).

Figure 38-2. Truth Table Output Value Selection



**Table 38-1. Truth Table of LUT**

IN[2]	IN[1]	IN[0]	OUT
0	0	0	TRUTH[0]
0	0	1	TRUTH[1]
0	1	0	TRUTH[2]
0	1	1	TRUTH[3]
1	0	0	TRUTH[4]
1	0	1	TRUTH[5]
1	1	0	TRUTH[6]
1	1	1	TRUTH[7]

### 38.6.2.4. Truth Table Inputs Selection

#### Input Overview

The inputs can be individually:

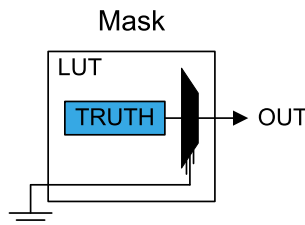
- Masked
- Driven by peripherals:
  - Analog comparator output (AC)
  - Timer/Counters waveform outputs (TC)
  - Serial Communication output transmit interface (SERCOM)
- Driven by internal events from Event System
- Driven by other CCL sub-modules

The Input Selection for each input  $y$  of LUT  $x$  is configured by writing the Input  $y$  Source Selection bit in the LUT  $x$  Control register (LUTCTRLx.INSELY).

#### Masked Inputs (MASK)

When a LUT input is masked (LUTCTRLx.INSELY=MASK), the corresponding TRUTH input (IN) is internally tied to zero, as shown in this figure:

**Figure 38-3. Masked Input Selection**



#### Internal Feedback Inputs (FEEDBACK)

When selected (LUTCTRLx.INSELY=FEEDBACK), the Sequential (SEQ) output is used as input for the corresponding LUT.

The output from an internal sequential sub-module can be used as input source for the LUT, see figure below for an example for LUT0 and LUT1. The sequential selection for each LUT follows the formula:

$$IN[2N][i] = SEQ[N]$$

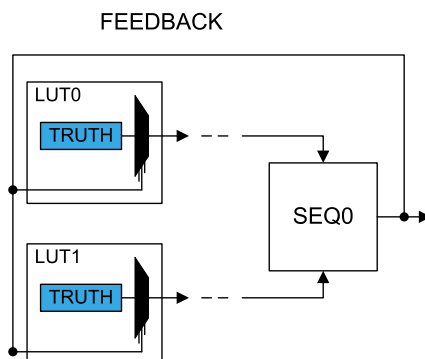


$$IN[2N+1][i] = SEQ[N]$$

With  $N$  representing the sequencer number and  $i=0,1,2$  representing the LUT input index.

For details, refer to [Sequential Logic](#) on page 914.

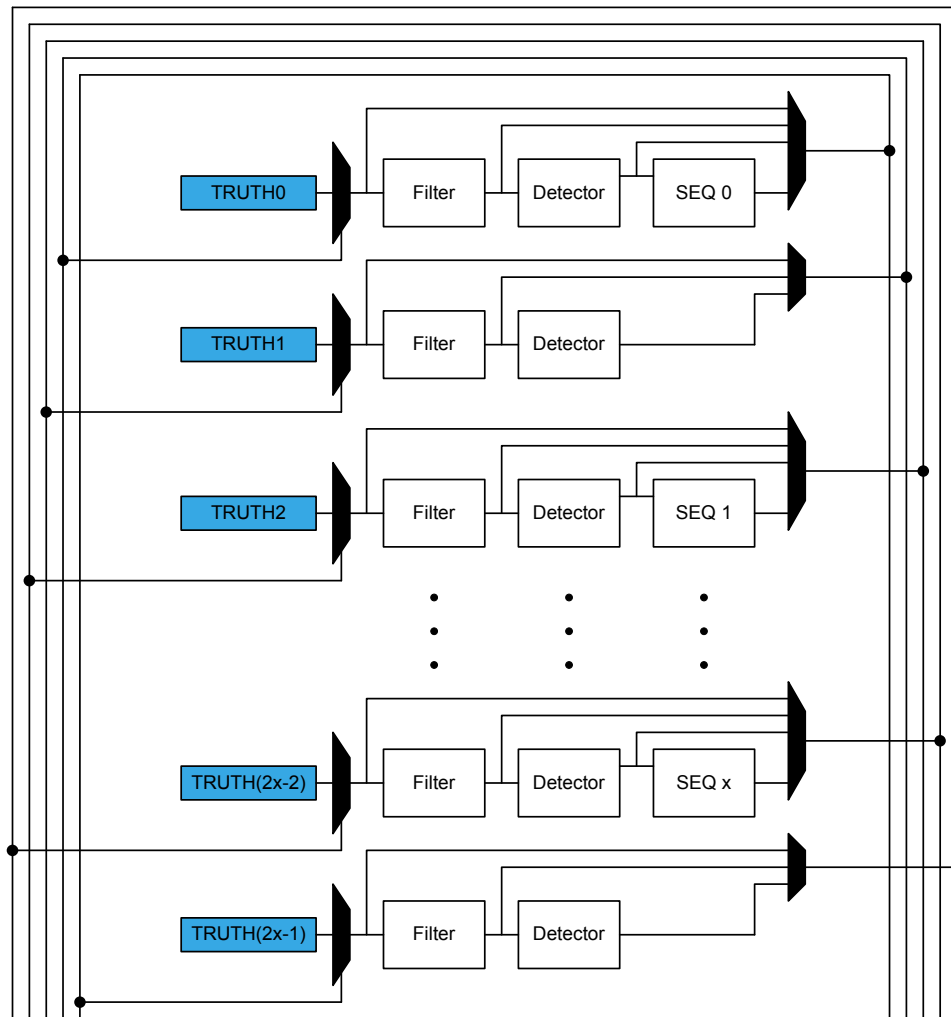
**Figure 38-4. Feedback Input Selection**



### Linked LUT (LINK)

When selected ( $LUTCTRLx.INSELY=LINK$ ), the subsequent LUT output is used as the LUT input (e.g., LUT2 is the input for LUT1), as shown in this figure:

Figure 38-5. Linked LUT Input Selection



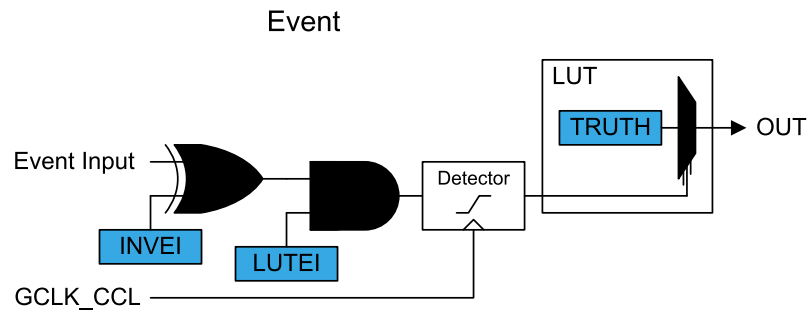
#### Internal Events Inputs Selection(EVENT)

Asynchronous events from the Event System can be used as input selection, as shown in [Figure 38-6 Event Input Selection](#) on page 911. For each LUT, one event input line is available and can be selected on each LUT input. Before enabling the event selection by writing `LUTCTRLx.INSELY=EVENT`, the Event System must be configured first.

The CCL includes an edge detector. When the event is received, an internal strobe is generated when a rising edge is detected. The pulse duration is one `GCLK_CCL` clock cycle. The following steps ensure proper operation:

1. Enable the `GCLK_CCL` clock
2. Configure the Event System to route the event asynchronously
3. If a strobe must be generated on the event input falling edge, write a '1' to the Inverted Event Input Enable bit in LUT Control register (`LUTCTRLx.INVEI`)
4. Enable the event input by writing the Event Input Enable bit in LUT Control register (`LUTCTRLx.LUTEI`) to '1'.

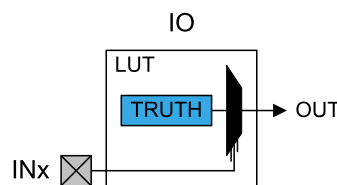
**Figure 38-6. Event Input Selection**



**I/O Pin Inputs (IO)**

When the IO pin is selected as LUT input (LUTCTRLx.INSELY=IO), the corresponding LUT input will be connected to the pin, as shown in the figure below.

**Figure 38-7. I/O Pin Input Selection**



**Analog Comparator Inputs (AC)**

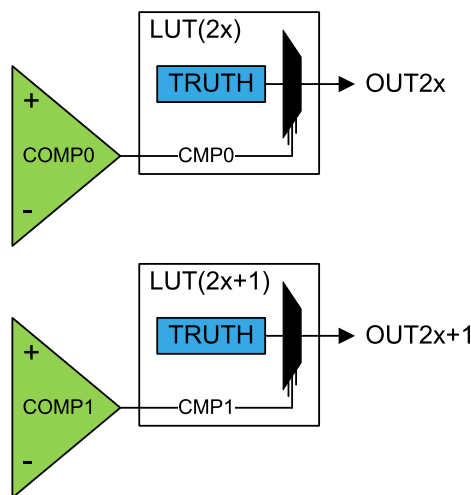
The AC outputs can be used as input source for the LUT (LUTCTRLx.INSELY=AC). The output of comparator 0 is available on even LUTs ("LUT(2x)": LUT0, LUT2) and the comparator 1 output is available on odd LUTs ("LUT(2x+1)": LUT1, LUT3), as shown in the figure below. More general, the analog comparator outputs are distributed following the formula:

$$IN[N][i] = AC[N \% \text{ComparatorOutput\_Number}]$$

With  $N$  representing the LUT number and  $i=[0,1,2]$  representing the LUT input index.

Before selecting the comparator output, the AC must be configured first.

**Figure 38-8. AC Input Selection**



### Timer/Counter Inputs (TC)

The TC waveform output WO[0] can be used as input source for the LUT (LUTCTRLx.INSELY=TC). Only consecutive instances of the TC, i.e. TCx and the subsequent TC(x+1), are available as default and alternative TC selections (e.g., TC0 and TC1 are sources for LUT0, TC1 and TC2 are sources for LUT1, etc). See the figure below for an example for LUT0. More general, the Timer/Counter selection for each LUT follows the formula:

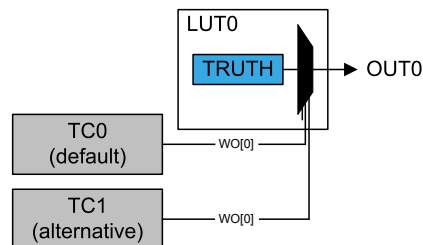
$$IN[N][i] = DefaultTC[N \% TC\_Instance\_Number]$$

$$IN[N][i] = AlternativeTC[(N + 1) \% TC\_Instance\_Number]$$

Where N represents the LUT number and i represents the LUT input index (i=0,1,2).

Before selecting the waveform outputs, the TC must be configured first.

**Figure 38-9. TC Input Selection**



### Timer/Counter for Control Application Inputs (TCC)

The TCC waveform outputs can be used as input source for the LUT. Only WO[2:0] outputs can be selected and routed to the respective LUT input (i.e., IN0 is connected to WO0, IN1 to WO1, and IN2 to WO2), as shown in the figure below.

#### Note:

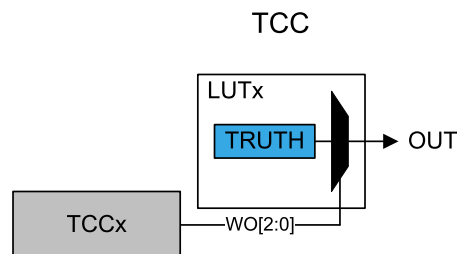
The TCC selection for each LUT follows the formula:

$$IN[N][i] = TCC[N \% TCC\_Instance\_Number]$$

Where N represents the LUT number.

Before selecting the waveform outputs, the TCC must be configured first.

**Figure 38-10. TCC Input Selection**



### Serial Communication Output Transmit Inputs (SERCOM)

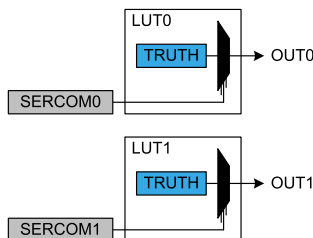
The serial engine transmitter output from Serial Communication Interface (SERCOM TX, TXd for USART, MOSI for SPI) can be used as input source for the LUT. The figure below shows an example for LUT0 and LUT1. The SERCOM selection for each LUT follows the formula:

$$IN[N][i] = SERCOM[N \% SERCOM\_Instance\_Number]$$

With N representing the LUT number and i=0,1,2 representing the LUT input index.

Before selecting the SERCOM as input source, the SERCOM must be configured first: the SERCOM TX signal must be output on SERCOMn/pad[0], which serves as input pad to the CCL.

**Figure 38-11. SERCOM Input Selection**



#### Related Links

- [I/O Multiplexing and Considerations](#) on page 27
- [PORT: IO Pin Controller](#) on page 444
- [GCLK - Generic Clock Controller](#) on page 114
- [AC – Analog Comparators](#) on page 1006
- [TC – Timer/Counter](#) on page 758
- [TCC – Timer/Counter for Control Applications](#) on page 809
- [SERCOM – Serial Communication Interface](#) on page 497
- [I/O Multiplexing and Considerations](#) on page 27

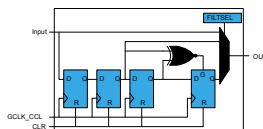
#### 38.6.2.5. Filter

By default, the LUT output is a combinatorial function of the LUT inputs. This may cause some short glitches when the inputs change value. These glitches can be removed by clocking through filters, if demanded by application needs.

The Filter Selection bits in LUT Control register (LUTCTRLx.FILTSEL) define the synchronizer or digital filter options. When a filter is enabled, the OUT output will be delayed by two to five GCLK cycles. One APB clock after the corresponding LUT is disabled, all internal filter logic is cleared.

**Note:** Events used as LUT input will also be filtered, if the filter is enabled.

**Figure 38-12. Filter**



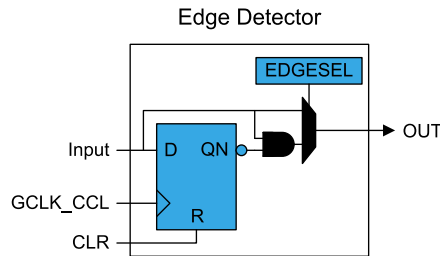
#### 38.6.2.6. Edge Detector

The edge detector can be used to generate a pulse when detecting a rising edge on its input. To detect a falling edge, the TRUTH table should be programmed to provide the opposite levels.

The edge detector is enabled by writing '1' to the Edge Selection bit in LUT Control register (LUTCTRLx.EDGESEL). In order to avoid unpredictable behavior, a valid filter option must be enabled as well.

Edge detection is disabled by writing a '0' to LUTCTRLx.EDGESEL. After disabling a LUT, the corresponding internal Edge Detector logic is cleared one APB clock cycle later.

**Figure 38-13. Edge Detector**



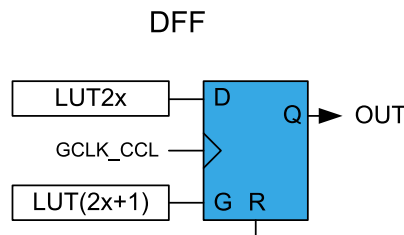
### 38.6.2.7. Sequential Logic

Each LUT pair can be connected to internal sequential logic: D flip flop, JK flip flop, gated D-latch or RS-latch can be selected by writing the corresponding Sequential Selection bits in Sequential Control x register (SEQCTRLx.SEQSEL). Before using sequential logic, the GCLK clock and optionally each LUT filter or edge detector, must be enabled.

#### Gated D Flip-Flop (DFF)

When the DFF is selected, the D-input is driven by the even LUT output (LUT2x), and the G-input is driven by the odd LUT output (LUT2x+1), as shown in [Figure 38-14 D Flip Flop](#) on page 914.

**Figure 38-14. D Flip Flop**



When the even LUT is disabled (LUTCTRL2x.ENABLE=0), the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one APB clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the GCLK\_CCL, as shown in [Table 38-2 DFF Characteristics](#) on page 914.

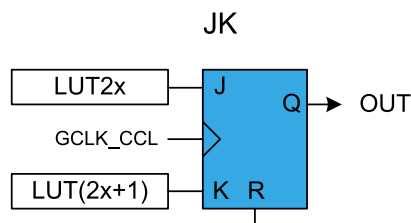
**Table 38-2. DFF Characteristics**

R	G	D	OUT
1	X	X	Clear
0	1	1	Set
		0	Clear
	0	X	Hold state (no change)

#### JK Flip-Flop (JK)

When this configuration is selected, the J-input is driven by the even LUT output (LUT2x), and the K-input is driven by the odd LUT output (LUT2x+1), as shown in [Figure 38-15 JK Flip Flop](#) on page 915.

**Figure 38-15. JK Flip Flop**



When the even LUT is disabled (LUTCTRL2x.ENABLE=0), the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one APB clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the GCLK\_CCL, as shown in [Table 38-3 JK Characteristics](#) on page 915.

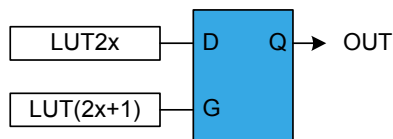
**Table 38-3. JK Characteristics**

R	J	K	OUT
1	X	X	Clear
0	0	0	Hold state (no change)
0	0	1	Clear
0	1	0	Set
0	1	1	Toggle

#### Gated D-Latch (DLATCH)

When the DLATCH is selected, the D-input is driven by the even LUT output (LUT2x), and the G-input is driven by the odd LUT output (LUT2x+1), as shown in [Figure 38-14 D Flip Flop](#) on page 914.

**Figure 38-16. D-Latch**



When the even LUT is disabled (LUTCTRL2x.ENABLE=0), the latch output will be cleared. The G-input is forced enabled for one more APB clock cycle, and the D-input to zero. In all other cases, the latch output (OUT) is refreshed as shown in [Table 38-4 D-Latch Characteristics](#) on page 915.

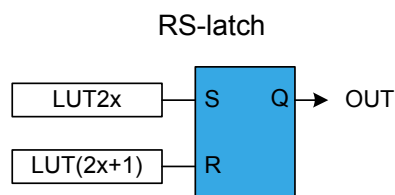
**Table 38-4. D-Latch Characteristics**

G	D	OUT
0	X	Hold state (no change)
1	0	Clear
1	1	Set

#### RS Latch (RS)

When this configuration is selected, the S-input is driven by the even LUT output (LUT2x), and the R-input is driven by the odd LUT output (LUT2x+1), as shown in [Figure 38-17 RS-Latch](#) on page 916.

**Figure 38-17. RS-Latch**



When the even LUT is disabled (`LUTCTRL2x.ENABLE=0`), the latch output will be cleared. The R-input is forced enabled for one more APB clock cycle and S-input to zero. In all other cases, the latch output (OUT) is refreshed as shown in [Table 38-5 RS-latch Characteristics](#) on page 916.

**Table 38-5. RS-latch Characteristics**

S	R	OUT
0	0	Hold state (no change)
0	1	Clear
1	0	Set
1	1	Forbidden state

### 38.6.3. Events

The CCL can generate the following output events:

- LUTOUTx: Lookup Table Output Value

Writing a '1' to the LUT Control Event Output Enable bit (`LUTCTRL.LUTEO`) enables the corresponding output event. Writing a '0' to this bit disables the corresponding output event. Refer to *EVSYS – Event System* for details on configuration.

The CCL can take the following actions on an input event:

- INx: The event is used as input for the TRUTH table. For further details refer to [Events](#) on page 906.

Writing a '1' to the LUT Control Event Input Enable bit (`LUTCTRL.LUTEI`) enables the corresponding action on input event. Writing a '0' to this bit disables the corresponding action on input event. Refer to *EVSYS – Event System* for details on configuration.

#### Related Links

[EVSYS – Event System](#) on page 474

### 38.6.4. Sleep Mode Operation

When using the GCLK\_CCL internal clocking, writing the Run In Standby bit in the Control register (`CTRL.RUNSTDBY`) to '1' will allow GCLK\_CCL to be enabled in all sleep modes.

If `CTRL.RUNSTDBY=0`, the GCLK\_CCL will be disabled. If the Filter, Edge Detector or Sequential logic are enabled, the LUT output will be forced to zero in STANDBY mode. In all other cases, the TRUTH table decoder will continue operation and the LUT output will be refreshed accordingly.



## 38.7. Register Summary

Offset	Name	Bit pos.							
0x00	CTRL	7:0		RUNSTDBY				ENABLE	SWRST
0x01	Reserved	7:0							
0x02	Reserved	7:0							
0x03	Reserved	7:0							
0x04	SEQCTRL0	7:0						SEQSEL[3:0]	
0x05	SEQCTRL1	7:0						SEQSEL[3:0]	
0x06	SEQCTRL2	7:0						SEQSEL[3:0]	
0x07	SEQCTRL3	7:0						SEQSEL[3:0]	
0x08	LUTCTRL0	7:0	EDGESEL		FILTSEL[1:0]			ENABLE	
0x09		15:8		INSEL1[3:0]				INSEL0[3:0]	
0x0A		23:16		LUTEO	LUTEI	INVEI		INSEL2[3:0]	
0x0B		31:24		TRUTH[7:0]					
0x0C	LUTCTRL1	7:0	EDGESEL		FILTSEL[1:0]			ENABLE	
0x0D		15:8		INSEL1[3:0]				INSEL0[3:0]	
0x0E		23:16		LUTEO	LUTEI	INVEI		INSEL2[3:0]	
0x0F		31:24		TRUTH[7:0]					
0x10	LUTCTRL2	7:0	EDGESEL		FILTSEL[1:0]			ENABLE	
0x11		15:8		INSEL1[3:0]				INSEL0[3:0]	
0x12		23:16		LUTEO	LUTEI	INVEI		INSEL2[3:0]	
0x13		31:24		TRUTH[7:0]					
0x14	LUTCTRL4	7:0	EDGESEL		FILTSEL[1:0]			ENABLE	
0x15		15:8		INSEL1[3:0]				INSEL0[3:0]	
0x16		23:16		LUTEO	LUTEI	INVEI		INSEL2[3:0]	
0x17		31:24		TRUTH[7:0]					
0x18	LUTCTRL4	7:0	EDGESEL		FILTSEL[1:0]			ENABLE	
0x19		15:8		INSEL1[3:0]				INSEL0[3:0]	
0x1A		23:16		LUTEO	LUTEI	INVEI		INSEL2[3:0]	
0x1B		31:24		TRUTH[7:0]					
0x1C	LUTCTRL5	7:0	EDGESEL		FILTSEL[1:0]			ENABLE	
0x1D		15:8		INSEL1[3:0]				INSEL0[3:0]	
0x1E		23:16		LUTEO	LUTEI	INVEI		INSEL2[3:0]	
0x1F		31:24		TRUTH[7:0]					
0x20	LUTCTRL6	7:0	EDGESEL		FILTSEL[1:0]			ENABLE	
0x21		15:8		INSEL1[3:0]				INSEL0[3:0]	
0x22		23:16		LUTEO	LUTEI	INVEI		INSEL2[3:0]	
0x23		31:24		TRUTH[7:0]					
0x24	LUTCTRL7	7:0	EDGESEL		FILTSEL[1:0]			ENABLE	
0x25		15:8		INSEL1[3:0]				INSEL0[3:0]	
0x26		23:16		LUTEO	LUTEI	INVEI		INSEL2[3:0]	
0x27		31:24		TRUTH[7:0]					

## 38.8. 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](#) on page 906.

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.

### 38.8.1. Control

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

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

#### Bit 6 – RUNSTDBY: Run in Standby

This bit indicates if the GCLK\_CCL clock must be kept running in standby mode. The setting is ignored for configurations where the generic clock is not required. For details refer to [Sleep Mode Operation](#) on page 916.

Value	Description
0	Generic clock is not required in standby sleep mode.
1	Generic clock is required in standby sleep mode.

#### Bit 1 – ENABLE: Enable

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

#### Bit 0 – SWRST: Software Reset

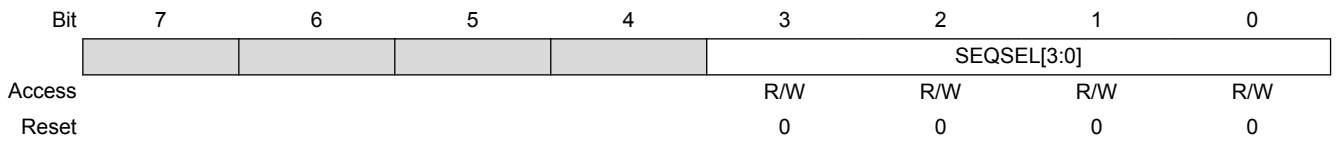
Writing a '0' to this bit has no effect.

Writing a '1' to this bit resets all registers in the CCL to their initial state.

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

### 38.8.2. Sequential Control x

**Name:** SEQCTRLx  
**Offset:** 0x04 + x (x = 0,...,3)  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected



#### Bits 3:0 – SEQSEL[3:0]: Sequential Selection

These bits select the sequential configuration:

Sequential Selection

Value	Name	Description
0x0	DISABLE	Sequential logic is disabled
0x1	DFF	D flip flop
0x2	JK	JK flip flop
0x3	LATCH	D latch
0x4	RS	RS latch
0x5 - 0xF		Reserved

### 38.8.3. LUT Control x

**Name:** LUTCTRLx

**Offset:** 0x08 + 4\*x (x = 0,...,7)

**Reset:** 0x00000000

**Property:** PAC Write-Protection, Enable-Protected (except LUTEN)

Bit	31	30	29	28	27	26	25	24
	TRUTH[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
		LUTEO	LUTEI	INVEI	INSEL2[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	15	14	13	12	11	10	9	8
	INSEL1[3:0]				INSEL0[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	7	6	5	4	3	2	1	0
	EDGESEL		FILTSEL[1:0]				ENABLE	
Access	R/W		R/W	R/W			R/W	
Reset	0		0	0			0	

#### Bits 31:24 – TRUTH[7:0]: Truth Table

These bits define the value of truth logic as a function of inputs IN[2:0].

#### Bit 22 – LUTEO: LUT Event Output Enable

Value	Description
0	LUT event output is disabled.
1	LUT event output is enabled.

#### Bit 21 – LUTEI: LUT Event Input Enable

Value	Description
0	LUT incoming event is disabled.
1	LUT incoming event is enabled.

#### Bit 20 – INVEI: Inverted Event Input Enable

Value	Description
0	Incoming event is not inverted.
1	Incoming event is inverted.

### Bit 7 – EDGESEL: Edge Selection

Value	Description
0	Edge detector is disabled.
1	Edge detector is enabled.

### Bits 5:4 – FILTSEL[1:0]: Filter Selection

These bits select the LUT output filter options:

Filter Selection

Value	Name	Description
0x0	DISABLE	Filter disabled
0x1	SYNCH	Synchronizer enabled
0x2	FILTER	Filter enabled
0x3	Reserved	

### Bit 1 – ENABLE: LUT Enable

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

### Bits 19:16,15:12,11:8 – INSELx: LUT Input x Source Selection

These bits select the LUT input x source:

Value	Name	Description
0x0	MASK	Masked input
0x1	FEEDBACK	Feedback input source
0x2	LINK	Linked LUT input source
0x3	EVENT	Event input source
0x4	IO	I/O pin input source
0x5	AC	AC input source
0x6	TC	TC input source
0x7	ALTTC	Alternative TC input source
0x8	TCC	TCC input source
0x9	SERCOM	SERCOM input source
0xA - 0xF	Reserved	

## 39. ADC – Analog-to-Digital Converter

### 39.1. Overview

The Analog-to-Digital Converter (ADC) converts analog signals to digital values. The ADC has up to 12-bit resolution, and is capable of a sampling rate of up to 1MSPS. The input selection is flexible, and both differential and single-ended measurements can be performed. In addition, several internal signal inputs are available. The ADC can provide both signed and unsigned results.

ADC measurements can be started by either application software or an incoming event from another peripheral in the device. ADC measurements can be started with predictable timing, and without software intervention.

Both internal and external reference voltages can be used.

An integrated temperature sensor is available for use with the ADC. The bandgap voltage as well as the scaled I/O and core voltages can also be measured by the ADC.

The ADC has a compare function for accurate monitoring of user-defined thresholds, with minimum software intervention required.

The ADC can be configured for 8-, 10- or 12-bit results. 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.

This device has two ADC instances. The two inputs can be sampled simultaneously, as each ADC includes sample and hold circuits.

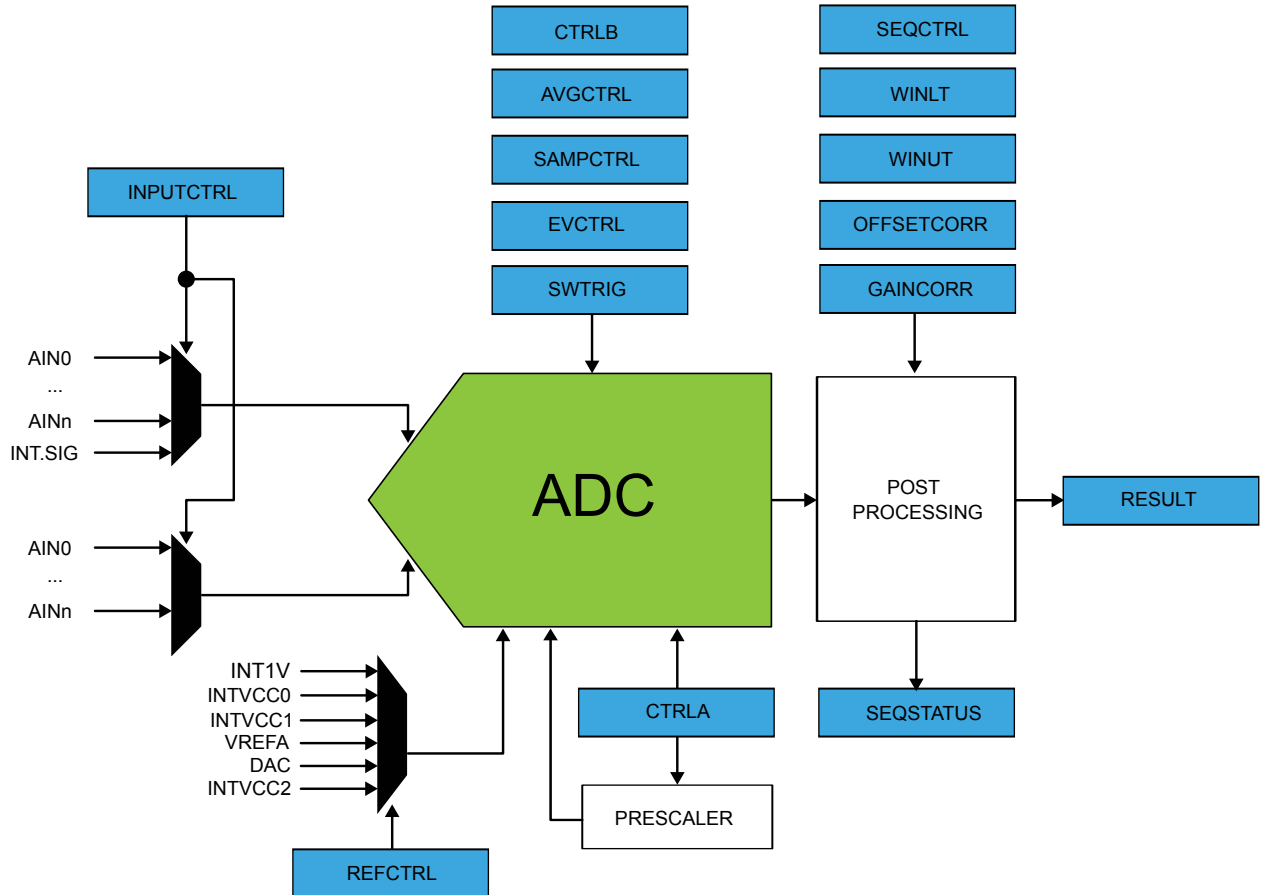
### 39.2. Features

- Two Analog to Digital Converters (ADC)
- 8-, 10- or 12-bit resolution
- Up to 1,000,000 samples per second (1 MSPS)
- Differential and single-ended inputs
  - Up to 12 analog inputs per ADC (20 unique channels total)  
16 positive and 7 negative, including internal and external
- Internal inputs:
  - Bandgap voltage
  - Scaled core supply
  - Scaled I/O supply
  - DAC
- Single, continuous and sequencing options
- Windowing monitor with selectable channel
- Conversion range:  $V_{ref} = [2.0V \text{ to } VDD_{ANA}]$
- Built-in internal reference and external reference options
- Event-triggered conversion for accurate timing (one event input)
- Optional DMA transfer of conversion settings or result
- Hardware gain and offset compensation
- Averaging and oversampling with decimation to support up to 16-bit result

- Selectable sampling time
- Flexible Power / Throughput rate management

### 39.3. Block Diagram

Figure 39-1. ADC Block Diagram



### 39.4. Signal Description

Signal	Description	Type
VREFA	Analog input	External reference voltage A
AIN[11..0]	Analog input	Analog input channels

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

#### Related Links

[Configuration Summary](#) on page 14

[I/O Multiplexing and Considerations](#) on page 27

### 39.5. Product Dependencies

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



### 39.5.1. 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: IO Pin Controller](#) on page 444

### 39.5.2. 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](#) on page 162

### 39.5.3. Clocks

The ADC bus clocks (CLK\_APB\_ADCx) can be enabled in the Main Clock, which also defines the default state.

Each ADC requires a generic clock (GCLK\_ADCx). 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](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

### 39.5.4. 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](#) on page 334

### 39.5.5. 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](#) on page 41

### 39.5.6. Events

The events are connected to the Event System.

#### Related Links

[EVSYS – Event System](#) on page 474

### 39.5.7. 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. Refer to [DBGCTRL](#) on page 961 for details.

### 39.5.8. 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](#) on page 46

### 39.5.9. Analog Connections

I/O-pins (AINx), as well as the VREFA reference voltage pins are analog inputs to the ADC.

### 39.5.10. 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](#) on page 37

## 39.6. Functional Description

### 39.6.1. 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, see [Conversion Timing and Sampling Rate](#) on page 928.

The ADC has an oversampling with decimation option that can extend the resolution to 16 bits. The input values can be either internal (e.g., internal temperature sensor) or external (connected I/O pins). The user can also configure whether the conversion should be single-ended or differential.

### 39.6.2. Basic Operation

#### 39.6.2.1. Initialization

The following registers are enable-protected, meaning that they can only be written when the ADC is disabled (CTRLA.ENABLE=0):

- Control B register (CTRLB)
- Reference Control register (REFCTRL)
- Event Control register (EVCTRL)
- Calibration register (CALIB)

Enable-protection is denoted by the "Enable-Protected" property in the register description.

#### 39.6.2.2. Enabling, Disabling and Resetting

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. Refer to [CTRLA](#) on page 940 for details.

### 39.6.2.3. 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 result of the conversion is stored in the Result register (RESULT) overwriting the result from the previous conversion.

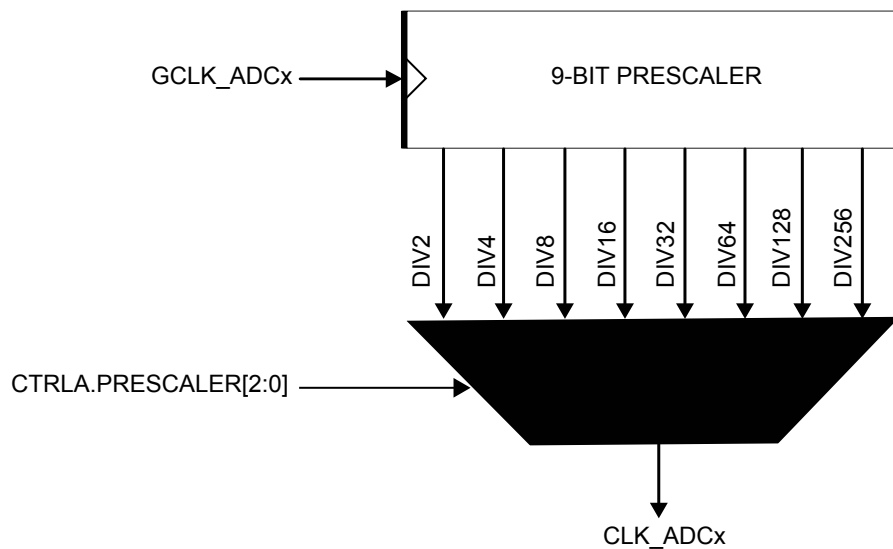
To avoid data loss if more than one channel is enabled, the conversion result must be read as soon as it is available (INTFLAG.RESRDY). Failing to do so will result in an overrun error condition, indicated by the OVERRUN bit in the Interrupt Flag Status and Clear register (INTFLAG.OVERRUN).

To enable one of the available interrupts sources, the corresponding bit in the Interrupt Enable Set register (INTENSET) must be written to '1'.

### 39.6.2.4. Prescaler Selection

The ADC is clocked by GCLK\_ADCx. There is also a prescaler in the ADC to enable conversion at lower clock rates. Refer to [CTRLA](#) for details on prescaler settings. Refer to [Conversion Timing and Sampling Rate](#) on page 928 for details on timing and sampling rate.

Figure 39-2. ADC Prescaler



**Note:** The minimum prescaling factor is DIV2.

### 39.6.2.5. Reference Configuration

The ADC has various reference configuration options. By default, the internal bandgap voltage reference is selected. The REFSEL value in Reference Control register (REFCTRL) determines which reference will be selected. Based on customer application requirements, the external or internal reference can be selected. Two external references are available. The supply accepted on these pins is from 2.0V to VDD<sub>ANA</sub>. Four internal inputs are also available. Refer to [REFCTRL](#) for further details on available selections.

### 39.6.2.6. 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 C register (CTRLC.RESEL). By default, the ADC resolution is set to 12 bits. The resolution affects the propagation delay, see also [Conversion Timing and Sampling Rate](#) on page 928.

### 39.6.2.7. Differential and Single-Ended Conversions

The ADC has two conversion options: differential and single-ended:

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.

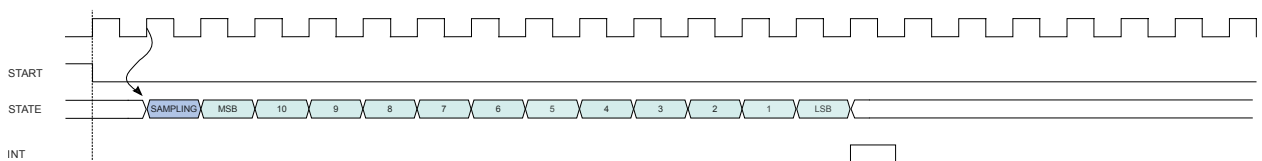
If the positive input may go below the negative input, the differential mode should be used in order to get correct results.

The differential mode is enabled by setting DIFFMODE bit in the Control C register (CTRLC.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.

### 39.6.2.8. Conversion Timing and Sampling Rate

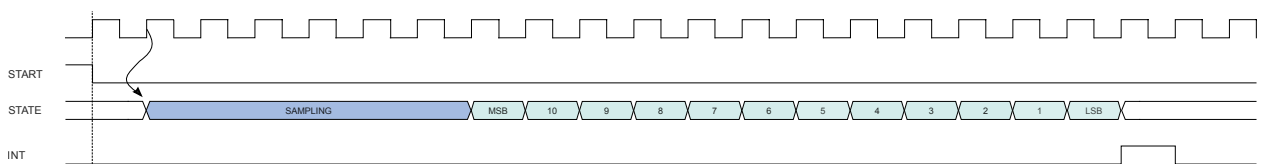
The following figure shows the ADC timing for one single conversion. A conversion starts after the software or event start are synchronized with the GCLK\_ADCx clock. The input channel is sampled in the first half CLK\_ADCx period.

**Figure 39-3. ADC Timing for One Conversion in 12-bit Resolution**



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 with sampling time increased to six CLK\_ADC cycles.

**Figure 39-4. ADC Timing for One Conversion with Increased Sampling Time, 12-bit**



The ADC provides also offset compensation, see the following figure. The offset compensation is enabled by the Offset Compensation bit in the Sampling Control register (SAMPCTRL.OFFCOMP).

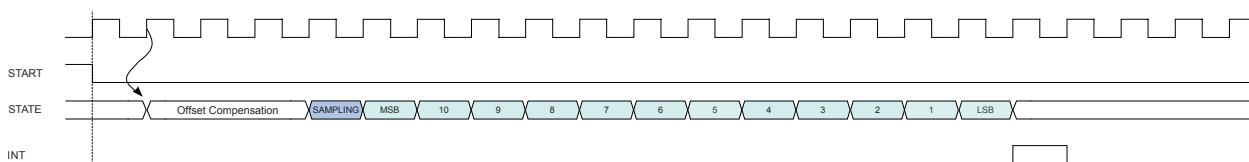
**Note:** If offset compensation is used, the sampling time must be set to one cycle of CLK\_ADCx.

In free running mode, the sampling rate  $R_S$  is calculated by

$$R_S = f_{CLK\_ADC} / (n_{SAMPLING} + n_{OFFCOMP} + n_{DATA})$$

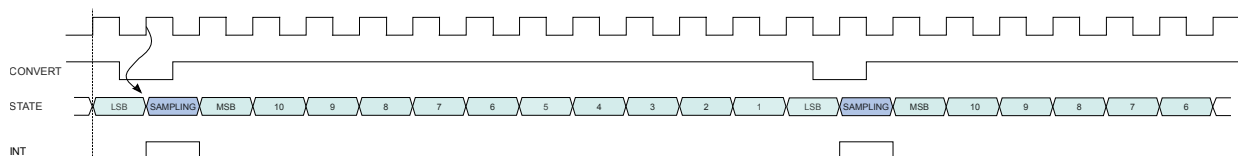
Here,  $n_{SAMPLING}$  is the sampling duration in CLK\_ADC cycles,  $n_{OFFCOMP}$  is the offset compensation duration in clock cycles, and  $n_{DATA}$  is the bit resolution.  $f_{CLK\_ADC}$  is the ADC clock frequency from the internal prescaler:  $f_{CLK\_ADC} = f_{GCLK\_ADC} / 2^{(1 + CTRLA.PRESCALER)}$

**Figure 39-5. ADC Timing for One Conversion with Offset Compensation, 12-bit**

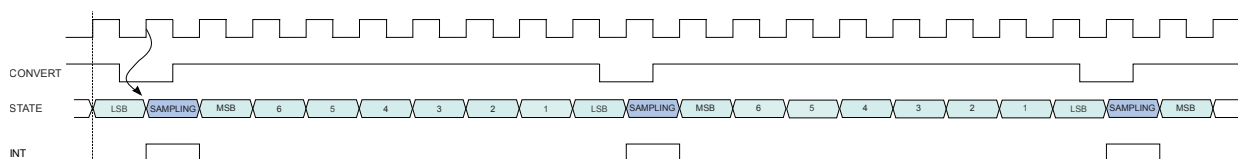


The impact of resolution on the sampling rate is seen in the next two figures, where free-running sampling in 12-bit and 8-bit resolution are compared.

**Figure 39-6. ADC Timing for Free Running in 12-bit Resolution**



**Figure 39-7. ADC Timing for Free Running in 8-bit Resolution**



The propagation delay of an ADC measurement is given by:

$$\text{PropagationDelay} = \frac{1 + \text{Resolution}}{f_{\text{ADC}}}$$

**Example.** In order to obtain 1MSPS in 12-bit resolution with a sampling time length of four CLK\_ADC cycles,  $f_{\text{CLK\_ADC}}$  must be  $1\text{MSPS} * (4 + 12) = 16\text{MHz}$ . As the minimal division factor of the prescaler is 2, GCLK\_ADC must be 32MHz.

### 39.6.2.9. Accumulation

The result from multiple consecutive conversions can be accumulated. The number of samples to be accumulated is specified by the Sample Number 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 C register (CTRLC.RESSEL) must be set.

**Table 39-1. Accumulation**

Number of Accumulated Samples	AVGCTRL.SAMPLENUM	Number of Automatic Right Shifts	Final Result Precision	Automatic Division Factor
1	0x0	0	12 bits	0
2	0x1	0	13 bits	0

Number of Accumulated Samples	AVGCTRL.SAMPLENUM	Number of Automatic Right Shifts	Final Result Precision	Automatic Division Factor
4	0x2	0	14 bits	0
8	0x3	0	15 bits	0
16	0x4	0	16 bits	0
32	0x5	1	16 bits	2
64	0x6	2	16 bits	4
128	0x7	3	16 bits	8
256	0x8	4	16 bits	16
512	0x9	5	16 bits	32
1024	0xA	6	16 bits	64
Reserved	0xB –0xF		12 bits	0

### 39.6.2.10. 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](#) on page 929, 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 as shown in [Table 39-2 Averaging](#) on page 930.

The division is obtained by a combination of the automatic right shift described above, and an additional right shift that must be specified by writing to the Adjusting Result/Division Coefficient field in AVGCTRL (AVGCTRL.ADJRES), as described in [Table 39-2 Averaging](#) on page 930.

**Note:** To perform the averaging of two or more samples, the Conversion Result Resolution field in the Control C register (CTRLC.RESSEL) must be set.

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 39-2. 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

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
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

### 39.6.2.11. 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,  $4n$  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 39-3. 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

### 39.6.2.12. Automatic Sequences

The ADC has the ability to automatically sequence a series of conversions. This means that each time the ADC receives a start-of-conversion request, it can perform multiple conversions automatically. All of the 32 positive inputs can be included in a sequence by writing to corresponding bits in the Sequence Control register (SEQCTRL). The order of the conversion in a sequence is the lower positive MUX selection to upper positive MUX (AIN0, AIN1, AIN2 ...). In differential mode, the negative inputs selected by MUXNEG field, will be used for the entire sequence.

When a sequence starts, the Sequence Busy status bit in Sequence Status register (SEQSTATUS.SEQBUSY) will be set. When the sequence is complete, the Sequence Busy status bit will be cleared.

Each time a conversion is completed, the Sequence State bit in Sequence Status register (SEQSTATUS.SEQSTATE) will store the input number from which the conversion is done. The result will be stored in the RESULT register, and the Result Ready Interrupt Flag (INTFLAG.RESRDY) is set.

If additional inputs must be scanned, the ADC will automatically start a new conversion on the next input present in the sequence list.

Note that if SEQCTRL register has no bits set to '1', the conversion is done with the selected MUXPOS input.

### 39.6.2.13. 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 Control C register (CTRLC.WINMODE). 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 C register (CTRLC.RESSEL). This means that for example 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.

### 39.6.2.14. 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).

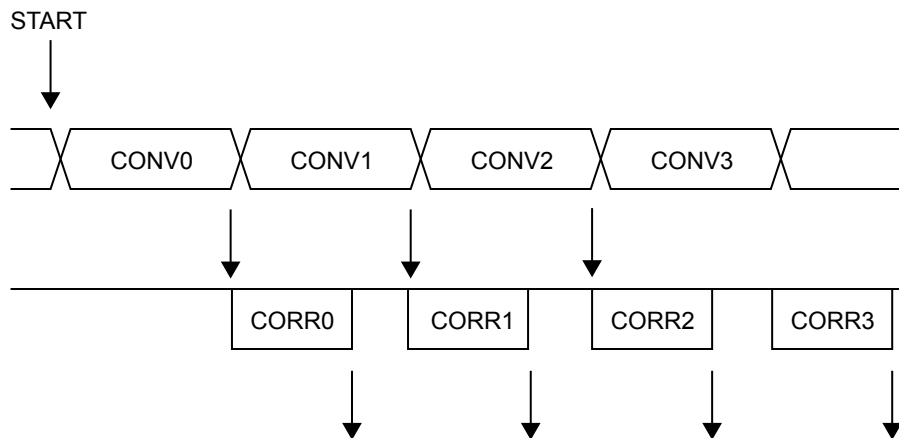
To correct these two errors, the Digital Correction Logic Enabled bit in the Control C register (CTRLC.CORREN) must be set.

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 39-8. ADC Timing Correction Enabled



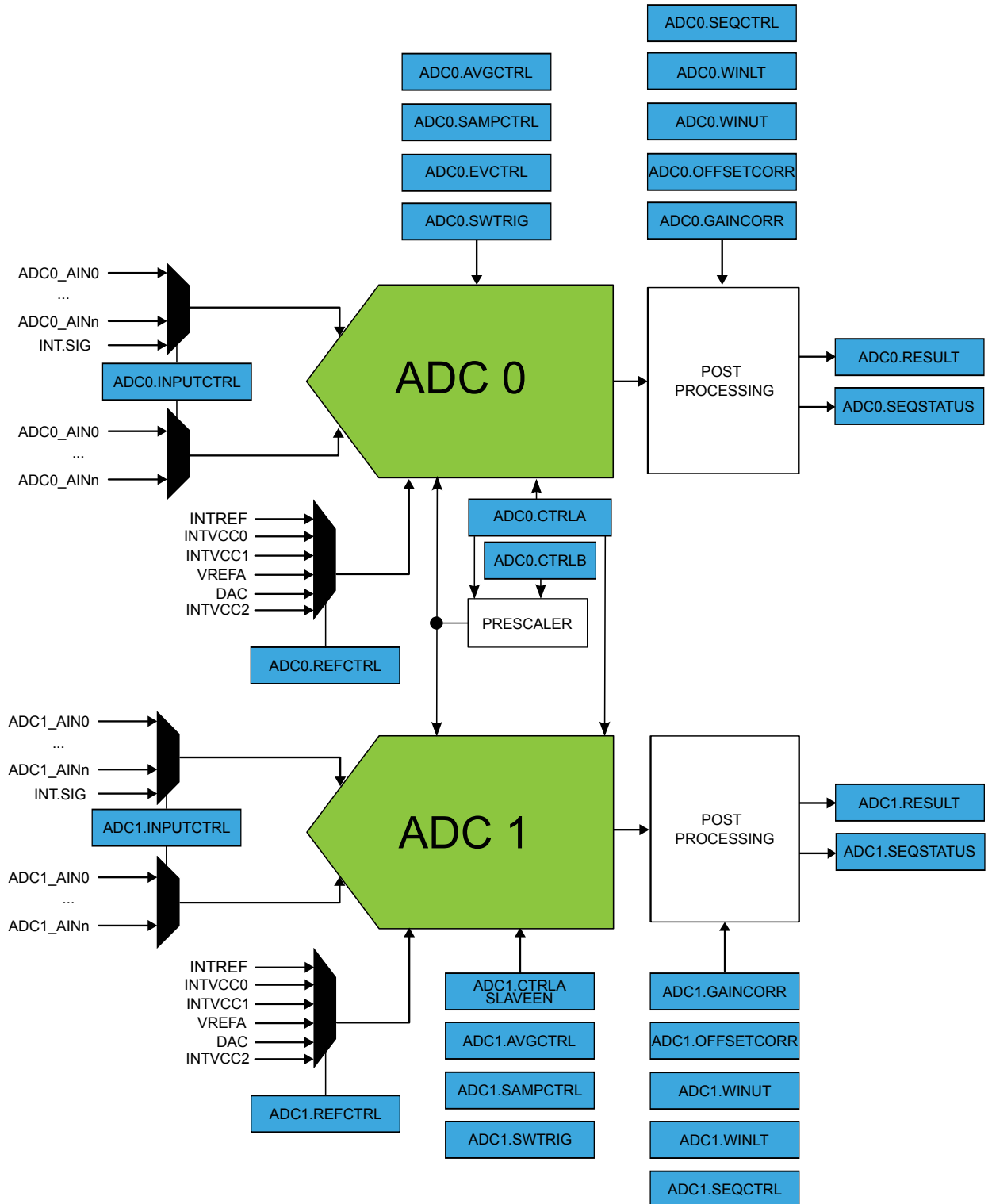


### 39.6.3. Additional Features

#### 39.6.3.1. Master - Slave Operation

The master - slave operation is available only on devices with two ADC instances. The ADC1 will be enabled as a slave of ADC0 instance when writing a one to the Slave Enable bit in Control A register of the ADC1 instance (ADC1.CTRLA.SLAVEEN). When enabled, GCLK\_ADC0 clock and ADC0 controls are internally routed to the ADC1 instance.

Figure 39-9. ADC Master - Slave Block Diagram



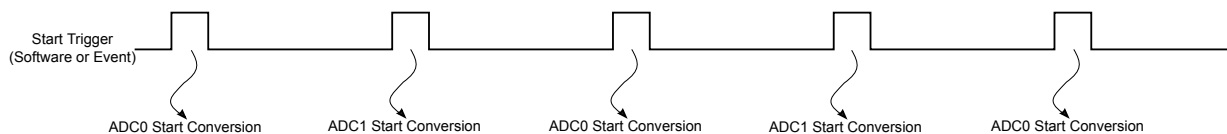
In this mode of operation, the slave ADC is enabled by accessing the CTRLA register of master ADC. In the same way, the master ADC event inputs will be automatically routed to the slave ADC, meaning that the input events configuration must be done in the master ADC (ADC0.EVCTRL).

ADC measurements can be started simultaneously on both ADC's or interleaved. The trigger mode selection is available in the master ADC Control C register (ADC0.CTRL.C.DUALSEL).

To restart an interleaved sequence, the user can apply different options:

- Flush the master ADC (ADC0.SWTRIG.FLUSH = 1)
- Disable/re-enable the master ADC (ADC0.CTRLA.ENABLE)
- Reset and reconfigure master ADC (ADC0.CTRLA.SWRST = 1)

**Figure 39-10. Interleaved Dual-Mode Trigger Selection**



### 39.6.3.2. Rail-to-Rail Operation

The accuracy of the ADC is highest when the input common mode voltage ( $V_{CMIN}$ ) is close to  $V_{REF}/2$ . To enable a full range of common mode voltages (rail-to-rail operation), the Rail-to-Rail bit in the Control C register (CTRLC.R2R) should be written to one. Rail-to-rail operation requires a sampling period of four cycles. This is achieved by enabling offset compensation (SAMPCTRL.OFFCOMP = 1). Rail-to-rail operation should not be used when offset compensation is disabled.

### 39.6.4. 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.

### 39.6.5. Interrupts

The ADC has the following interrupt sources:

- Result Conversion Ready: RESRDY
- Window Monitor: WINMON
- Overrun: OVERRUN

These interrupts are asynchronous wake-up sources. See *Sleep Mode Controller* 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 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. See [INTFLAG](#) on page 948 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 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](#) on page 41

[Sleep Mode Controller](#) on page 164

### 39.6.6. Events

The ADC can generate the following output events:

- Result Ready (RESRDY): Generated when the conversion is complete and the result is available. Refer to [EVCTRL](#) on page 944 for details.
- Window Monitor (WINMON): Generated when the window monitor condition match. Refer to [CTRLC](#) on page 952 for details.

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 ADC can take the following actions on an input event:

- Start conversion (START): Start a conversion. Refer to [SWTRIG](#) on page 960 for details.
- Conversion flush (FLUSH): Flush the conversion. Refer to [SWTRIG](#) on page 960 for details.

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.

The ADC uses only asynchronous events, so the asynchronous Event System channel path must be configured. By default, the ADC will detect a rising edge on the incoming event. If the ADC action must be performed on the falling edge of the incoming event, the event line must be inverted first. This is done by setting the corresponding Event Invert Enable bit in Event Control register (EVCTRL.xINV=1).

**Note:** If several events are connected to the ADC, the enabled action will be taken on any of the incoming events. If FLUSH and START events are available at the same time, the FLUSH event has priority.

#### Related Links

[EVSYS – Event System](#) on page 474

### 39.6.7. Sleep Mode Operation

The ONDEMAND and RUNSTDBY bits in the Control A register (CTRLA) control the behavior of the ADC during standby sleep mode, in cases where the ADC is enabled (CTRLA.ENABLE = 1). For further details on available options, refer to [Table 39-4 ADC Sleep Behavior](#) on page 936.

**Note:** When CTRLA.ONDEMAND=1, the analog block is powered-off when the conversion is complete. When a start request is detected, the system returns from sleep and starts a new conversion after the start-up time delay.

**Table 39-4. ADC Sleep Behavior**

CTRLA.RUNSTDBY	CTRLA.ONDEMAND	CTRLA.ENABLE	Description
x	x	0	Disabled
0	0	1	Run in all sleep modes except STANDBY.
0	1	1	Run in all sleep modes on request, except STANDBY.
1	0	1	Run in all sleep modes.
1	1	1	Run in all sleep modes on request.

### 39.6.8. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

The following bits are synchronized when written:

- Software Reset bit in Control A register (CTRLA.SWRST)
- Enable bit in Control A register (CTRLA.ENABLE)

The following registers are synchronized when written:

- Input Control register (INPUTCTRL)
- Control C register (CTRLC)
- Average control register (AVGCTRL)
- Sampling time control register (SAMPCTRL)
- Window Monitor Lower Threshold register (WINLT)
- Window Monitor Upper Threshold register (WINUT)
- Gain correction register (GAINCORR)
- Offset Correction register (OFFSETCORR)
- Software Trigger register (SWTRIG)

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

#### Related Links

[Register Synchronization](#) on page 110

## 39.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	SLAVEEN				ENABLE	SWRST
0x01	CTRLB	7:0						PRESCALER[2:0]		
0x02	REFCTRL	7:0	REFCOMP					REFSEL[3:0]		
0x03	EVCTRL	7:0			WINMONEO	RESRDYEO	STARTINV	FLUSHINV	STARTEI	FLUSHEI
0x04	INTENCLR	7:0						WINMON	OVERRUN	RESRDY
0x05	INTENSET	7:0						WINMON	OVERRUN	RESRDY
0x06	INTFLAG	7:0						WINMON	OVERRUN	RESRDY
0x07	SEQSTATUS	7:0	SEQBUSY					SEQSTATE[4:0]		
0x08	INPUTCTRL	7:0						MUXPOS[4:0]		
0x09		15:8						MUXNEG[4:0]		
0x0A	CTRLC	7:0	R2R		RESSEL[1:0]	CORREN	FREERUN	LEFTADJ	DIFFMODE	
0x0B		15:8			DUALSEL[1:0]			WINMODE[2:0]		
0x0C	AVGCTRL	7:0		ADJRES[2:0]			SAMPLENUM[3:0]			
0x0D	SAMPCTRL	7:0	OFFCOMP				SAMPLLEN[5:0]			
0x0E	WINLT	7:0					WINLT[7:0]			
0x0F		15:8					WINLT[15:8]			
0x10	WINUT	7:0					WINUT[7:0]			
0x11		15:8					WINUT[15:8]			
0x12	GAINCORR	7:0					GAINCORR[7:0]			
0x13		15:8					GAINCORR[11:8]			
0x14	OFFSETCORR	7:0					OFFSETCORR[7:0]			
0x15		15:8					OFFSETCORR[11:8]			
0x16	...									
0x17	Reserved									
0x18	SWTRIG	7:0							START	FLUSH
0x19	...									
0x1A	Reserved									
0x1B	...									
0x1C	DBGCTRL	7:0								DBGRUN
0x1D	...									
0x1E	Reserved									
0x1F	...									
0x20	SYNCBUSY	7:0	WINUT	WINLT	SAMPCTRL	AVGCTRL	CTRLC	INPUTCTRL	ENABLE	SWRST
0x21		15:8						SWTRIG	OFFSETCORR	GAINCORR
0x22	...									
0x23	Reserved									
0x24	RESULT	7:0	RESULT[7:0]							
0x25		15:8	RESULT[15:8]							
0x26	...									
0x27	Reserved									

Offset	Name	Bit Pos.								
0x28	SEQCTRL	7:0	SEQEN7	SEQEN6	SEQEN5	SEQEN4	SEQEN3	SEQEN2	SEQEN1	SEQEN0
0x29		15:8	SEQEN15	SEQEN14	SEQEN13	SEQEN12	SEQEN11	SEQEN10	SEQEN9	SEQEN8
0x2A		23:16	SEQEN23	SEQEN22	SEQEN21	SEQEN20	SEQEN19	SEQEN18	SEQEN17	SEQEN16
0x2B		31:24	SEQEN31	SEQEN30	SEQEN29	SEQEN28	SEQEN27	SEQEN26	SEQEN25	SEQEN24
0x2C ... 0x47	Reserved									
0x48	CALIB	7:0						BIASCOMP[2:0]		
0x49		15:8						BIASREFBUF[2:0]		

### 39.8. 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](#) on page 926.

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](#) on page 937.

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.

### 39.8.1. Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY	SLAVEEN				ENABLE	SWRST
Access	R/W	R/W	R/W				R/W	R/W
Reset	0	0	0				0	0

#### Bit 7 – ONDEMAND: On Demand Control

The On Demand operation mode allows the ADC to be enabled or disabled, depending on other peripheral requests.

In On Demand operation mode, i.e., if the ONDEMAND bit has been previously set, the ADC will only be running when requested by a peripheral. If there is no peripheral requesting the ADC will be in a disable state.

If On Demand is disabled the ADC will always be running when enabled.

In standby sleep mode, the On Demand operation is still active if the CTRLA.RUNSTDBY bit is '1'. If CTRLA.RUNSTDBY is '0', the ADC is disabled.

This bit is not synchronized.

For the slave ADC, this bit has no effect when the SLAVEEN bit is set (CTRLA.SLAVEEN= 1). ONDEMAND bit from master ADC instance will control the On Demand operation mode.

Value	Description
0	The ADC is always on , if enabled.
1	The ADC is enabled, when a peripheral is requesting the ADC conversion. The ADC is disabled if no peripheral is requesting it.

#### Bit 6 – RUNSTDBY: Run in Standby

This bit controls how the ADC behaves during standby sleep mode.

This bit is not synchronized.

For the slave ADC, this bit has no effect when the SLAVEEN bit is set (CTRLA.SLAVEEN= 1). RUNSTDBY bit from master ADC instance will control the slave ADC operation in standby sleep mode.

Value	Description
0	The ADC is halted during standby sleep mode.
1	The ADC is not stopped in standby sleep mode. If CTRLA.ONDEMAND=1, the ADC will be running when a peripheral is requesting it. If CTRLA.ONDEMAND=0, the ADC will always be running in standby sleep mode.

#### Bit 5 – SLAVEEN: Slave Enable

This bit enables the master/slave operation and it is available only in the slave ADC instance.

This bit is not synchronized.



This bit can be set only for the slave ADC. For the master ADC, this bit is always read zero.

Value	Description
0	The master-slave operation is disabled.
1	The ADC1 is enabled as a slave of ADC0

**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 ENABLE bit in the SYNCBUSY register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

For the slave ADC, this bit has no effect when the SLAVEEN bit is set (CTRLA.SLAVEEN= 1).

Value	Description
0	The ADC is disabled.
1	The ADC 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 ADC, except DBGCTRL, to their initial state, and the ADC 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.

### 39.8.2. Control B

**Name:** CTRLB  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

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

#### Bits 2:0 – PRESCALER[2:0]: Prescaler Configuration

This field defines the ADC clock relative to the peripheral clock.

This field is not synchronized. For the slave ADC, these bits have no effect when the SLAVEEN bit is set (CTRLA.SLAVEEN= 1).

Value	Name	Description
0x0	DIV2	Peripheral clock divided by 2
0x1	DIV4	Peripheral clock divided by 4
0x2	DIV8	Peripheral clock divided by 8
0x3	DIV16	Peripheral clock divided by 16
0x4	DIV32	Peripheral clock divided by 32
0x5	DIV64	Peripheral clock divided by 64
0x6	DIV128	Peripheral clock divided by 128
0x7	DIV256	Peripheral clock divided by 256

### 39.8.3. Reference Control

**Name:** REFCTRL  
**Offset:** 0x02  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-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 gain error can be reduced 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.

Value	Name	Description
0x0	INTREF	internal bandgap reference
0x1	INTVCC0	1/1.6 VDDANA
0x2	INTVCC1	1/2 VDDANA (only for VDDANA > 4.0V)
0x3	VREFA	External reference
0x4	DAC	DAC internal output
0x5	INTVCC2	VDDANA
0x6 - 0xF		Reserved

### 39.8.4. Event Control

**Name:** EVCTRL  
**Offset:** 0x03  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
			WINMONEO	RESRDYEO	STARTINV	FLUSHINV	STARTEI	FLUSHEI
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	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 3 – STARTINV: Start Conversion Event Invert Enable

For the slave ADC, this bit has no effect when the SLAVEEN bit is set (CTRLA.SLAVEEN= 1).

Value	Description
0	Start event input source is not inverted.
1	Start event input source is inverted.

#### Bit 2 – FLUSHINV: Flush Event Invert Enable

For the slave ADC, this bit has no effect when the SLAVEEN bit is set (CTRLA.SLAVEEN= 1).

Value	Description
0	Flush event input source is not inverted.
1	Flush event input source is inverted.

#### Bit 1 – STARTEI: Start Conversion Event Input Enable

For the slave ADC, this bit has no effect when the SLAVEEN bit is set (CTRLA.SLAVEEN= 1).

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.

**Bit 0 – FLUSHEI: Flush Event Input Enable**

For the slave ADC, this bit has no effect when the SLAVEEN bit is set (CTRLA.SLAVEEN= 1).

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.

### 39.8.5. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x04

**Reset:** 0x00

**Property:** PAC Write-Protection

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

#### Bit 2 – WINMON: Window Monitor Interrupt Enable

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

Writing a '1' to this bit will clear the Window Monitor Interrupt Enable bit, which disables 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 '0' to this bit has no effect.

Writing a '1' to this bit will clear the Overrun Interrupt Enable bit, which disables 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 '0' to this bit has no effect.

Writing a '1' to this bit will clear the Result Ready Interrupt Enable bit, which disables the corresponding interrupt request.

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.

### 39.8.6. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x05

**Reset:** 0x00

**Property:** PAC Write-Protection

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

#### Bit 2 – WINMON: Window Monitor Interrupt Enable

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

Writing a '1' to this bit will set the Window Monitor Interrupt bit, which enables 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 '0' to this bit has no effect.

Writing a '1' to this bit will set the Overrun Interrupt bit, which enables 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 '0' to this bit has no effect.

Writing a '1' to this bit will set the Result Ready Interrupt bit, which enables the Result Ready interrupt.

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

### 39.8.7. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** –

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

#### Bit 2 – WINMON: Window Monitor

This flag is cleared by writing a '1' 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 '1'.

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

Writing a '1' to this bit clears the Window Monitor interrupt flag.

#### Bit 1 – OVERRUN: Overrun

This flag is cleared by writing a '1' 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=1.

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

Writing a '1' to this bit clears the Overrun interrupt flag.

#### Bit 0 – RESRDY: Result Ready

This flag is cleared by writing a '1' 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=1.

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

Writing a '1' to this bit clears the Result Ready interrupt flag.



### 39.8.8. Sequence Status

**Name:** SEQSTATUS

**Offset:** 0x07

**Reset:** 0x00

**Property:** -

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

#### Bit 7 – SEQBUSY: Sequence busy

This bit is set when the sequence start.

This bit is clear when the last conversion in a sequence is done.

#### Bits 4:0 – SEQSTATE[4:0]: Sequence State

These bit fields are the pointer of sequence. This value identifies the last conversion done in the sequence.

### 39.8.9. Input Control

**Name:** INPUTCTRL  
**Offset:** 0x08  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Write-Synchronized

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 12:8 – MUXNEG[4:0]: Negative MUX Input Selection

These bits define the MUX selection for the negative ADC input.

Value	Name	Description
0x00	AIN0	ADC AIN0 pin
0x01	AIN1	ADC AIN1 pin
0x02	AIN2	ADC AIN2 pin
0x03	AIN3	ADC AIN3 pin
0x04	AIN4	ADC AIN4 pin
0x05	AIN5	ADC AIN5 pin
0x06 - 0x17	-	Reserved
0x18	GND	Internal ground
0x19 - 0x1F	-	Reserved

#### Bits 4:0 – MUXPOS[4:0]: Positive MUX Input Selection

These bits define the MUX selection for the positive ADC input. If the internal bandgap voltage or temperature sensor input channel is selected, then the Sampling Time Length bit group in the Sampling Control register must be written with a corresponding value.

Value	Name	Description
0x00	AIN0	ADC AIN0 pin
0x01	AIN1	ADC AIN1 pin
0x02	AIN2	ADC AIN2 pin
0x03	AIN3	ADC AIN3 pin

Value	Name	Description
0x04	AIN4	ADC AIN4 pin
0x05	AIN5	ADC AIN5 pin
0x06	AIN6	ADC AIN6 pin
0x07	AIN7	ADC AIN7 pin
0x08	AIN8	ADC AIN8 pin
0x09	AIN9	ADC AIN9 pin
0x0A	AIN10	ADC AIN10 pin
0x0B	AIN11	ADC AIN11 pin
0xC - 0x17	-	Reserved
0x18	-	Reserved
0x19	BANDGAP	Bandgap Voltage
0x1A	SCALED COREVCC	1/4 Scaled Core Supply
0x1B	SCALED IOVCC	1/4 Scaled I/O Supply
0x1C	DAC	DAC Output
0x1D	-	Reserved
0x1E	-	Reserved
0x1F	-	Reserved

### 39.8.10. Control C

**Name:** CTRLC  
**Offset:** 0x0A  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
			DUALSEL[1:0]				WINMODE[2:0]	
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
	R2R		RESSEL[1:0]		CORREN	FREERUN	LEFTADJ	DIFFMODE
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

#### Bits 13:12 – DUALSEL[1:0]: Dual Mode Trigger Selection

These bits define the trigger mode. These bits are available in the master ADC and have no effect if the master-slave operation is disabled (ADC1.CTRLA.SLAVEEN=0).

Value	Name	Description
0x0	BOTH	Start event or software trigger will start a conversion on both ADCs.
0x1	INTERLEAVE	Start event or software trigger will alternately start a conversion on ADC0 and ADC1.
0x2 - 0x3	-	Reserved

#### Bits 10:8 – WINMODE[2:0]: Window Monitor Mode

These bits enable and define the window monitor mode.

Value	Name	Description
0x0	DISABLE	No window mode (default)
0x1	MODE1	RESULT > WINLT
0x2	MODE2	RESULT < WINUT
0x3	MODE3	WINLT < RESULT < WINUT
0x4	MODE4	WINUT < RESULT < WINLT
0x5 - 0x7		Reserved

#### Bit 7 – R2R: Rail-to-Rail Operation

Value	Description
0	Disable rail-to-rail operation.
1	Enable rail-to-rail operation to increase the allowable range of the input common mode voltage ( $V_{CMIN}$ ). When R2R is one, a sampling period of four cycles is required. Offset compensation (SAMPCTRL.OFFCOMP) must be written to one when using this period.

### Bits 5:4 – RESSEL[1:0]: Conversion Result Resolution

These bits define whether the ADC completes the conversion 12-, 10- or 8-bit result resolution.

Value	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

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 GAINCORR and OFFSETCORR registers. Conversion time will be increased by 13 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 in 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.

### 39.8.11. Average Control

**Name:** AVGCTRL  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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<sup>n</sup> steps.

#### Bits 3:0 – SAMPLENUM[3:0]: Number of Samples to be Collected

These bits define how many samples are added together. The result will be available in the Result register (RESULT). Note: if the result width increases, CTRLC.RESSEL must be changed.

Value	Description
0x0	1 sample
0x1	2 samples
0x2	4 samples
0x3	8 samples
0x4	16 samples
0x5	32 samples
0x6	64 samples
0x7	128 samples
0x8	256 samples
0x9	512 samples
0xA	1024 samples
0xB - 0xF	Reserved

### 39.8.12. Sampling Time Control

**Name:** SAMPCTRL  
**Offset:** 0x0D  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	OFFCOMP		SAMPLEN[5:0]					
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 – OFFCOMP: Comparator Offset Compensation Enable

Setting this bit enables the offset compensation for each sampling period to ensure low offset and immunity to temperature or voltage drift. This compensation increases the sampling time by three clock cycles.

This bit must be set to zero to validate the SAMPLEN value. It's not possible to use OFFCOMP=1 and SAMPLEN>0.

#### Bits 5:0 – SAMPLEN[5:0]: Sampling Time Length

These bits control the ADC sampling time in number of CLK\_ADC cycles, depending of the prescaler value, thus controlling the ADC input impedance. Sampling time is set according to the equation:

$$\text{Sampling time} = (\text{SAMPLEN} + 1) \cdot (\text{CLK}_{\text{ADC}})$$

### 39.8.13. Window Monitor Lower Threshold

**Name:** WINLT

**Offset:** 0x0E

**Reset:** 0x0000

**Property:** PAC Write-Protection, 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.



### 39.8.14. Window Monitor Upper Threshold

**Name:** WINUT

**Offset:** 0x10

**Reset:** 0x0000

**Property:** PAV Write-Protection, 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.

### 39.8.15. Gain Correction

**Name:** GAINCORR  
**Offset:** 0x12  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Write-Synchronized

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 CTRL.CORREN=1, 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  $\frac{1}{2} \leq \text{GAINCORR} < 2$ . GAINCORR values range from 0.1000000000 to 1.1111111111.

### 39.8.16. Offset Correction

**Name:** OFFSETCORR

**Offset:** 0x14

**Reset:** 0x0000

**Property:** PAC Write-Protection, Write-Synchronized

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 CTRL.CORREN=1, 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.

### 39.8.17. Software Trigger

**Name:** SWTRIG  
**Offset:** 0x18  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
							START	FLUSH
Access							W	W
Reset							0	0

#### Bit 1 – START: ADC Start Conversion

Writing a '1' to this bit will start a conversion or sequence. The bit is cleared by hardware when the conversion has started. Writing a '1' to this bit when it is already set has no effect.

Writing a '0' to this bit will have no effect.

#### Bit 0 – FLUSH: ADC Conversion Flush

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 is cleared until the ADC has been flushed.

After the flush, the ADC will resume where it left off; i.e., if a conversion was pending, the ADC will start a new conversion.

Writing this bit to '0' will have no effect.

### 39.8.18. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x1C  
**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.

This bit should be written only while a conversion is not ongoing.

Value	Description
0	The ADC is halted when the CPU is halted by an external debugger.
1	The ADC continues normal operation when the CPU is halted by an external debugger.

### 39.8.19. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x20  
**Reset:** 0x0000  
**Property:** -

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

Bit	7	6	5	4	3	2	1	0
	WINUT	WINLT	SAMPCTRL	AVGCTRL	CTRLC	INPUTCTRL	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bit 10 – SWTRIG: Software Trigger Synchronization Busy

This bit is cleared when the synchronization of SWTRIG register between the clock domains is complete.  
This bit is set when the synchronization of SWTRIG register between clock domains is started.

#### Bit 9 – OFFSETCORR: Offset Correction Synchronization Busy

This bit is cleared when the synchronization of OFFSETCORR register between the clock domains is complete.  
This bit is set when the synchronization of OFFSETCORR register between clock domains is started.

#### Bit 8 – GAINCORR: Gain Correction Synchronization Busy

This bit is cleared when the synchronization of GAINCORR register between the clock domains is complete.  
This bit is set when the synchronization of GAINCORR register between clock domains is started.

#### Bit 7 – WINUT: Window Monitor Lower Threshold Synchronization Busy

This bit is cleared when the synchronization of WINUT register between the clock domains is complete.  
This bit is set when the synchronization of WINUT register between clock domains is started.

#### Bit 6 – WINLT: Window Monitor Upper Threshold Synchronization Busy

This bit is cleared when the synchronization of WINLT register between the clock domains is complete.  
This bit is set when the synchronization of WINLT register between clock domains is started.

#### Bit 5 – SAMPCTRL: Sampling Time Control Synchronization Busy

This bit is cleared when the synchronization of SAMPCTRL register between the clock domains is complete.  
This bit is set when the synchronization of SAMPCTRL register between clock domains is started.

#### Bit 4 – AVGCTRL: Average Control Synchronization Busy

This bit is cleared when the synchronization of AVGCTRL register between the clock domains is complete.

This bit is set when the synchronization of AVGCTRL register between clock domains is started.

**Bit 3 – CTRLC: Control C Synchronization Busy**

This bit is cleared when the synchronization of CTRLC register between the clock domains is complete.

This bit is set when the synchronization of CTRLC register between clock domains is started.

**Bit 2 – INPUTCTRL: Input Control Synchronization Busy**

This bit is cleared when the synchronization of INPUTCTRL register between the clock domains is complete.

This bit is set when the synchronization of INPUTCTRL register between clock domains is started.

**Bit 1 – ENABLE: ENABLE Synchronization Busy**

This bit is cleared when the synchronization of ENABLE register between the clock domains is complete.

This bit is set when the synchronization of ENABLE register between clock domains is started.

**Bit 0 – SWRST: SWRST Synchronization Busy**

This bit is cleared when the synchronization of SWRST register between the clock domains is complete.

This bit is set when the synchronization of SWRST register between clock domains is started

### 39.8.20. Result

**Name:** RESULT  
**Offset:** 0x24  
**Reset:** 0x0000  
**Property:** -

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 conversion 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 CTRLC.LEFTADJ.

If the result is left-adjusted (CTRLC.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 needed; i.e., one can read only the high byte of the entire 16-bit register.

If the result is not left-adjusted (CTRLC.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.



### 39.8.21. Sequence Control

**Name:** SEQCTRL  
**Offset:** 0x28  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	SEQEN31	SEQEN30	SEQEN29	SEQEN28	SEQEN27	SEQEN26	SEQEN25	SEQEN24
Access	R/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
	SEQEN23	SEQEN22	SEQEN21	SEQEN20	SEQEN19	SEQEN18	SEQEN17	SEQEN16
Access	R/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
	SEQEN15	SEQEN14	SEQEN13	SEQEN12	SEQEN11	SEQEN10	SEQEN9	SEQEN8
Access	R/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
	SEQEN7	SEQEN6	SEQEN5	SEQEN4	SEQEN3	SEQEN2	SEQEN1	SEQEN0
Access	R/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 – SEQENn: Enable Positive Input in the Sequence

For details on available positive mux selection, refer to [INPUTCTRL.MUXENG](#).

The sequence start from the lowest input, and go to the next enabled input automatically when the conversion is done. If no bits are set the sequence is disabled.

Value	Description
0	Disable the positive input mux n selection from the sequence.
1	Enable the positive input mux n selection to the sequence.

### 39.8.22. Calibration

**Name:** CALIB  
**Offset:** 0x48  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

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

#### Bits 10:8 – BIASREFBUF[2:0]: Bias Reference Buffer Scaling

This value from production test must be loaded from the NVM software calibration row into the CALIB register by software to achieve the specified accuracy. Refer to *NVM Software Calibration Area Mapping* for further details.

The value must be copied only, and must not be changed.

#### Bits 2:0 – BIASCOMP[2:0]: Bias Comparator Scaling

This value from production test must be loaded from the NVM software calibration row into the CALIB register by software to achieve the specified accuracy. Refer to *NVM Software Calibration Area Mapping* for further details.

The value must be copied only, and must not be changed.

## 40. SDADC – Sigma-Delta Analog-to-Digital Converter

### 40.1. Overview

The Sigma-Delta Analog-to-Digital Converter (SDADC) converts analog signals to digital values. The SDADC has 24-bit resolution, and is capable of converting up to 1.5 Msp/s divided by the data over sampling ratio (OSR). The input selection is up to three differential analog channels. The SDADC provides signed 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.

The SDADC also integrates a sleep mode and a conversion sequencer. These features reduce power consumption and processor intervention.

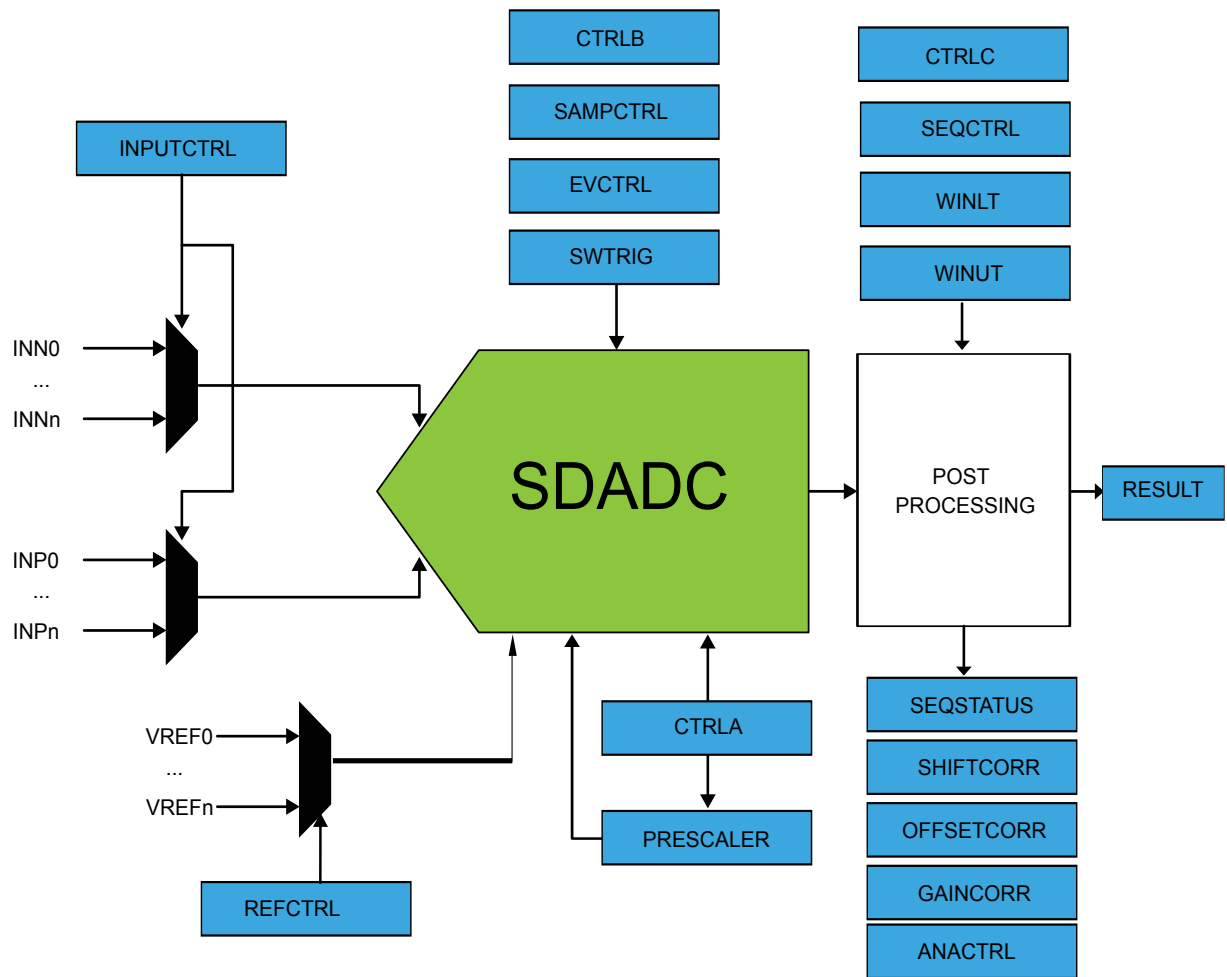
A set of reference voltages is generated internally.

### 40.2. Features

- 24-bit resolution
- Up to 1,500,000 divided by Over Sampling Ratio (OSR) samples per second
- Three analog differential inputs
  - Up to 3 external analog differential pairs.
- Conversion Range
  - 0V to  $V_{ref}$
- Event-triggered conversion (one event input)
- Optional DMA transfer of conversion settings or result
- Single, continuous and sequencing options
- Hardware gain, offset and shift compensation
- Windowing monitor

### 40.3. Block Diagram

Figure 40-1. SDADC Block Diagram.



### 40.4. Signal Description

One signal can be mapped on several pins.

Signal	Description	Type
VREF	Analog input	External reference voltage
AINN0	Analog input	Analog input channel
AINP0	Analog input	Analog input channel
AINN1	Analog input	Analog input channel
AINP1	Analog input	Analog input channel
AINN2	Analog input	Analog input channel
AINP2	Analog input	Analog input channel

#### Related Links

## 40.5. Product Dependencies

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

### 40.5.1. I/O Lines

Using the SDADC's I/O lines requires the I/O pins to be configured using the port configuration (PORT).

External Anti-alias filter must be placed in front of each SDADC input to ensure high-frequency signals to not alias into measurement bandwidth. Typical values could be  $R = 1k\Omega$ ,  $C = 3.3nF$  to  $10nF$ . Use capacitors of X5R type for DC measurement, or capacitors of COG or NPO type for AC measurement.

#### Related Links

[PORT - I/O Pin Controller](#) on page 444

### 40.5.2. Power Management

The SDADC will continue to operate in any sleep mode where the selected source clock is running. The SDADC's interrupts can be used to wake up the device from sleep modes. Events connected to the event system can trigger other operations in the system without exiting sleep modes. Refer to the Power Manager chapter for details on the different sleep modes.

### 40.5.3. Clocks

The SDADC bus clock (CLK\_SDADC\_APB) can be enabled and disabled in the Main Clock module, and the default state of CLK\_SDADC\_APB can be found in the Peripheral Clock Masking section.

A generic clock (GCLK\_SDADC) is required to generate the CLK\_SDADC to the SDADC analog module. This clock must be configured and enabled in the Generic Clock Controller (GCLK) before using the SDADC. The CLK\_SDADC is the SDADC clock connects to SDADC analog module and its range is between  $GCLK\_SDADC/2$ , if PRESCALER is 0, and  $GCLK\_SDADC/512$ , if PRESCALER is set to 255 (0xFF). Please refers to CTRLB register for more detail.

The SDADC data sampling clock CLK\_SDADC\_FS in the SDADC analog module is the  $CLK\_SDADC/4$ .

This GCLK\_SDADC is asynchronous to the bus clock (CLK\_SDADC\_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains.

#### Related Links

[Peripheral Clock Masking](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

### 40.5.4. DMA

The DMA request line is connected to the DMA Controller (DMAC). Using the SDADC DMA requests requires the DMA

Controller to be configured first. .

#### Related Links

[DMAC – Direct Memory Access Controller](#) on page 334

### 40.5.5. Interrupts

The interrupt request line is connected to the interrupt controller. Using the SDADC interrupt requires the interrupt controller to be configured first.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

#### 40.5.6. Events

The events are connected to the Event System. Refer to the Event System section for details on how to configure the Event System.

##### Related Links

[EVSYS – Event System](#) on page 474

#### 40.5.7. Debug Operation

When the CPU is halted in debug mode the SDADC will halt normal operation. The SDADC can be forced to continue operation

during debugging. Refer to [DBGCTRL](#) on page 1005 for details.

#### 40.5.8. 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.

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

#### 40.5.9. Analog Connections

I/O-pins (AINx), as well as the REF reference voltage pins are analog inputs to the SDADC.

### 40.6. Functional Description

#### 40.6.1. Principle of Operation

The Sigma-Delta Analog-to-Digital Converter (SDADC) converts analog signals to digital values. The SDADC has 24-bit resolution, and is capable of converting up to 1.5 Msps divided by the OSR data over sampling ratio. The input selection is up to three input analog channels. The SDADC provides unsigned results.

#### 40.6.2. Basic Operation

##### 40.6.2.1. Initialization

The following registers are enable-protected, meaning that they can only be written when the SDADC is disabled (CTRLA.ENABLE is zero):

- CTRLA ONEDEMAND and RUNSTDBY bits
- CTRLB
- CTRLC
- EVCTRL
- ANACTRL

Enable-protection is denoted by the Enable-Protected property in the register description.

##### 40.6.2.2. Enabling, Disabling and Resetting

The SDADC is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The SDADC is disabled by writing a zero to CTRLA.ENABLE.

The SDADC is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the ADC will be reset to their initial state, and the SDADC will be disabled. Refer to [CTRLA](#) on page 981 for details.

#### 40.6.2.3. Operation

In the most basic configuration, the SDADC sample values from the configured external sources (input ctrl register). The rate of the conversion depends on the combination of the GCLK\_SDADC frequency, the clock prescaler from CTRLB.PRESCALER and the Over Sampling Ratio from CTRLB.OSR.

To convert analog values to digital values, the SDADC needs to be initialized first, as described in [Initialization](#) on page 970 . Data conversion can be started either manually, by writing a one to the Start bit in the Software Trigger register (SWTRIG.START), or automatically, by configuring an automatic trigger to initiate the conversions. A free-running mode could be used to continuously convert an input channel. There is no need for a trigger to start the conversion. It will start automatically at the end of previous conversion.

The first valid sample starts from the third sample onward. It can skip the first few samples by programming the SKPCNT[3:0] in CTRLB register. The result of the conversion is stored in the Result register (RESULT) overwriting the result from the previous conversion.

To avoid data loss 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).

To use an interrupt handler, the corresponding bit in the Interrupt Enable Set register (INTENSET) must be written to one.

#### 40.6.2.4. Conversion Reference

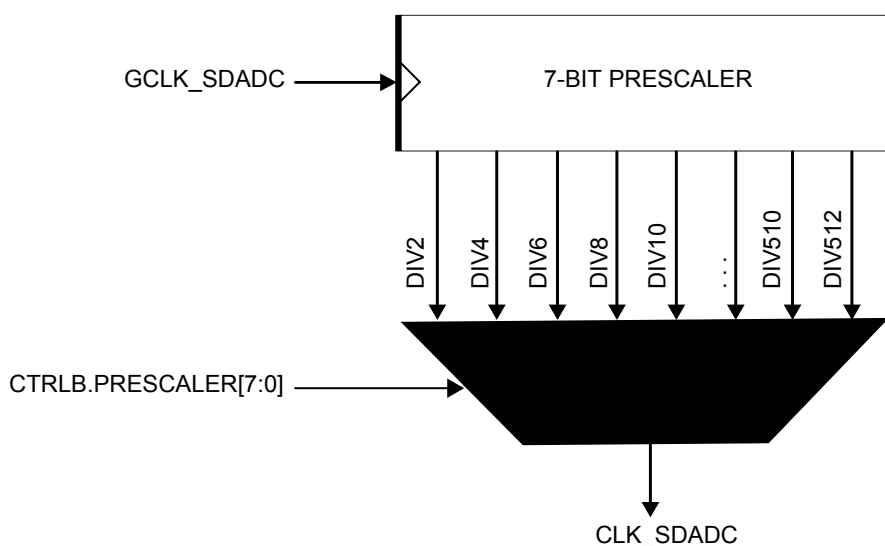
The conversion is performed on a full range between 0V and the reference voltage. Analog inputs between these voltages convert to values based on a linear conversion.

#### 40.6.2.5. Prescaler Selection

The SDADC is clocked by GCLK\_SDADC. There is also a prescaler in the SDADC to enable conversion at lower clock rates.

Refer to [CTRLB](#) on page 984 for details on prescaler settings.

Figure 40-2. SDADC Prescaler Diagram.



#### 40.6.2.6. SDADC Resolution

The SDADC provides 24-bit resolution.

#### 40.6.2.7. Automatic Sequences

The SDADC has the ability to automatically sequence a series of conversion. This means that each time the SDADC receives a start-of-conversion request, it can perform multiple conversions automatically. All of the three inputs can be included in a sequence, by writing to the Sequence Control register (SEQCTRL). The order of the conversion in a sequence is the lower positive input pair selection to upper positive input pair (AINN0, AINP0, AINN1, AINP1 ...).

When a sequence starts, the Sequence Busy status bit in Sequence Status register (SEQSTATUS.SEQBUSY) will be set to one. When the sequence is complete, the Sequence Busy status bit will be cleared.

Each time a conversion is completed, the Sequence State status in Sequence Status register (SEQSTATUS.SEQSTATE) will store the input number from which the conversion is done. The result will be stored in RESULT register and the Result Ready Interrupt Flag (INTFLAG.RESRDY) is set.

If additional inputs must be scanned, the SDADC will automatically start a new conversion on the next input present in the sequence list.

Note that if SEQCTRL register has no bits set to one, the conversion is done with the selected INPUTCTRL input.

#### 40.6.2.8. 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 writing the Window Monitor Mode bits in the Window Monitor Control register (WINCTRL.WINMODE). Threshold values must be written in the Window Monitor Lower Threshold register (WINLT) and Window Monitor Upper Threshold register (WINUT).

The INTFLAG.WINMON interrupt flag will be set if the conversion result matches the window monitor condition.



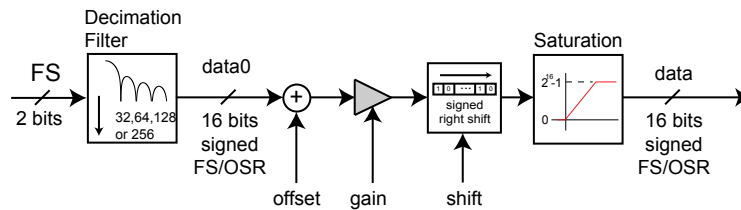
### 40.6.3. CIC (Cascaded Integrator-Comb) Decimation Filter

#### 40.6.3.1. Description

The Analog-to-Digital Converter filters and decimates the sigma-delta ADC output bitstream. Its output is defined on 16bits unsigned format with the following programmable output rates: CLK\_SDADC\_FS/64, CLK\_SDADC\_FS/128, CLK\_SDADC\_FS/256, CLK\_SDADC\_FS/512 and CLK\_SDADC\_FS/1024, where CLK\_SDADC\_FS is the sigma-delta ADC's sampling frequency: CLK\_SDADC\_FS = CLK\_SDADC\_PRESCALER/4, the reduction comes from the phase generator between the prescaler and the SDADC.

The filtering and the decimation is performed by a SINC-based filter whose zeros are placed in order to minimize aliasing effects of the decimation.

Figure 40-3. Decimation Filter Block Diagram



#### 40.6.3.2. Decimation Filter

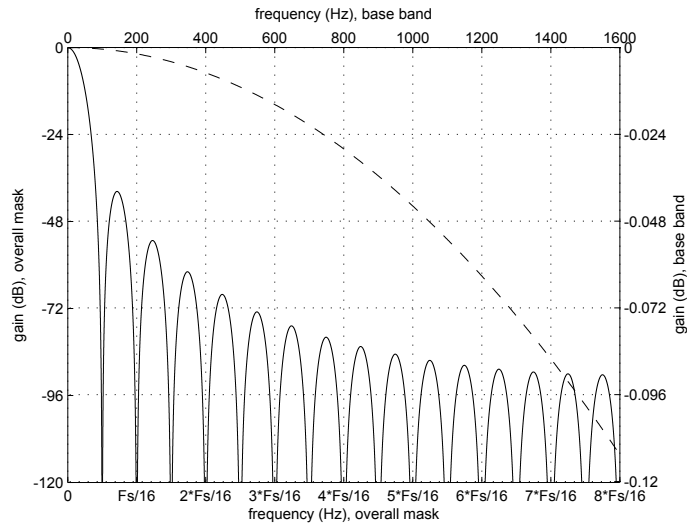
The sigma-delta architecture of the SDADC implies a filtering and a decimation of the bitstream at the output of the SDADC. The decimation filter decimates the bitstream by 64, 128 or 256, 512, 1024. To perform the decimation operation, a 3rd order SINC filter with programmable Over Sampling Ratio is implemented with the following transfer function:

$$H(z) = \frac{1}{OSR^3} \left( \sum_{i=0}^{OSR-1} z^{-i} \right)^3$$

OSR is the Over Sampling Ratio which can be modified to change the output data rate (See CTRLC for the setting of this parameter).

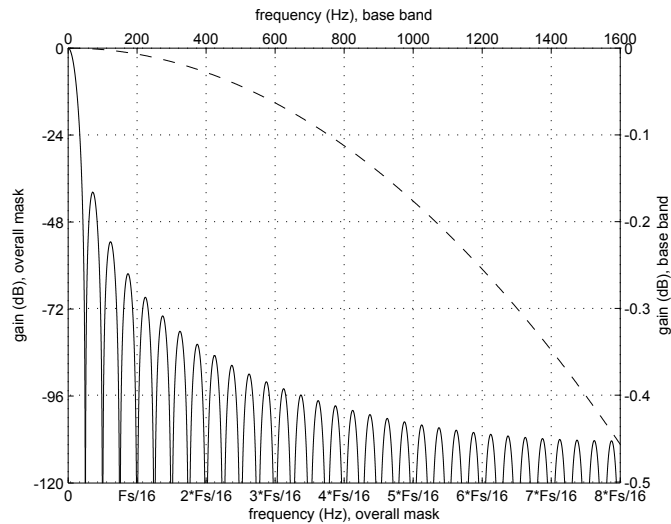
The DC gain of this filter is unity and does not depend on its OSR. However, as it generates a 3rd order zero at (CLK\_SDADC\_FS / OSR) frequency multiples, its frequency response depends on the OSR parameter. See next section for frequency plots.

**Figure 40-4. Spectral Mask of an OSR = 32, CLK\_SDADC\_FS= 1 MHz, 3rd Order Sinc Filter Overall Response (Continuous Line) and 0–1600 Hz Bandwidth Response (Dashed Line)**



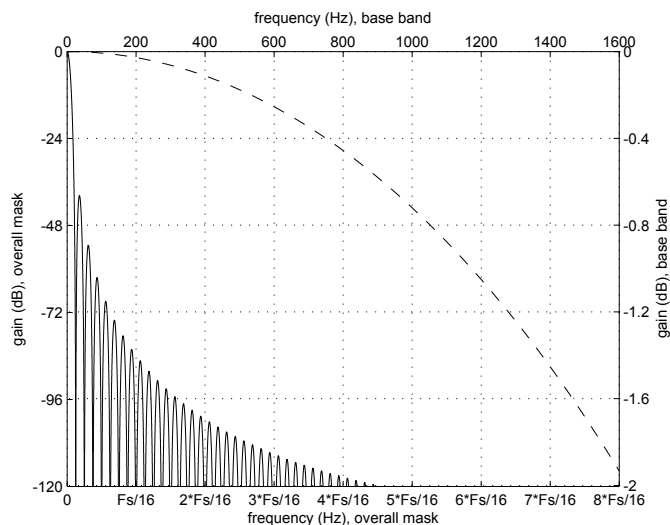
The zeros of this filter are located at multiples of  $\text{CLK\_SDADC\_FS}/32$ .

**Figure 40-5. Spectral Mask of an OSR = 64, CLK\_SDADC\_FS = 1 MHz, 3rd Order Sinc Filter Overall Response (Continuous Line) and 0–1600Hz Bandwidth Response (Dashed Line)**



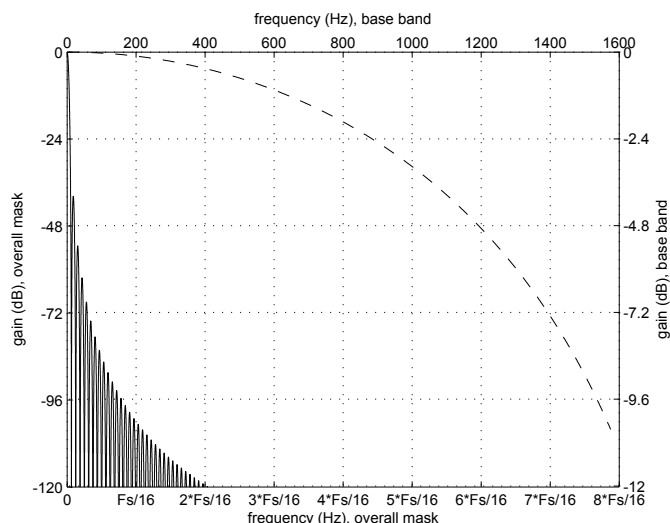
The zeros of this filter are located at multiples of  $\text{CLK\_SDADC\_FS}/64$ .

**Figure 40-6. Spectral Mask of an OSR = 128, CLK\_SDADC\_FS = 1 MHz, 3rd Order Sinc Filter Overall Response (Continuous Line) and 0–1600 Hz Bandwidth Response (Dashed Line)**



The zeros of this filter are located at multiples of CLK\_SDADC\_FS/128.

**Figure 40-7. Spectral Mask of an OSR = 256, CLK\_SDADC\_FS = 1 MHz, 3rd Order Sinc Filter Overall Response (Continuous Line) and 0–1600 Hz Bandwidth Response (Dashed Line)**



The zeros of this filter are located at multiples of CLK\_SDADC\_FS/256.

#### 40.6.3.3. Conversion Time

The time needed to convert a value depends on the selected OSR, PRESCALER and on the frequency of the SDADC.

For example, a sigma-delta converter running at CLK\_GEN\_SDADC = 1MHz with program the OSR of 64 and PRESCALER of 0. The output sampling rate equation is  $\text{CLK\_GEN\_SDADC}/(\text{OSR} * \text{PRESCALER} * 4)$  which means to converts data every  $(64*2*4)/1e6 = 512\mu\text{s}$ . The output data rate is then 1.953ksp/s.

Note: The CLK\_SDADC\_PRESCAL clock range is CLK\_GEN\_SDADC/2, if PRESCAL is 0

The OSR and PRESCALER are described in CTRLB on page 984 register.

#### 40.6.3.4. Gain and Offset Compensation

A specific offset, gain and shift can be applied to each source of the SDADC by performing the following operation:

$$\text{Data} = (\text{Data}_0 + \text{OFFSET}) \times \frac{\text{GAIN}}{2^{\text{SHIFT}}}$$

Where:

Data0 is an unsigned integer defined on 16 bits. It is the output of the decimation filter.

OFFSET is a signed integer defined on 24 bits (OFFSETCORR register).

GAIN is an unsigned integer defined on 14 bits (GAINCORR register).

SHIFT is an unsigned integer defined on 4 bits (SHIFTCORR register).

The result of the operation is then saturated to be within [0:216-1] and the 16 LSBs of this saturation operation are sent to the controller as the result of the SDADC conversion.

#### 40.6.4. DMA Operation

The SDADC 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.

#### 40.6.5. Interrupts

The SDADC 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 SDADC is reset. See [INTFLAG](#) on page 989 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](#) on page 41

#### 40.6.6. Events

The SDADC can generate the following output events:

- Result Ready (RESRDY): Generated when the conversion is complete and the result is available. Refer to [EVCTRL](#) on page 985 for details.
- Window Monitor (WINMON): Generated when the window monitor condition match. Refer to [WINCTRL](#) on page 993 register for details.

Writing a one to an Event Output bit in the Event Control Register (EVCTRL.xxEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. Refer to the Event System chapter for details on configuring the event system.

The SDADC can take the following actions on an input event:

- Start conversion (START): Start a conversion. Refer to [SWTRIG](#) on page 999 for details.
- Conversion flush (FLUSH): Flush the conversion. Refer to [SWTRIG](#) on page 999 for details.

Writing a one to an Event Input bit into the Event Control register (EVCTRL.xxEI) enables the corresponding action on input event. Writing a zero to this bit disables the corresponding action on input event.

The SDADC uses only asynchronous events and asynchronous Event System channel path must be configured. By default, the SDADC will detect a rising edge on the incoming event. If the SDADC action must be performed on the falling edge of the incoming event, the event line must be inverted first, by writing to one the corresponding Event Invert Enable bit in Event Control register (EVCTRL.xINV).

Note that If FLUSH and START events are available at the same time, the FLUSH event has higher priority.

#### Related Links

[EVSYS – Event System](#) on page 474

### 40.6.7. Sleep Mode Operation

The ONDEMAND and RUNSTDBY bits in the Control A register (CTRLA) control the behavior of the SDADC during standby sleep mode, in cases where the SDADC is enabled (CTRLA.ENABLE = 1). Note that when CTRLA.ONDEMAND is one, the analog

block is powered-off when the conversion is complete. When a start request is detected, the system returns from sleep and starts a new conversion after the start-up time delay.

**Table 40-1. SDADC Sleep Behavior**

CTRLA.RUNSTDBY	CTRLA.ONDEMAND	CTRLA.ENABLE	Description
x	x	0	Disabled
0	0	1	Run in all sleep modes except STANDBY.
0	1	1	Run in all sleep modes on request, except STANDBY.
1	0	1	Run in all sleep modes.
1	1	1	Run in all sleep modes on request.

### 40.6.8. Synchronization

Due to the asynchronicity between CLK\_SDADC\_APB and CLK\_GEN\_SDADC some registers must be synchronized when accessed. A register can require:

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

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

If an operation that require synchronization is executed while its busy bit is on, the operation is discarded and a bus error is generated.

The following bits need synchronization when written:

- Software Reset bit in Control A register (CTRLA.SWRST)
- Enable bit in Control A register (CTRLA.ENABLE)

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

The following registers need synchronization when written:

- Input Control register (INPUTCTRL)
- Reference Control register (REFCTRL)
- Control C register (CTRLC)
- Window Monitor Lower Threshold register (WINLT)
- Window Monitor Upper Threshold register (WINUT)
- Offset correction register (OFFSETCORR)
- Gain correction register (GAINCORR)
- Shift correction register (SHIFTCORR)
- Software Trigger register (SWTRIG)
- Analog Control Register (ANACTRL)

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

#### 40.6.9. Test Mode

The SDADC can generate the following output when TEST.TEST\_EN set to one:

- SDADC analog module data output
- SDADC analog module output enable
- SDADC Result[23:8] data output
- SDADC Result output enable

## 40.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY					ENABLE	SWRST
0x01	REFCTRL	7:0	ONREFBUF		REFRANGE[1:0]				REFSEL[1:0]	
0x02	CTRLB	7:0	PRESCALER[7:0]							
0x03		15:8	SKPCNT[3:0]					OSR[2:0]		
0x04	EVCTRL	7:0			WINMONEO	RESRDYEO	STARTINV	FLUSHINV	STARTEI	FLUSHEI
0x05	INTENCLR	7:0						WINMON	OVERRUN	RESRDY
0x06	INTENSET	7:0						WINMON	OVERRUN	RESRDY
0x07	INTFLAG	7:0						WINMON	OVERRUN	RESRDY
0x08	SEQSTATUS	7:0	SEQBUSY					SEQSTATE[3:0]		
0x09	INPUTCTRL	7:0						MUXSEL[3:0]		
0x0A	CTRLC	7:0								FREERUN
0x0B	WINCTRL	7:0						WINMODE[2:0]		
0x0C	WINLT	7:0	WINLT[7:0]							
0x0D		15:8	WINLT[15:8]							
0x0E		23:16	WINLT[23:16]							
0x0F		31:24								
0x10	WINUT	7:0	WINUT[7:0]							
0x11		15:8	WINUT[15:8]							
0x12		23:16	WINUT[23:16]							
0x13		31:24								
0x14	OFFSETCORR	7:0	OFFSETCORR[7:0]							
0x15		15:8	OFFSETCORR[15:8]							
0x16		23:16	OFFSETCORR[23:16]							
0x17		31:24								
0x18	GAINCORR	7:0								
0x19		15:8	GAINCORR[1:0]							
0x1A	SHIFTCORR	7:0	SHIFTCORR[7:0]							
0x1B	Reserved									
0x1C	SWTRIG	7:0							START	FLUSH
0x1D	Reserved									
...										
0x1F										
0x20	SYNCBUSY	7:0	OFFSETCORR	WINUT	WINLT	WINCTRL	MUXCTRL	CTRLC	ENABLE	SWRST
0x21		15:8					ANACTRL	SWTRIG	SHIFTCORR	GAINCORR
0x22		23:16								
0x23		31:24								
0x24	RESULT	7:0	RESULT[7:0]							
0x25		15:8	RESULT[15:8]							
0x26		23:16	RESULT[23:16]							
0x27		31:24								
0x28	SEQCTRL	7:0						SEQEN2	SEQEN1	SEQEN0

Offset	Name	Bit Pos.								
0x29	Reserved									
...										
0x2B										
0x2C	ANACTRL	7:0	BUFTTEST	ONCHOP		CTLSADADC[4:0]				
0x2D	Reserved									
0x2E	DBGCTRL	7:0								DBGRUN

## 40.8. 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

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

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.



### 40.8.1. Control A

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized (ENABLE, SWRST)

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

#### Bit 7 – ONDEMAND: On Demand Control

The On Demand operation modes allows the SDADC to be enabled or disabled, depending on other peripheral request.

In On Demand operation mode, i.e., if the ONDEMAND bit has been previously written to one, the SDADC will only be running when requested by a peripheral. If there is no peripheral requesting the SDADC will be in a disable state.

If On Demand is disable the SDADC will always be running when enabled.

In standby sleep mode, the On Demand operation is still active if the CTRLA.RUNSTDBY bit is one. If CTRLA.RUNSTDBY is zero, the SDADC is disabled.

This bit is not synchronized.

Value	Description
0	The SDADC is always on , if enabled.
1	The SDADC is enabled, when a peripheral is requesting the SDADC conversion. The SDADC is disabled if no peripheral is requesting it.

#### Bit 6 – RUNSTDBY: Run in Standby

This bit controls how the SDADC behaves during standby sleep mode:

This bit is not synchronized.

Value	Description
0	The SDADC is halted during standby sleep mode.
1	The SDADC is not stopped in standby sleep mode. If CTRLA.ONDEMAND is one, the SDADC will be running when a peripheral is requesting it. If CTRLA.ONDEMAND is zero, the SDADC will always be running in standby sleep mode.

#### 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 SDADC is disabled.
1	The SDADC 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 SDADC, except **SYNCBUSY** , to their initial state, and the SDADC 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 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.

## 40.8.2. Reference Control

**Name:** REFCTRL  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

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

### Bit 7 – ONREFBUF: Reference Buffer On

Turning on the buffer increases the impedance seen on the external reference, so that the current load reduces from  $5\mu A \pm 0.10\mu A$ . This needs to be matched with whatever type of reference circuit is used.

Value	Description
0	Reference Buffer Off
1	Reference Buffer On

### Bits 5:4 – REFRANGE[1:0]: Reference Range

REFRANGE[1:0]	Reference Voltage
0x0	$V_{ref} < 1.4V$
0x1	$1.4V < V_{ref} < 2.4V$
0x2	$2.4 < V_{ref} < 3.6V$
0x3	$V_{ref} > 3.6V$

### Bits 1:0 – REFSEL[1:0]: Reference Selection

These bits select the reference for the ADC.

Value	Name	Description
0x0	Internal bandgap	Internal 1.024V, 2.048V, 4.096V
0x1	AREFB pin	External 1-5.5V
0x2	DAC output	Internal 1-5.5V
0x3	VDDANA	Supply 2.7-5.5V

### 40.8.3. Control B

**Name:** CTRLB  
**Offset:** 0x02  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
	SKPCNT[3:0]					OSR[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
	PRESCALER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

#### Bits 15:12 – SKPCNT[3:0]: Skip Count

How many skip samples before retrieve the first valid sample.

The first valid sample starts from the third sample onward.

#### Bits 10:8 – OSR[2:0]: Over Sampling Ratio

OSR is the Over Sampling Ratio which can be modified to change the output data rate.

The OSR must never be changed while the SDADC is running. One must first place the SDADC in reset state, modify the OSR and then run the SDADC again.

Value	Name	Description
0x0	OSR64	Over Sampling Ratio is 64
0x1	OSR128	Over Sampling Ratio is 128
0x2	OSR256	Over Sampling Ratio is 256
0x3	OSR512	Over Sampling Ratio is 512
0x4	OSR1024	Over Sampling Ratio is 1024
0x4 - 0xF		Reserved

#### Bits 7:0 – PRESCALER[7:0]: Prescaler Configuration

The ADC uses the SDADC Clock to perform conversions.

The CLK\_SDADC\_PRESCAL clock range is between CLK\_GEN\_SDADC/2, if PRESCAL is 0, and CLK\_GEN\_SDADC/512, if PRESCAL is set to 255 (0xFF). PRESCAL must be programmed in order to provide an CLK\_SDADC\_PRESCAL clock frequency according to the parameters given in the product Electrical Characteristics section.

#### 40.8.4. Event Control

**Name:** EVCTRL  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
			WINMONEO	RESRDYEO	STARTINV	FLUSHINV	STARTEI	FLUSHEI
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	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 3 – STARTINV: Start Conversion Event Invert Enable

Value	Description
0	start event input source is not inverted.
1	start event input source is inverted.

##### Bit 2 – FLUSHINV: Flush Event Invert Enable

Value	Description
0	flush event input source is not inverted.
1	flush event input source is inverted.

##### Bit 1 – STARTEI: Start Conversion Event Input Enable

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.

##### Bit 0 – FLUSHEI: Flush Event Input Enable

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.

#### 40.8.5. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x05

**Reset:** 0x00

**Property:** PAC Write-Protection

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

##### 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, which disables 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, which disables 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, which disables the corresponding interrupt request.

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.

#### 40.8.6. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x06

**Reset:** 0x00

**Property:** PAC Write-Protection

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

##### 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, which enables 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, which enables 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, which enables the Result Ready interrupt.

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



## 40.8.7. Interrupt Flag Status and Clear

**Name:** INTFLAG

**Offset:** 0x07

**Reset:** 0x00

**Property:** –

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

### 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 CLK\_GEN\_SDADC 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.

## 40.8.8. Sequence Status

**Name:** SEQSTATUS

**Offset:** 0x08

**Reset:** 0x00

**Property:** -

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

### Bit 7 – SEQBUSY: Sequence busy

This bit is set when the sequence start.

This bit is clear when the last conversion in a sequence is done.

### Bits 3:0 – SEQSTATE[3:0]: Sequence State

This bit field is the pointer of sequence. This value identifies the last conversion done in the sequence.

## 40.8.9. Input Control

**Name:** INPUTCTRL

**Offset:** 0x09

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized

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

### Bits 3:0 – MUXSEL[3:0]: ADC Analog Input Selection

These bits define the Mux selection for the SDADC input.

Value	Name	Description
0x00	AIN0	Select ADC AINN0 and AINP0 pins
0x01	AIN1	Select ADC AINN1 and AINP1 pins
0x02	AIN2	Select ADC AINN2 and AINP2 pins
0x03 - 0x0F		Reserved

#### 40.8.10. Control C

**Name:** CTRLC

**Offset:** 0x0A

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized

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

##### Bit 0 – FREERUN: Free Running Mode

Value	Description
0	The SDADC run in single conversion mode.
1	The SDADC is in free running mode and a new conversion will be initiated when a previous conversion completes.

#### 40.8.11. Window Monitor Control

**Name:** WINCTRL  
**Offset:** 0x0B  
**Reset:** 0x00  
**Property:** PAC Write-Protection, 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.

Value	Name	Description
0x0	DISABLE	No window mode (default)
0x1	ABOVE	RESULT > WINLT
0x2	BELOW	RESULT < WINUT
0x3	INSIDE	WINLT < RESULT < WINUT
0x4	OUTSIDE	WINUT < RESULT or RESULT < WINLT
0x5 - 0x7		Reserved

## 40.8.12. Window Monitor Lower Threshold

**Name:** WINLT  
**Offset:** 0x0C  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	WINLT[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
	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 23:0 – WINLT[23:0]: Window Lower Threshold

If the window monitor is enabled, these bits define the lower threshold value.

### 40.8.13. Window Monitor Upper Threshold

**Name:** WINUT  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	WINUT[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
	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 23:0 – WINUT[23:0]: Window Upper Threshold

If the window monitor is enabled, these bits define the upper threshold value.

#### 40.8.14. Offset Correction

**Name:** OFFSETCORR  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	OFFSETCORR[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
	OFFSETCORR[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
	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 23:0 – OFFSETCORR[23:0]: Offset Correction

The OFFSETCORR is a signed integer value.

A specific offset, gain and shift can be applied to SDADC by performing the following operation:

$$\text{AD14Code Calibrated} = (\text{ADC14Code} + \text{OFFSET}) * \text{GAIN} / 2^{\text{SHIFT}}$$



#### 40.8.15. Gain Correction

**Name:** GAINCORR  
**Offset:** 0x18  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	GAINCORR[1:0]							
Access	R	R						
Reset	0	0						
Bit	7	6	5	4	3	2	1	0
Access								
Reset								

##### **Bits 15:14 – GAINCORR[1:0]: Gain Correction**

A specific offset, gain and shift can be applied to SDADC by performing the following operation:

$$\text{AD14Code Calibrated} = (\text{ADC14Code} + \text{OFFSET}) * \text{GAIN}/2^{\text{SHIFT}}$$

#### 40.8.16. Shift Correction

**Name:** SHIFTCORR  
**Offset:** 0x1A  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	SHIFTCORR[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 – SHIFTCORR[23:0]: Shift Correction

A specific offset, gain and shift can be applied to SDADC by performing the following operation:

$$\text{AD14Code Calibrated} = (\text{ADC14Code} + \text{OFFSET}) * \text{GAIN}/2^{\text{SHIFT}}$$

## 40.8.17. Software Trigger

**Name:** SWTRIG

**Offset:** 0x1C

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
							START	FLUSH
Access							W	W
Reset							0	0

### Bit 1 – START: SDADC Start Conversion

Writing a one to this bit will start a conversion or sequence. The bit is cleared by hardware when the conversion has started. Setting this bit when it is already set has no effect.

Writing this bit to zero will have no effect.

### Bit 0 – FLUSH: SDADC Conversion Flush

Writing a one to this bit will flush the SDADC pipeline. A flush will restart the SDADC conversion and all conversions in progress will be aborted and lost. This bit is cleared until the SDADC has been flushed.

After the flush, the ADC will resume where it left off; i.e., if a conversion was pending, the ADC will start a new conversion.

Writing this bit to zero will have no effect.

## 40.8.18. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x20  
**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					R	R	R	R
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

### Bit 11 – ANACTRL: Analog Control Synchronization Busy

This bit is cleared when the synchronization of ANACTRL register between the clock domains is complete.

This bit is set when the synchronization of ANACTRL register between clock domains is started.

### Bit 10 – SWTRIG: Software Trigger Synchronization Busy

This bit is cleared when the synchronization of SWTRIG register between the clock domains is complete.

This bit is set when the synchronization of SWTRIG register between clock domains is started.

### Bit 9 – SHIFTCORR: Shift Correction Synchronization Busy

This bit is cleared when the synchronization of SHIFTCORR register between the clock domains is complete.

This bit is set when the synchronization of SHIFTCORR register between clock domains is started.

### Bit 8 – GAINCORR: Gain Correction Synchronization Busy

This bit is cleared when the synchronization of GAINCORR register between the clock domains is complete.

This bit is set when the synchronization of GAINCORR register between clock domains is started.

**Bit 7 – OFFSETCORR: Offset Correction Synchronization Busy**

This bit is cleared when the synchronization of OFFSETCORR register between the clock domains is complete.

This bit is set when the synchronization of OFFSETCORR register between clock domains is started.

**Bit 6 – WINUT: Window Monitor Lower Threshold Synchronization Busy**

This bit is cleared when the synchronization of WINUT register between the clock domains is complete.

This bit is set when the synchronization of WINUT register between clock domains is started.

**Bit 5 – WINLT: Window Monitor Upper Threshold Synchronization Busy**

This bit is cleared when the synchronization of WINLT register between the clock domains is complete.

This bit is set when the synchronization of WINLT register between clock domains is started.

**Bit 4 – WINCTRL: Window Monitor Control Synchronization Busy**

This bit is cleared when the synchronization of WINCTRL register between the clock domains is complete.

This bit is set when the synchronization of WINCTRL register between clock domains is started.

**Bit 3 – MUXCTRL: Mux Control Synchronization Busy**

This bit is cleared when the synchronization of MUXCTRL register between the clock domains is complete.

This bit is set when the synchronization of MUXCTRL register between clock domains is started.

**Bit 2 – CTRLC: Control C Synchronization Busy**

This bit is cleared when the synchronization of CTRLC register between the clock domains is complete.

This bit is set when the synchronization of CTRLC register between clock domains is started.

**Bit 1 – ENABLE: ENABLE Synchronization Busy**

This bit is cleared when the synchronization of ENABLE register between the clock domains is complete.

This bit is set when the synchronization of ENABLE register between clock domains is started.

**Bit 0 – SWRST: SWRST Synchronization Busy**

This bit is cleared when the synchronization of SWRST register between the clock domains is complete.

This bit is set when the synchronization of SWRST register between clock domains is started

## 40.8.19. Result

**Name:** RESULT  
**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
	RESULT[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
	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 23:0 – RESULT[23:0]: Result Conversion Value

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

The RESULT is a signed integer value.

## 40.8.20. Sequence Control

**Name:** SEQCTRL  
**Offset:** 0x28  
**Reset:** 0x00  
**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
						SEQEN2	SEQEN1	SEQEN0
Access						R/W	R/W	R/W
Reset						0	0	0

### Bits 2:0 – SEQENn: Enable Positive Input in the Sequence

For details on available mux selections, refer to [INPUTCTRL](#) on page 991.

The sequence start from the lowest input, and go to the next enabled input automatically when the conversion is done. If no bits are set the sequence is disabled.

Value	Description
0	Disable the positive input mux n selection from the sequence.
1	Enable the positive input mux n selection to the sequence.

## 40.8.21. Analog Control

**Name:** ANACTRL

**Offset:** 0x2C

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized.

Bit	7	6	5	4	3	2	1	0
	BUFTEST	ONCHOP		CTLSDADC[4:0]				
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 – BUFTEST: Buffer Test**

**Bit 6 – ONCHOP: ONCHOP**

Value	Description
0	No Chopper at SDADC input
1	Chopper at SDADC input

**Bits 4:0 – CTLSDADC[4:0]: CTLSDADC**

SDADC Bias Current Control and used for Debug/Characterization



## 40.8.22. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x2E  
**Reset:** 0x00  
**Property:** PAC Write-Protectedion

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.

This bit should be written only while a conversion is not ongoing.

Value	Description
0	The SDADC is halted when the CPU is halted by an external debugger.
1	The SDADC continues normal operation when the CPU is halted by an external debugger.

## 41. AC – Analog Comparators

### 41.1. Overview

The Analog Comparator (AC) supports multiple 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 and propagation delay are two important properties of the comparators' dynamic behavior. Both parameters may 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 always grouped in pairs on each port. The AC peripheral implements one or two pairs of comparators. These are called Comparator 0 (COMP0) and Comparator 1 (COMP1) for the first pair and Comparator 2 (COMP2) and Comparator 3 (COMP3) for the second pair. They have identical behaviors, but separate control registers. Each pair can be set in window mode to compare a signal to a voltage range instead of a single voltage level.

### 41.2. Features

- Four individual comparators
- Selectable propagation delay versus current consumption
- Selectable hysteresis
  - 4-levels 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 (if available)
- 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
- Low-power option
  - Single-shot support

### 41.3. Block Diagram

Figure 41-1. Analog Comparator Block Diagram (First Pair)

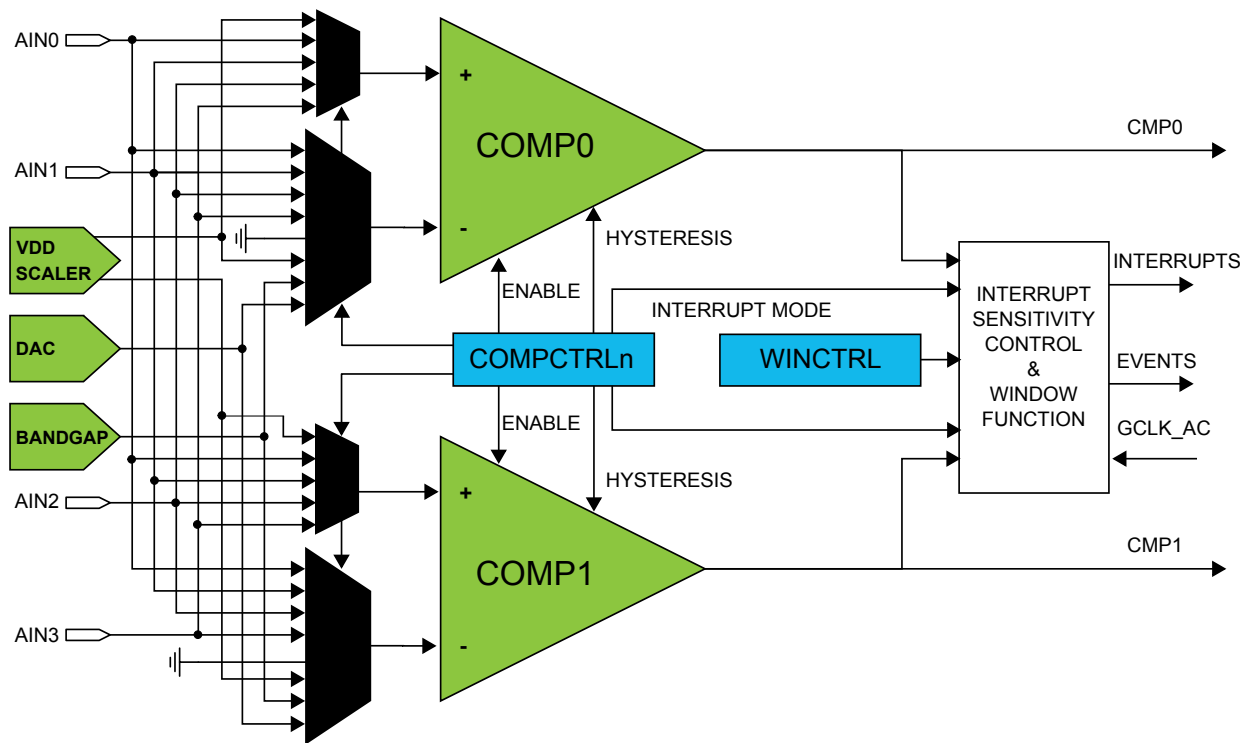
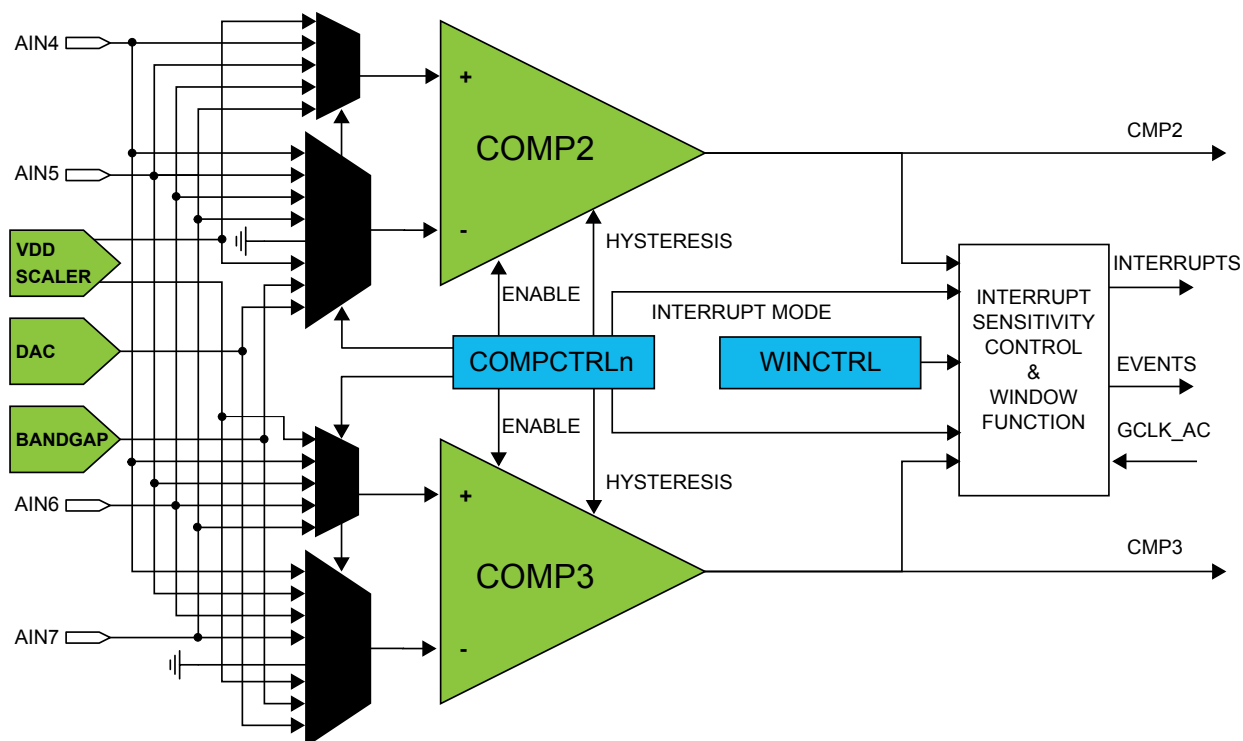


Figure 41-2. Analog Comparator Block Diagram (Second Pair)



## 41.4. Signal Description

Signal	Description	Type
AIN[7..0]	Analog input	Comparator inputs
CMP[3..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](#) on page 27

## 41.5. Product Dependencies

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

### 41.5.1. 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: IO Pin Controller](#) on page 444

### 41.5.2. 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](#) on page 162

### 41.5.3. Clocks

The AC bus clock (CLK\_AC\_APB) can be enabled and disabled in the Power Manager, and the default state of CLK\_AC\_APB can be found in the Peripheral Clock Masking section in the Power Manager description.

A generic clock (GCLK\_AC) is required to clock the AC. This clock must be configured and enabled in the generic clock controller before using the AC. Refer to the Generic Clock Controller chapter for details.

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](#) on page 162

### 41.5.4. DMA

Not applicable.

### 41.5.5. 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](#) on page 41

### 41.5.6. 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](#) on page 474

### 41.5.7. Debug Operation

When the CPU is halted in debug mode, the AC will halt normal operation after any on-going comparison is completed. The AC can be forced to continue normal operation during debugging. Refer to [DBGCTRL](#) for details. If the AC 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.

### 41.5.8. Register Access Protection

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

- 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](#) on page 46

### 41.5.9. 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.

## 41.6. Functional Description

### 41.6.1. 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).

### 41.6.2. Basic Operation

#### 41.6.2.1. Initialization

Some registers are enable-protected, meaning they can only be written when the module is disabled.

The following register is enable-protected:

- Event Control register (EVCTRL)

Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

#### 41.6.2.2. 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.

#### 41.6.2.3. 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 desired hysteresis with COMPCTRLx.HYSTEN and COMPCTRLx.HYST. 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.

#### 41.6.2.4. 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](#) on page 1105

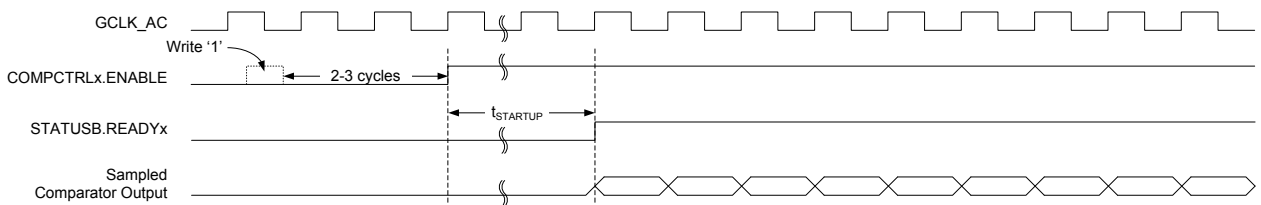
#### Continuous Measurement

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 GCLK\_AC frequency. An example of continuous measurement is shown in the [Figure 41-3 Continuous Measurement Example](#) on page 1011.

**Figure 41-3. 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 GCLK\_AC to register the appropriate peripheral events and interrupts. The GCLK\_AC clock is then disabled again automatically, unless configured to wake up the system from sleep.

#### Related Links

[Electrical Characteristics](#) on page 1105

#### Single-Shot

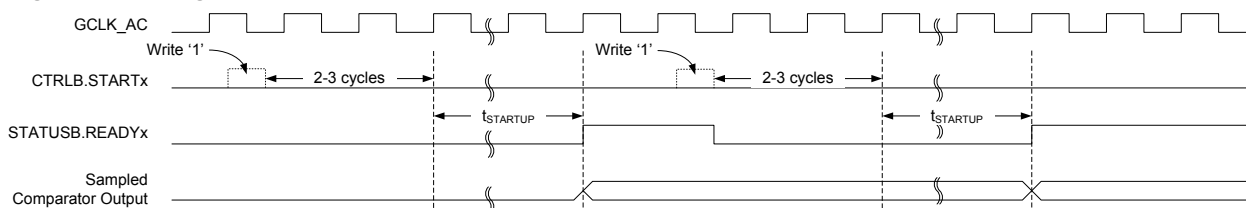
Single-shot operation is selected by writing COMPCTRLx.SINGLE to '1'. During single-shot operation, the comparator is normally idle. The user starts a single comparison by writing '1' to the respective Start Comparison bit in the write-only Control B register (CTRLB.STARTx). The comparator is enabled, 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.

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 [Figure 41-4 Single-Shot Example](#) on page 1012.

**Figure 41-4. 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 GCLK\_AC. 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 GCLK\_AC are then disabled again automatically, unless configured to wake up the system from sleep.

#### Related Links

[Electrical Characteristics](#) on page 1105

### 41.6.3. Selecting Comparator Inputs

Each comparator has one positive and one negative input. The positive input is fed from an external input pin (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 usage 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.

#### 41.6.4. Window Operation

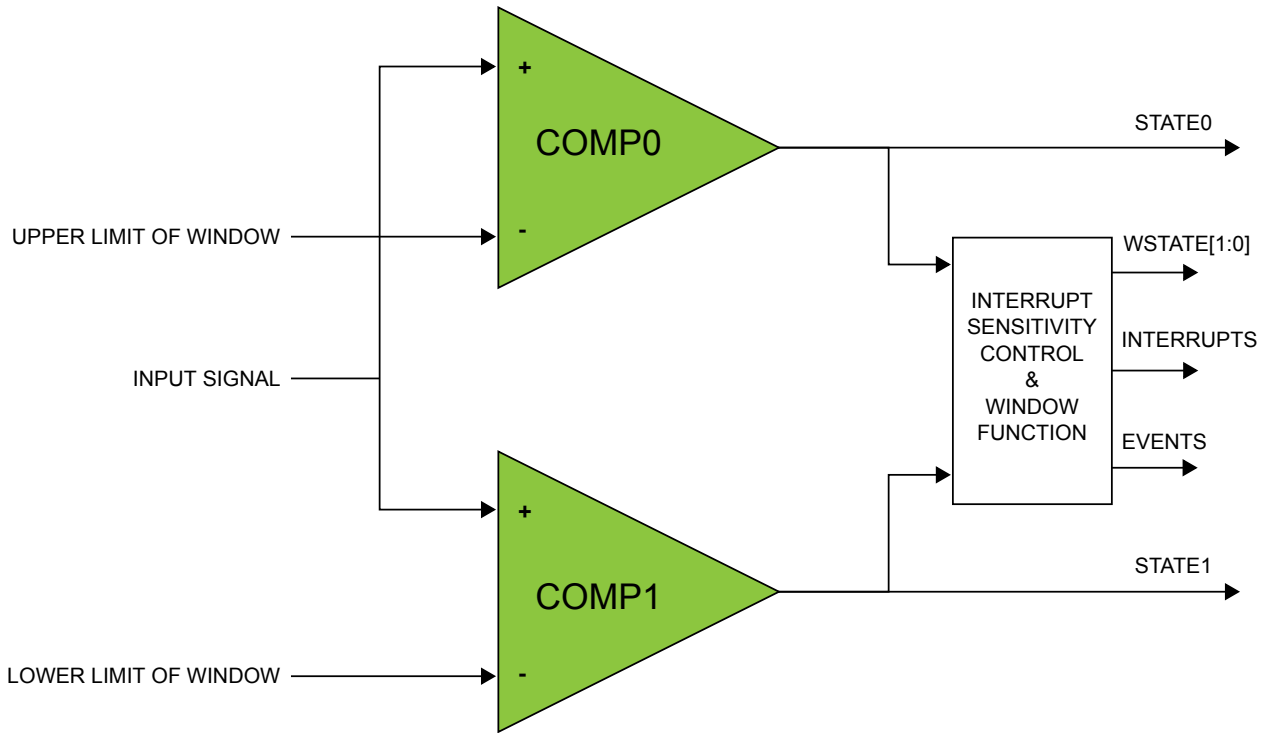
Each comparator pair can be configured to work together in window mode. In this mode, a voltage range is defined, and the comparators give information about whether an input signal is within this range or not. Window mode is enabled by the Window Enable x bit in the Window Control register (WINCTRL.WENx). 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 41-5 Comparators in Window Mode](#) on page 1014, 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.

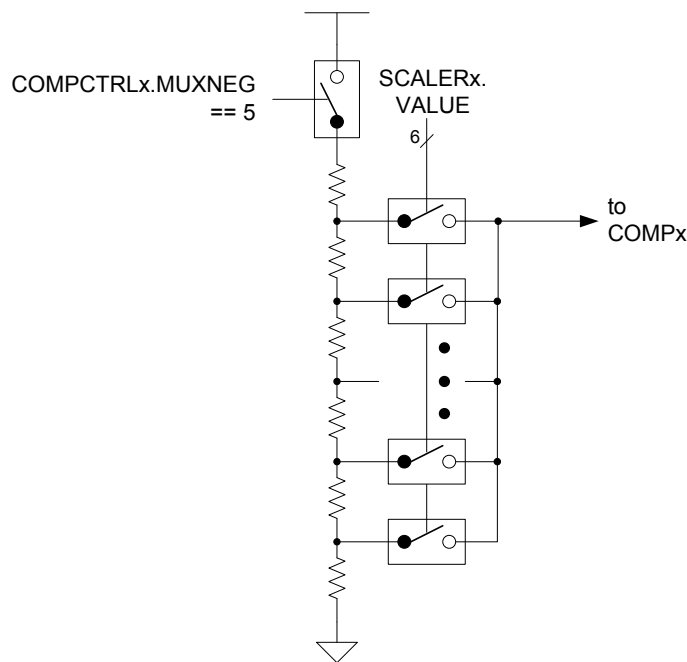
**Figure 41-5. Comparators in Window Mode**



#### 41.6.5. VDD Scaler

The VDD 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 41-6. VDD Scaler**



### 41.6.6. 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). Furthermore, when enabled, the level of hysteresis is programmable through the Hysteresis Level bits also in the Comparator x Control register (COMPCTRLx.HYST). Hysteresis is available only in continuous mode (COMPCTRLx.SINGLE=0).

### 41.6.7. Propagation Delay vs. Power Consumption

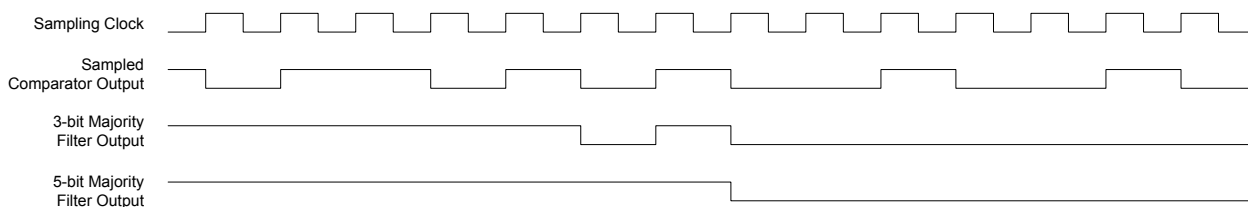
It is possible to trade off comparison speed for power efficiency to get the shortest possible propagation delay or the lowest power consumption. The speed setting is configured for each comparator individually by the Speed bit group in the Comparator x Control register (COMPCTRLx.SPEED). The Speed bits select the amount of bias current provided to the comparator, and as such will also affect the start-up time.

### 41.6.8. Filtering

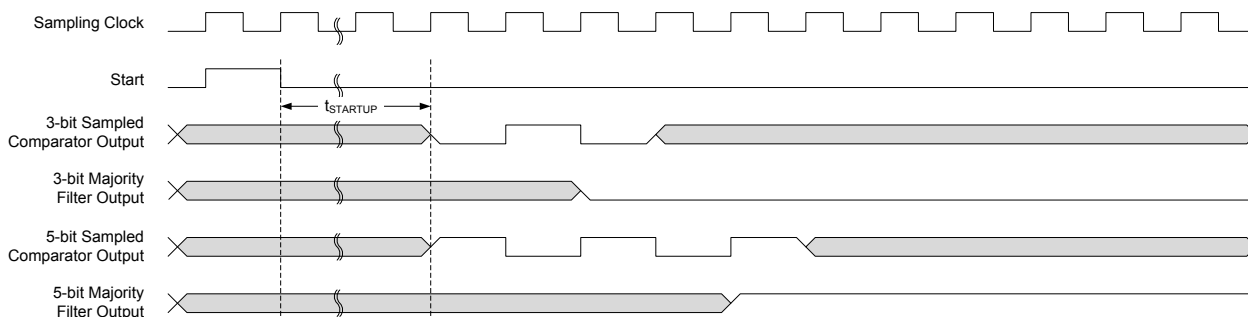
The output of the comparators can be filtered digitally to reduce noise. The filtering is determined by the Filter Length bits in the Comparator Control x register (COMPCTRLx.FLEN), and is independent for each comparator. Filtering is selectable from none, 3-bit majority (N=3) or 5-bit majority (N=5) 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 41-7 Continuous Mode Filtering](#) on page 1015. For single-shot mode, the comparison completes after the Nth filter sample, as shown in [Figure 41-8 Single-Shot Filtering](#) on page 1015.

**Figure 41-7. Continuous Mode Filtering**



**Figure 41-8. 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.

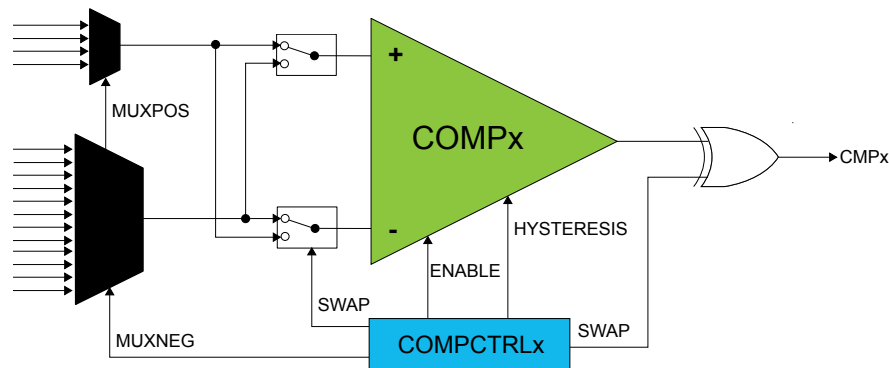
### 41.6.9. Comparator Output

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.

### 41.6.10. 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 41-9 Input Swapping for Offset Compensation](#) on page 1016. 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 41-9. Input Swapping for Offset Compensation**



### 41.6.11. DMA Operation

Not applicable.

### 41.6.12. Interrupts

The AC has the following interrupt sources:

- Comparator (COMP0, COMP1, COMP2, COMP3): Indicates a change in comparator status.
- Window (WIN0, WIN1): 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.WINTSELx[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](#) 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](#) on page 41

### 41.6.13. Events

The AC can generate the following output events:

- Comparator (COMP0, COMP1, COMP2, COMP3): Generated as a copy of the comparator status
- Window (WIN0, WIN1): Generated as a copy of the window inside/outside status

Writing a one to an Event Output bit in the Event Control Register (EVCTRL.xxEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. Refer to the Event System chapter for details on configuring the event system.

The AC can take the following action on an input event:

- Start comparison (START0, START1, START2, START3): Start a comparison.

Writing a one to an Event Input bit into the Event Control register (EVCTRL.COMPxEI) 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.

### 41.6.14. 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.

For Window Mode operation, both comparators in a pair must have the same RUNSTDBY configuration.

When RUNSTDBY is one, any enabled AC interrupt source can wake up the CPU. 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, as listed in [Table 41-1 Sleep Mode Operation](#) on page 1017.

**Table 41-1. Sleep Mode Operation**

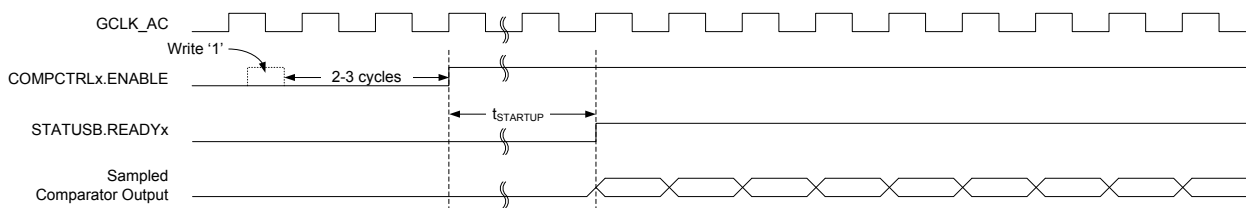
COMPCTRLx.MODE	RUNSTDBY=0	RUNSTDBY=1
0 (Continuous)	COMPx disabled	GCLK_AC stopped, COMPx enabled
1 (Single-shot)	COMPx disabled	GCLK_AC stopped, COMPx enabled only when triggered by an input event

#### 41.6.14.1. Continuous Measurement during Sleep

When a comparator is enabled in continuous measurement mode and GCLK\_AC 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 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 is disabled until the next edge detection. Filtering is not possible with this configuration.

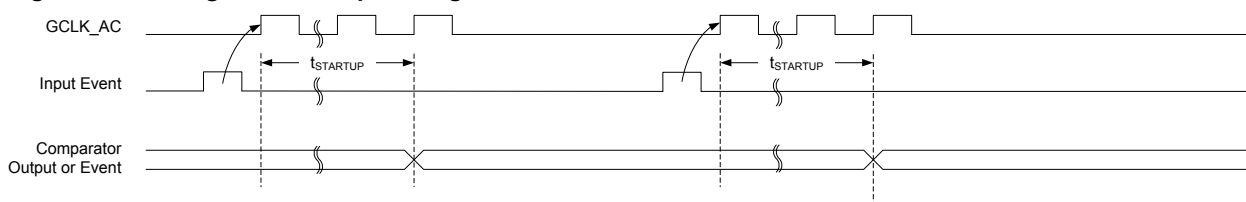
**Figure 41-10. Continuous Mode SleepWalking**



#### 41.6.14.2. 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. 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 shown in [Figure 41-11 Single-Shot SleepWalking](#) on page 1018. The comparator and GCLK\_AC are then disabled again automatically, unless configured to wake the system from sleep. Filtering is allowed with this configuration.

**Figure 41-11. Single-Shot SleepWalking**



#### 41.6.15. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

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](#) on page 110

## 41.7. Register Summary

Offset	Name	Bit Pos.									
0x00	CTRLA	7:0							ENABLE	SWRST	
0x01	CTRLB	7:0					START3	START2	START1	START0	
0x02	EVCTRL	7:0			WINEO1	WINEO0	COMPEO3	COMPEO2	COMPEO1	COMPEO0	
0x03		15:8	INVEI3	INVEI2	INVEI1	INVEI0	COMPEI3	COMPEI2	COMPEI1	COMPEI0	
0x04	INTENCLR	7:0			WIN1	WIN0	COMP3	COMP2	COMP1	COMP0	
0x05	INTENSET	7:0			WIN1	WIN0	COMP3	COMP2	COMP1	COMP0	
0x06	INTFLAG	7:0			WIN1	WIN0	COMP3	COMP2	COMP1	COMP0	
0x07	STATUSA	7:0	WSTATE1[1:0]		WSTATE0[1:0]		STATE3	STATE2	STATE1	STATE0	
0x08	STATUSB	7:0					READY3	READY2	READY1	READY0	
0x09	DBGCTRL	7:0								DBGRUN	
0x0A	WINCTRL	7:0		WINTSEL1[1:0]		WEN1		WINTSEL0[1:0]		WEN0	
0x0B	Reserved										
0x0C	SCALER0	7:0					VALUE[5:0]				
0x0D	SCALER1	7:0					VALUE[5:0]				
0x0E	SCALER2	7:0					VALUE[5:0]				
0x0F	SCALER3	7:0					VALUE[5:0]				
0x10	COMPCTRL0	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE		
0x11		15:8	SWAP		MUXPOS[2:0]			MUXNEG[2:0]			
0x12		23:16			HYST[1:0]		HYSTEN		SPEED[1:0]		
0x13		31:24			OUT[1:0]			FLEN[2:0]			
0x14	COMPCTRL1	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE		
0x15		15:8	SWAP		MUXPOS[2:0]			MUXNEG[2:0]			
0x16		23:16			HYST[1:0]		HYSTEN		SPEED[1:0]		
0x17		31:24			OUT[1:0]			FLEN[2:0]			
0x18	COMPCTRL2	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE		
0x19		15:8	SWAP		MUXPOS[2:0]			MUXNEG[2:0]			
0x1A		23:16			HYST[1:0]		HYSTEN		SPEED[1:0]		
0x1B		31:24			OUT[1:0]			FLEN[2:0]			
0x1C	COMPCTRL3	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE		
0x1D		15:8	SWAP		MUXPOS[2:0]			MUXNEG[2:0]			
0x1E		23:16			HYST[1:0]		HYSTEN		SPEED[1:0]		
0x1F		31:24			OUT[1:0]			FLEN[2:0]			
0x20	SYNDBUSY	7:0		COMPCTRL3	COMPCTRL2	COMPCTRL1	COMPCTRL0	WINCTRL	ENABLE	SWRST	
0x21		15:8									
0x22		23:16									
0x23		31:24									

## 41.8. 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.



### 41.8.1. Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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

#### Bit 1 – ENABLE: Enable

Due to synchronization, there is delay from updating the register until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately and the corresponding bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE 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 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.

## 41.8.2. Control B

**Name:** CTRLB  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** –

Bit	7	6	5	4	3	2	1	0
					START3	START2	START1	START0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

### Bits 3,2,1,0 – 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.

### 41.8.3. Event Control

**Name:** EVCTRL  
**Offset:** 0x02  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
	INVEI3	INVEI2	INVEI1	INVEI0	COMPEI3	COMPEI2	COMPEI1	COMPEI0
Access	R/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
			WINEO1	WINEO0	COMPEO3	COMPEO2	COMPEO1	COMPEO0
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

#### Bits 15,14,13,12 – INVEIx: Inverted Event Input Enable x

Value	Description
0	Incoming event is not inverted for comparator x.
1	Incoming event is inverted for comparator x.

#### Bits 11,10,9,8 – 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.

#### Bits 5,4 – WINEOx: Window x Event Output Enable

These bits indicate whether the window x function can generate a peripheral event or not.

Value	Description
0	Window x Event is disabled.
1	Window x Event is enabled.

#### Bits 3,2,1,0 – 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.

#### 41.8.4. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x04

**Reset:** 0x00

**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
			WIN1	WIN0	COMP3	COMP2	COMP1	COMP0
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

##### Bits 5,4 – WINx: Window x Interrupt Enable

Reading this bit returns the state of the Window x interrupt enable.

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

Writing a '1' to this bit disables the Window x interrupt.

Value	Description
0	The Window x interrupt is disabled.
1	The Window x interrupt is enabled.

##### Bits 3,2,1,0 – 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.

### 41.8.5. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x05

**Reset:** 0x00

**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
			WIN1	WIN0	COMP3	COMP2	COMP1	COMP0
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

#### Bits 5,4 – WINx: Window x Interrupt Enable

Reading this bit returns the state of the Window x interrupt enable.

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

Writing a '1' to this bit enables the Window x interrupt.

Value	Description
0	The Window x interrupt is disabled.
1	The Window x interrupt is enabled.

#### Bits 3,2,1,0 – 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.

## 41.8.6. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** –

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

### Bits 5,4 – WINx: Window x

This flag is set according to the Window x 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 x interrupt flag.

### Bits 3,2,1,0 – 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.

## 41.8.7. Status A

**Name:** STATUSA

**Offset:** 0x07

**Reset:** 0x00

**Property:** –

Bit	7	6	5	4	3	2	1	0
	WSTATE1[1:0]		WSTATE0[1:0]		STATE3	STATE2	STATE1	STATE0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

### Bits 7:6 – WSTATE1[1:0]: Window 1 Current State

These bits show the current state of the signal if the window 1 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 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 3,2,1,0 – STATEx: Comparator x Current State

This bit shows the current state of the output signal from COMPx. STATEx is valid only when STATUSB.READYx is one.

### 41.8.8. Status B

**Name:** STATUSB

**Offset:** 0x08

**Reset:** 0x00

**Property:** –

Bit	7	6	5	4	3	2	1	0
					READY3	READY2	READY1	READY0
Access					R	R	R	R
Reset					0	0	0	0

#### **Bits 3,2,1,0 – READYx: 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.

If comparator x is not implemented, READYx always reads as zero.



### 41.8.9. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x09  
**Reset:** 0x00  
**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
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 AC is halted when the CPU is halted by an external debugger. Any on-going comparison will complete.
1	The AC continues normal operation when the CPU is halted by an external debugger.

#### 41.8.10. Window Control

**Name:** WINCTRL  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
		WINTSEL1[1:0]		WEN1		WINTSEL0[1:0]		WEN0
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

##### Bits 6:5 – WINTSEL1[1:0]: Window 1 Interrupt Selection

These bits configure the interrupt mode for the comparator window 1 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 4 – WEN1: Window 1 Mode Enable

Value	Description
0	Window mode is disabled for comparators 2 and 3.
1	Window mode is enabled for comparators 2 and 3.

##### 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.

### 41.8.11. Scaler n

**Name:** SCALERn  
**Offset:** 0x0C+n\*0x1 [n=0..3]  
**Reset:** 0x00  
**Property:** Write-Protected

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:

$$V_{SCALE} = \frac{V_{DD} \cdot (VALUE + 1)}{64}$$

## 41.8.12. Comparator Control n

**Name:** COMPCTRLn  
**Offset:** 0x10+n\*0x4 [n=0..3]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
			OUT[1:0]			FLEN[2:0]		
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
			HYST[1:0]		HYSTEN		SPEED[1:0]	
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
	SWAP	MUXPOS[2:0]				MUXNEG[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
		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE	
Access		R/W		R/W	R/W	R/W	R/W	
Reset		0		0	0	0	0	

### Bits 29:28 – 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

### 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

#### Bits 21:20 – HYST[1:0]: Hysteresis Level

These bits indicate the hysteresis level of comparator n when hysteresis is enabled (COMPCTRLn.HYSTEN=1). Hysteresis is available only for continuous mode (COMPCTRLn.SINGLE=0). COMPCTRLn.HYST can be written only while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Name	Description
0x0	HYST50	50mV
0x1	HYST70	70mV
0x2	HYST90	90mV
0x3	HYST110	110mV

#### Bit 19 – HYSTEN: 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.

Value	Description
0	Hysteresis is disabled.
1	Hysteresis is enabled.

#### Bits 17:16 – 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	MEDLOW	Medium low speed
0x2	MEDHIGH	Medium high speed
0x3	HIGH	High speed

#### 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 14:12 – MUXPOS[2:0]: 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
0x4	VSCALE	VDD scaler
0x5–0x7		Reserved

#### 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
0x4	GND	Ground
0x5	VSCALE	VDD scaler
0x6	BANDGAP	Internal bandgap voltage
0x7	DAC	DAC output

#### Bit 6 – RUNSTDBY: Run in Standby

This bit controls the behavior of the comparator during standby sleep mode.

This bit is not synchronized

Value	Description
0	The comparator is disabled during sleep.
1	The comparator continues to operate during sleep.

#### Bits 4:3 – 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)

#### Bit 2 – 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 1 – ENABLE: Enable

Writing a zero to this bit disables comparator n.

Writing a one to this bit enables comparator n.

Due to synchronization, there is delay from updating the register until the comparator is enabled/disabled. The value written to COMPCTRLn.ENABLE will read back immediately after being written.

SYNCBUSY.COMPCTRLn is set. SYNCBUSY.COMPCTRLn is cleared when the peripheral is enabled/disabled.

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.

### 41.8.13. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x20  
**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
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

#### Bit 2 – WINCTRL: WINCTRL Synchronization Busy

This bit is cleared when the synchronization of the WINCTRL register between the clock domains is complete.

This bit is set when the synchronization of the WINCTRL register between clock domains is started.

#### Bit 1 – ENABLE: Enable Synchronization Busy

This bit is cleared when the synchronization of the CTRLA.ENABLE bit between the clock domains is complete.

This bit is set when the synchronization of the CTRLA.ENABLE bit between clock domains is started.

#### Bit 0 – SWRST: Software Reset Synchronization Busy

This bit is cleared when the synchronization of the CTRLA.SWRST bit between the clock domains is complete.

This bit is set when the synchronization of the CTRLA.SWRST bit between clock domains is started.

#### Bits 6,5,4,3 – COMPCTRLx: COMPCTRLx Synchronization Busy

This bit is cleared when the synchronization of the COMPCTRLx register between the clock domains is complete.

This bit is set when the synchronization of the COMPCTRLx register between clock domains is started.



## 42. DAC – Digital-to-Analog Converter

### 42.1. Overview

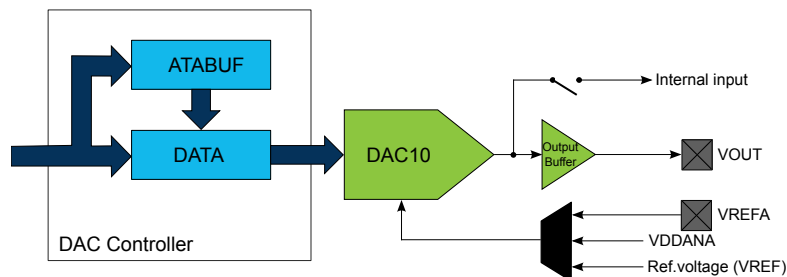
The Digital-to-Analog Converter (DAC) converts a digital value to a voltage. The DAC has one channel with 10-bit resolution, and it is capable of converting up to 350,000 samples per second (350kps).

### 42.2. Features

- DAC with 10-bit resolution
- Up to 350kps conversion rate
- Hardware support for 14-bit using dithering
- Multiple trigger sources
- High-drive capabilities
- Output can be used as input to the Analog Comparator (AC), ADC or SDADC
- DMA support

### 42.3. Block Diagram

Figure 42-1. DAC Block Diagram



### 42.4. Signal Description

Signal Name	Type	Description
VOUT	Analog output	DAC output
VREFA	Analog input	External reference

#### Related Links

[I/O Multiplexing and Considerations](#) on page 27

### 42.5. Product Dependencies

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

#### 42.5.1. 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](#) on page 444

#### 42.5.2. 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](#) on page 162

#### 42.5.3. Clocks

The DAC bus clock (CLK\_DAC\_APB) can be enabled and disabled by the Main Clock module, 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](#) on page 1041 for further details.

### Related Links

[GCLK - Generic Clock Controller](#) on page 114

#### 42.5.4. 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](#) on page 334

#### 42.5.5. 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](#) on page 41

#### 42.5.6. Events

The events are connected to the Event System.

### Related Links

[EVSYS – Event System](#) on page 474

#### 42.5.7. 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.

### 42.5.8. 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](#) on page 46

### 42.5.9. 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.

## 42.6. Functional Description

### 42.6.1. 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.

### 42.6.2. Basic Operation

#### 42.6.2.1. 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).

#### 42.6.2.2. 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.

#### 42.6.2.3. Enabling the Output Buffer

To enable the DAC output on the V<sub>OUT</sub> 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.

#### 42.6.2.4. 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:

$$V_{\text{OUT}} = \frac{\text{DATA}}{0x3FF} \cdot V_{\text{REF}}$$

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](#) on page 1041 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.

#### 42.6.3. 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.

#### 42.6.4. 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](#) on page 1041 for details.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear register (INTENCLR).

An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the DAC is reset. See [INTFLAG](#) on page 1051 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

#### 42.6.5. 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](#) on page 1040 for details.

Writing a '1' to an Event Output bit in the Event Control register (EVCTRL.EMPTYEO) 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](#) on page 1040 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.INVEIx.

#### Related Links

[EVSYS – Event System](#) on page 474

#### 42.6.6. 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.

#### 42.6.7. 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 corresponding status bit in the Synchronization Busy register (SYNCBUSY) will be set immediately, and cleared when synchronization is complete.

If an operation that requires synchronization is executed while 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.

No bits need synchronization when read.

## 42.6.8. Additional Features

### 42.6.8.1. 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.

### 42.6.8.2. 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.

### 42.6.8.3. 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.

### 42.6.8.4. Dithering mode

In dithering mode, DATA is a 14-bit signed value where DATA[13:4] is the 10-bit data converted by DAC and DATA[3:0] the dither bits, used to minimize the quantization error.

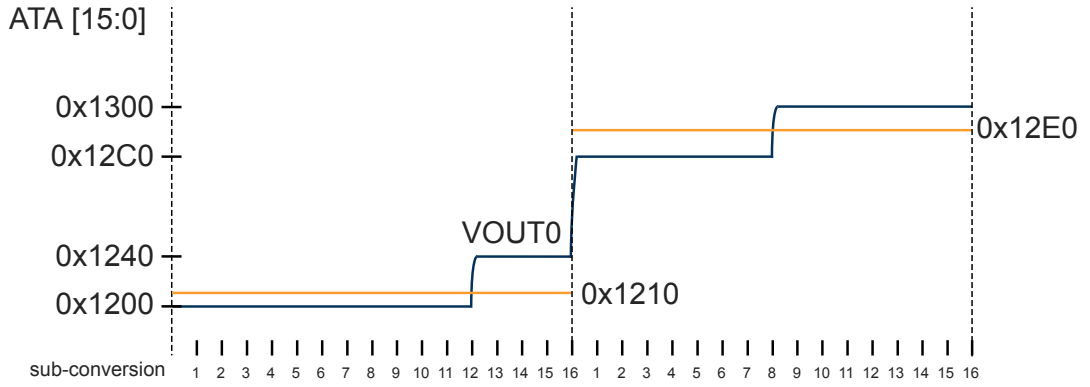
The principle is to make 16 sub-conversions of DATA[13:4] value or (DATA[13:4] + 1) value so that by averaging those 2 values, the 14-bit value (DATA[13:0]) conversion is accurate.

To operate, START event must be configured to generate 16 events for each DATA[15:0] conversion and DATABUF must be loaded every 16 DAC conversions. EMPTY event and DMA request are therefore generated every 16 DATABUF to DATA transfer.

Writing a one to the Left Adjust bit in Control B register (CTRLB.LEFTADJ) change the data to DATA[15:6] and the dithering bits to DATA[5:2]. Refer to [DATA](#) on page 1053 description for further details.

Following timing diagram shows examples with DATA[15:0] = 0x1210 then DATA[15:0] = 0x12E0 and CTRLB.LEFTADJ=1.

Figure 42-2. DAC Conversions in Dithering Mode (CTRLB.LEFTADJ=1)



## 42.7. Register Summary

Offset	Name	Bit Pos.							
0x00	CTRLA	7:0		RUNSTDBY				ENABLE	SWRST
0x01	CTRLB	7:0	REFSEL[1:0]	DITHER		VPD	LEFTADJ	IOEN	EOEN
0x02	EVCTRL	7:0					INVEI	EMPTYEO	STARTEI
0x03	Reserved								
0x04	INTENCLR	7:0						EMPTY	UNDERRUN
0x05	INTENSET	7:0						EMPTY	UNDERRUN
0x06	INTFLAG	7:0						EMPTY	UNDERRUN
0x07	STATUS	7:0							READY
0x08	DATA	7:0	DATA[7:0]						
0x09		15:8	DATA[15:8]						
0x0A ... 0x0B	Reserved								
0x0C	DATABUF	7:0	DATABUF[7:0]						
0x0D		15:8	DATABUF[15:8]						
0x0E ... 0x0F	Reserved								
0x10	SYNCBUSY	7:0				DATABUF	DATA	ENABLE	SWRST
0x11		15:8							
0x12		23:16							
0x13		31:24							
0x14 ... 0x17	Reserved								
0x18	DBGCTRL	7:0							DBGRUN

## 42.8. 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](#) on page 1039.

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](#) on page 1041.

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.



## 42.8.1. Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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

### Bit 6 – 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.

## 42.8.2. 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]		DITHER		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	VREF	Internal voltage reference
0x1	VDDANA	Analog voltage supply
0x2	VREFFP	External reference
0x3		Reserved

### Bit 5 – DITHER: Dithering Mode

This bit controls dithering operation according to [Dithering mode](#) on page 1042.

Value	Description
0	Dithering mode is disabled.
1	Dithering mode is enabled.

### 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**

Value	Description
0	The DAC output is turned off.
1	The high-drive output buffer drives the DAC output to the V <sub>OUT</sub> pin.

### 42.8.3. Event Control

**Name:** EVCTRL  
**Offset:** 0x02  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### Bit 2 – INVEI: Enable Inversion Data Buffer Empty Event Output

This bit defines the edge detection of the input event for STARTEI.

Value	Description
0	Rising edge.
1	Falling edge.

#### 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.

#### 42.8.4. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x04

**Reset:** 0x00

**Property:** PAC Write-Protection

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

##### 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.

#### 42.8.5. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x05

**Reset:** 0x00

**Property:** PAC Write-Protection

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

##### 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.

## 42.8.6. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

### 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.

### 42.8.7. Status

**Name:** STATUS  
**Offset:** 0x07  
**Reset:** 0x00  
**Property:** -

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

#### Bit 0 – READY: DAC Ready

Value	Description
0	DAC is not ready for conversion.
1	Startup time has elapsed, DAC is ready for conversion.



## 42.8.8. 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	R/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 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.

Four additional bits are also used for the dithering feature according to [Dithering mode](#) on page 1042.

**Table 42-1. Valid Data Bits**

CTRLB.DITHER	CTRLB.LEFTADJ	DATA	Description
0	0	DATA[9:0]	Right adjusted, 10-bits
0	1	DATA[15:6]	Left adjusted, 10-bits
1	0	DATA[13:4], DATA[3:0]	Right adjusted, 14-bits
1	1	DATA[15:6], DATA[5:2]	Left adjusted, 14-bits

## 42.8.9. 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.

## 42.8.10. Synchronization Busy

**Name:** SYNCBUSY  
**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								
Reset								
Bit	15	14	13	12	11	10	9	8
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
Access					R	R	R	R
Reset					0	0	0	0

### Bit 3 – DATABUF: Data Buffer DAC0

This bit is set when DATABUF register is written.

This bit is cleared when DATABUF synchronization is completed.

Value	Description
0	No ongoing synchronized access.
1	Synchronized access is ongoing.

### Bit 2 – DATA: Data

This bit is set when DATA register is written.

This bit is cleared when DATA synchronization is completed.

Value	Description
0	No ongoing synchronized access.
1	Synchronized access is ongoing.

### Bit 1 – ENABLE: DAC Enable Status

This bit is set when CTRLA.ENABLE bit is written.

This bit is cleared when CTRLA.ENABLE synchronization is completed.

Value	Description
0	No ongoing synchronization.
1	Synchronization is ongoing.

**Bit 0 – SWRST: Software Reset**

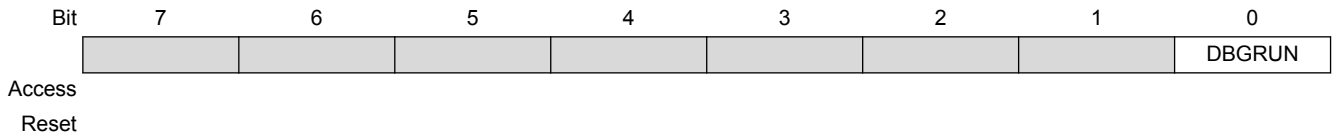
This bit is set when CTRLA.SWRST bit is written.

This bit is cleared when CTRLA.SWRST synchronization is completed.

Value	Description
0	No ongoing synchronization.
1	Synchronization is ongoing.

### 42.8.11. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x18  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### 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 DAC is halted when the CPU is halted by an external debugger. Any ongoing conversion will complete.
1	The DAC continues normal operation when the CPU is halted by an external debugger.

## 43. PTC - Peripheral Touch Controller

### 43.1. Overview

The Peripheral Touch Controller (PTC) acquires signals in order to detect touch on 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 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 self-capacitance mode, the PTC requires only one pin (Y-line) for each touch sensor.

### 43.2. Features

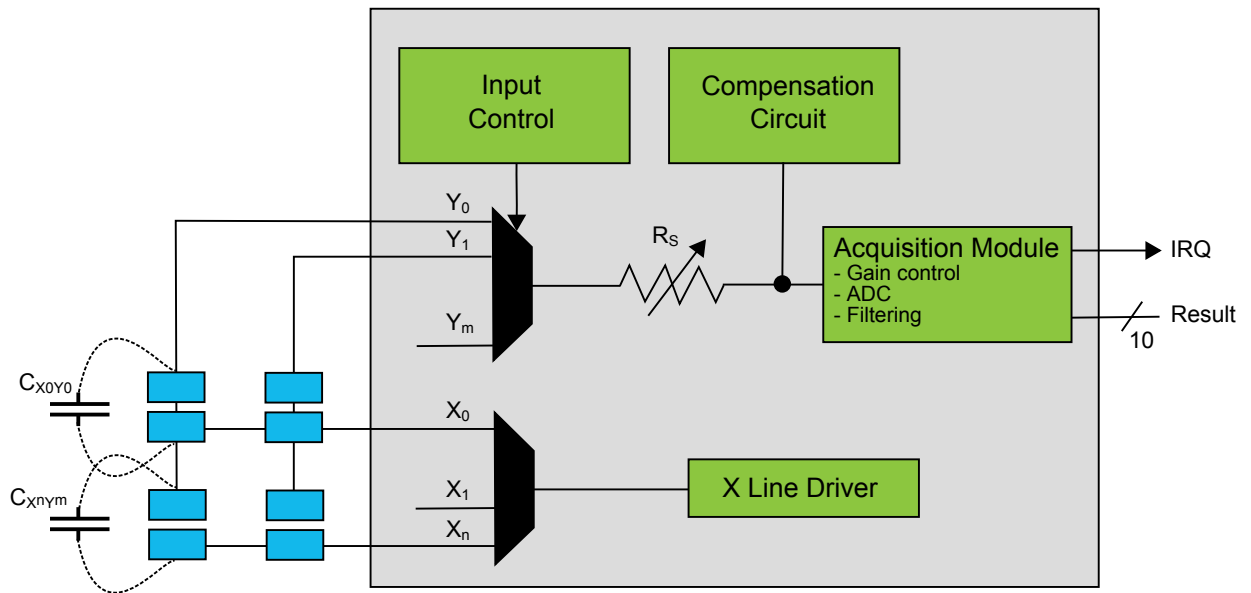
- Low-power, high-sensitivity, environmentally robust capacitive touch buttons, sliders, wheels and proximity sensing
- Supports mutual capacitance and self-capacitance sensing
  - Mix-and-match mutual-and self-capacitance sensors
- One pin per electrode – no external components
- Load compensating charge sensing
  - Parasitic capacitance compensation and adjustable gain for superior sensitivity
- Zero drift over the temperature and  $V_{DD}$  range
  - Auto calibration and re-calibration of sensors
- Single-shot and free-running charge measurement
- Hardware noise filtering and noise signal de-synchronization 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
  - 5% CPU utilization scanning 10 channels at 50ms scan rate
- Supported by the Atmel® QTouch® Composer development tool, which comprises QTouch Library project builder and QTouch analyzer

#### Related Links

[Configuration Summary](#) on page 14

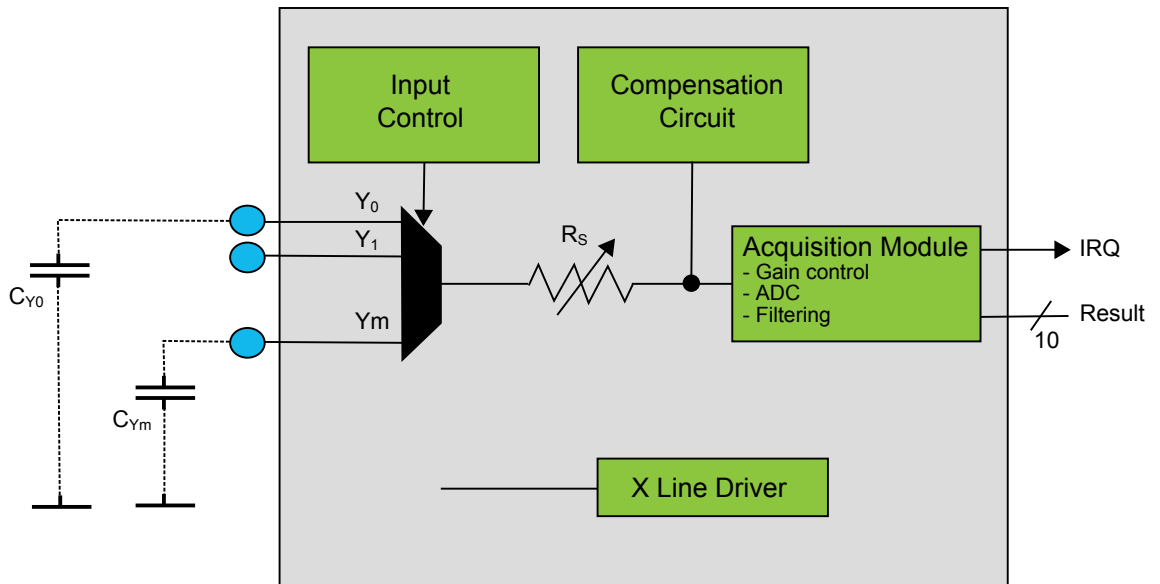
### 43.3. Block Diagram

Figure 43-1. PTC Block Diagram Mutual-Capacitance



**Note:** For SAM C21 the  $R_S = 0, 20, 50, 100 \text{ K}\Omega$ .

Figure 43-2. PTC Block Diagram Self-Capacitance



**Note:** For SAM C21 the  $R_S = 0, 20, 50, 100 \text{ K}\Omega$ .

## 43.4. Signal Description

Name	Type	Description
X[n:0]	Digital	X-line (Output)
Y[m:0]	Analog	Y-line (Input/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

[Configuration Summary](#) on page 14

## 43.5. Product Dependencies

In order to use this Peripheral, configure the other components of the system as described in the following sections.

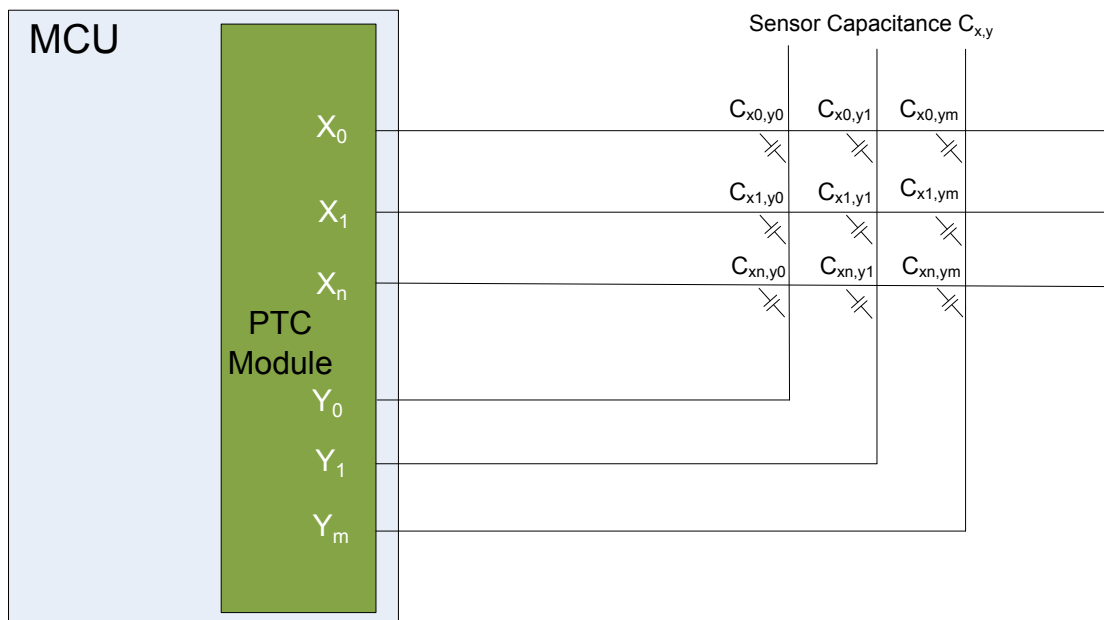
### 43.5.1. I/O Lines

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 K $\Omega$  can be used on X-lines and Y-lines.

#### 43.5.1.1. Mutual-capacitance Sensor Arrangement

A mutual-capacitance sensor is formed between two I/O lines - an X electrode for transmitting and Y electrode for receiving. The mutual capacitance between the X and Y electrode is measured by the Peripheral Touch Controller.

**Figure 43-3. Mutual Capacitance Sensor Arrangement**

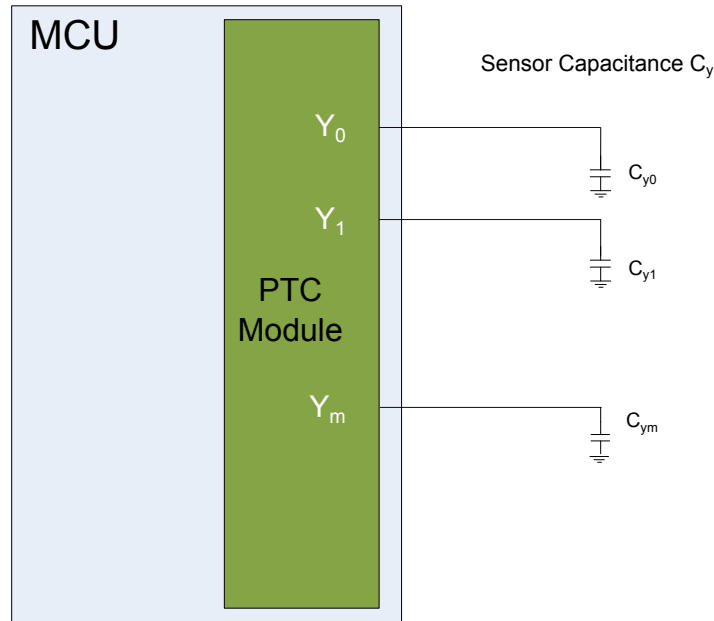




### 43.5.1.2. Self-capacitance Sensor Arrangement

The self-capacitance sensor is connected to a single pin on the Peripheral Touch Controller through the Y electrode for receiving the signal. The sense electrode capacitance is measured by the Peripheral Touch Controller.

Figure 43-4. Self-capacitance Sensor Arrangement



For more information about designing the touch sensor, refer to Buttons, Sliders and Wheels Touch Sensor Design Guide on <http://www.atmel.com>.

### 43.5.2. 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.

For more details, refer to *PM – Power Manager*.

#### Related Links

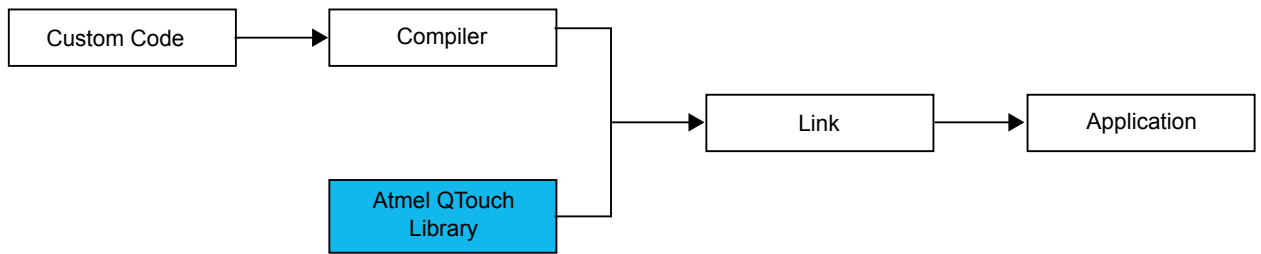
[GCLK - Generic Clock Controller](#) on page 114

[PM – Power Manager](#) on page 162

## 43.6. Functional Description

In order to access the PTC, the user must use the QTouch Composer tool to configure and link the QTouch Library firmware with the application code. QTouch Library can be used to implement buttons, sliders, wheels and proximity sensor in a variety of combinations on a single interface.

**Figure 43-5. QTouch Library Usage**



For more information about QTouch Library, refer to the [Atmel QTouch Library Peripheral Touch Controller User Guide](#).

## 44. TSENS – Temperature Sensor

### 44.1. Overview

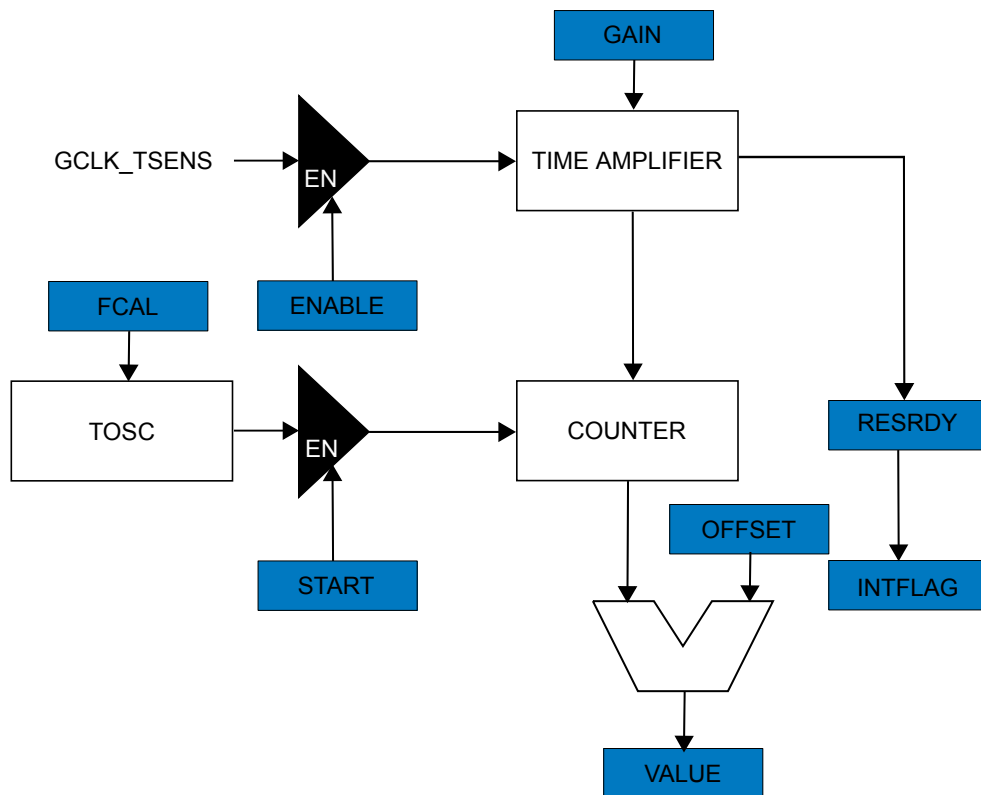
The Temperature Sensor (TSENS) can be used to accurately measure the operating temperature of the device.

### 44.2. Features

- Accurately measures a temperature
  - $\pm 1^{\circ}\text{C}$  over  $0^{\circ}\text{C}$  to  $60^{\circ}\text{C}$
  - $\pm 3^{\circ}\text{C}$  over  $-40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$
  - $\pm 5^{\circ}\text{C}$  over  $-40^{\circ}\text{C}$  to  $105^{\circ}\text{C}$
- A selectable reference clock source

### 44.3. Block Diagram

Figure 44-1. Temperature Sensor Block Diagram.



### 44.4. Signal Description

Not applicable.

## 44.5. Product Dependencies

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

### 44.5.1. I/O Lines

Not applicable.

### 44.5.2. Power Management

The TSENS will continue to operate in any sleep mode where the selected source clock is running. The TSENS's interrupts can be used to wake up the device from sleep modes. Events connected to the event system can trigger other operations in the system without exiting sleep modes. Refer to the Power Manager chapter for details on the different sleep modes.

### 44.5.3. Clocks

The TSENS bus clock (CLK\_TSENS\_APB) can be enabled and disabled in the Main Clock module, and the default state of CLK\_TSENS\_APB can be found in the Peripheral Clock Masking section.

A generic clock (GCLK\_TSENS) is required to clock the TSENS. This clock must be configured and enabled in the generic clock controller before using the TSENS.

This generic clock is asynchronous to the bus clock (CLK\_TSENS\_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to [Synchronization](#) on page 1068 for details.

#### Related Links

[Peripheral Clock Masking](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

### 44.5.4. DMA

The DMA request line is connected to the DMA Controller (DMAC). Using the TSENS Controller DMA request requires the DMA Controller to be configured first.

#### Related Links

[DMAC – Direct Memory Access Controller](#) on page 334

### 44.5.5. Interrupts

The interrupt request lines are connected to the interrupt controller. Using the TSENS interrupts requires the interrupt controller to be configured first.

#### Related Links

[Nested Vector Interrupt Controller](#) on page 41

### 44.5.6. Events

The events are connected to the Event System. Refer to the Event System section for details on how to configure the Event System.

#### Related Links

[EVSYS – Event System](#) on page 474

### 44.5.7. Debug Operation

When the CPU is halted in debug mode the TSENS will halt normal operation. Any on-going measurements will be completed. The TSENS can be forced to continue operation during debugging. Refer to [DBGCTRL](#) on page 1089 for details.

### 44.5.8. Register Access Protection

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

- Control B ([CTRLB](#) on page 1073) register
- Interrupt Flag Status and Clear ([INTFLAG](#) on page 1080) register

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

Write-protection does not apply to accesses through an external debugger. Refer to the Peripheral Access Controller chapter for details.

### 44.5.9. Calibration

The GAIN, OFFSET, FCAL, and TCAL calibration values from the production test must be loaded from the NVM Temperature Calibration Area into the TSENS Gain register (GAIN), Offset register (OFFSET) and Calibration register (CAL) by software to achieve specified accuracy.

#### Related Links

[NVM Software Calibration Area Mapping](#) on page 37

## 44.6. Functional Description

### 44.6.1. Principle of Operation

The TSENS accurately measures the operating temperature of the device by comparing the difference in two temperature dependent frequencies to a known frequency. The frequency of the temperature dependent oscillator (TOSC) is measured twice: first with the min configuration and next with the max configuration. The number of periods of GCLK\_TSENS used for the measurement is defined by the GAIN register. The width of the resulting pulse is measured using a counter clocked by GCLK\_TSENS in the up direction for the 1st phase and in the down 2nd phase.

The resulting signed value is proportional to the temperature and is corrected for offset by the contents of the OFFSET register.

$$\text{VALUE} = \text{OFFSET} + \text{GAIN} \times \left( \frac{f_{\text{TOSCMIN}}}{f_{\text{GCLK}}} + - \frac{f_{\text{TOSCMAX}}}{f_{\text{GCLK}}} \right)$$

**Note:** The values of GAIN and OFFSET are factory programmed to give a specific temperature slope when using the undivided internal 48MHz oscillator (OSC48M) as the GCLK\_TSENS source. Other frequencies/sources may be used, but the GAIN setting and/or expected slope will need to be scaled accordingly.

### 44.6.2. Basic Operation

#### 44.6.2.1. Initialization

The generic clocks (GCLK\_TSENS) should be configured and enabled. Refer to the Generic Clock Controller chapter for details.

The following bits are enable-protected, meaning that they can only be written when the TSENS is disabled ([CTRLA](#) on page 1071.ENABLE is zero):

- Run in Standby bit in Control A register ([CTRLA](#) on page 1071.RUNSTDBY)

The following registers are enable-protected:

- Control C ([CTRLC](#) on page 1074)

- Event Control ([EVCTRL](#) on page 1075)
- Window Monitor Lower Threshold ([WINLT](#) on page 1084)
- Window Monitor Upper Threshold ([WINUT](#) on page 1085)
- Gain Correction ([GAIN](#) on page 1086)
- Offset Correction ([OFFSET](#) on page 1087)
- Calibration ([CAL](#) on page 1088)

Enable-protection is denoted by the Enable-Protected property in the register description.

#### 44.6.2.2. Enabling, Disabling and Resetting

The TSENS is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The TSENS is disabled by writing a zero to CTRLA.ENABLE.

The TSENS is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the TSENS will be reset to their initial state, and the TSENS will be disabled. Refer to [CTRLA](#) on page 1071 for details.

#### 44.6.2.3. Measurement

After the TSENS is enabled, a measurement can be started either manually, by writing a one to the START bit in Control B register (CTRLB.START), or automatically by configuring an event input. A free-running mode can be used to continuously measure the temperature. When the Free running bit in the Control C register (CTRLC.FREERUN) is written to one, there is no need for a trigger to start the measurement. It will start automatically at the end of previous measurement.

The result of the measurement is stored in the Value register (VALUE), overwriting the result from the previous measurement and setting the Result Ready flag in the Interrupt Flag Status and Clear register (INTFLAG.RESRDY). To avoid data loss, the conversion result must be read as soon as it is available. Failing to do so will result in an overrun error condition, indicated by the OVERRUN bit in the Interrupt Flag Status and Clear register (INTFLAG.OVERRUN).

To use an interrupt handler, the corresponding bit in the Interrupt Enable Set register (INTENSET) must be written to one.

#### 44.6.2.4. Window Monitor

The window monitor feature allows the measurement result in the VALUE register to be compared to predefined threshold values. The window mode is selected by writing the Window Monitor Mode bits in the Control C register (CTRLC.WINMODE). Threshold values must be written in the Window Monitor Lower Threshold register (WINLT) and Window Monitor Upper Threshold register (WINUT).

#### 44.6.3. DMA Operation

The TSENS generates the following DMA request:

- Result Ready (RESRDY): the request is set when a measurement result is available, and cleared when the VALUE register is read. The request is generated independent of any Window Monitor condition.

#### Related Links

[DMAC – Direct Memory Access Controller](#) on page 334

#### 44.6.4. Interrupts

The TSENS has the following interrupt sources:

- Result Ready (RESRDY): Indicates when a measurement result is available.
- Window Monitor (WINMON): Generated when the measurement result matches the window monitor condition. Refer to [CTRLC](#) on page 1074 for details.

- **Overrun (OVERRUN):** Indicates that a new result is ready before the previous result has been read.
- **Overflow (OVF):** Indicates that the result is invalid because the result required more than 16 bits and overflowed the VALUE register.

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 TSENS is reset. See [INTFLAG](#) on page 1080 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](#) on page 41

#### 44.6.5. Events

The TSENS can generate the following output event:

- **Window Monitor (WINMON):** Generated when the measurement results matches the window monitor condition. Refer to [CTRLC](#) on page 1074 for details.

Writing a one to an Event Output bit in the Event Control Register (EVCTRL.WINEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. Refer to the Event System chapter for details on configuring the event system.

The ADC can take the following action on an input event:

- **Start measurement (START):** Start a measurement. Refer to [CTRLB](#) on page 1073 for details.

Writing a one to an Event Input bit into the Event Control register (EVCTRL.STARTEI) enables the corresponding action on input event. Writing a zero to this bit disables the corresponding action on input event. Refer to the Event System chapter for details. By default, the TSENS will detect a rising edge on the incoming event. If the TSENS action must be performed on the falling edge of the incoming event, the event line must be inverted first, by writing to one the corresponding Event Invert Enable bit in Event Control register (EVCTRL.STARTINV).

#### Related Links

[EVSYS – Event System](#) on page 474

#### 44.6.6. Sleep Mode Operation

The Run in Standby bit in the Control A register ([CTRLA](#) on page 1071.RUNSTDBY) controls the behavior of the TSENS during standby sleep mode, in cases where the TSENS is enabled (CTRLA.ENABLE = 1).

**Table 44-1. TSENS Sleep Behavior**

CTRLA.RUNSTDBY	CTRLC.FREERUN	CTRLA.ENABLE	Description
x	x	0	Disabled
0	0	1	Run in all sleep modes on request, except STANDBY.

CTRLA.RUNSTDBY	CTRLC.FREERUN	CTRLA.ENABLE	Description
0	1	1	Run in all sleep modes, except STANDBY.
1	0	1	Run in all sleep modes on request.
1	1	1	Run in all sleep modes.

#### 44.6.7. Synchronization

Due to the asynchronicity between the main clock domain (CLK\_TSENS\_APB) and the peripheral clock domain (GCLK\_TSENS) some registers are synchronized when written. When a write-synchronized register is written, the corresponding bit in the Synchronization Busy register (SYNCBUSY) is set immediately. When the write-synchronization is complete, this bit is cleared. Reading a write-synchronized register while the synchronization is ongoing will return the value written, and not the current value in the peripheral clock domain. To read the current value in the peripheral clock domain after writing a register, the user must wait for the corresponding SYNCBUSY bit to be cleared before reading the value.

If an operation that require synchronization is executed while its busy bit is on, the operation is discarded and a bus error is generated.

The following bits need synchronization when written:

- Software Reset bit in Control A register ([CTRLA](#) on page 1071.SWRST)
- Enable bit in Control A register ([CTRLA](#) on page 1071.ENABLE)

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



## 44.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		RUNSTDBY					ENABLE	SWRST
0x01	CTRLB	7:0								START
0x02	CTRLC	7:0				FREERUN			WINMODE[2:0]	
0x03	EVCTRL	7:0						WINEO	STARTINV	STARTEI
0x04	INTENCLR	7:0					OVF	WINMON	OVERRUN	RESRDY
0x05	INTENSET	7:0					OVF	WINMON	OVERRUN	RESRDY
0x06	INTFLAG	7:0					OVF	WINMON	OVERRUN	RESRDY
0x07	STATUS	7:0								OVF
0x08	SYNCBUSY	7:0							ENABLE	SWRST
0x09		15:8								
0x0A		23:16								
0x0B		31:24								
0x0C	VALUE	7:0	VALUE[7:0]							
0x0D		15:8	VALUE[15:8]							
0x0E		23:16	VALUE[23:16]							
0x0F		31:24								
0x10	WINLT	7:0	WINLT[7:0]							
0x11		15:8	WINLT[15:8]							
0x12		23:16	WINLT[23:16]							
0x13		31:24								
0x14	WINUT	7:0	WINUT[7:0]							
0x15		15:8	WINUT[15:8]							
0x16		23:16	WINUT[23:16]							
0x17		31:24								
0x18	GAIN	7:0	GAIN[7:0]							
0x19		15:8	GAIN[15:8]							
0x1A		23:16	GAIN[23:16]							
0x1B		31:24								
0x1C	OFFSET	7:0	OFFSETC[7:0]							
0x1D		15:8	OFFSETC[15:8]							
0x1E		23:16	OFFSETC[23:16]							
0x1F		31:24								
0x20	CAL	7:0	FCAL[5:0]							
0x21		15:8	TCAL[5:0]							
0x22		23:16								
0x23		31:24								
0x24	DBGCTRL	7:0								DBGRUN

## 44.8. 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.

## 44.8.1. Control A

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized (ENABLE, SWRST)

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

### Bit 6 – RUNSTDBY: Run in Standby

This bit controls how the TSENS behaves during standby sleep mode:

This bit is not synchronized.

Value	Description
0	The TSENS is halted during standby sleep mode.
1	The TSENS is not stopped in standby sleep mode. If CTRLC.FREERUN is zero, the TSENS will be running when a peripheral is requesting it. If CTRLC.FREERUN is one, the TSENS will always be running in standby sleep mode.

### 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 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.
1	The peripheral 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 TSENS, except GAIN, OFFSET, CAL and DBGCTRL, to their initial state, and the TSENS 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 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.

## 44.8.2. Control B

**Name:** CTRLB  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** –

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

### Bit 0 – START: Start Measurement

Value	Description
0	Writing a zero to this bit has no effect.
1	Writing a one to this bit starts a measurement

### 44.8.3. Control C

**Name:** CTRLC  
**Offset:** 0x02  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-protected

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

#### Bit 4 – FREERUN: Free Running Measurement

Value	Description
0	TSENS operates in single measurement mode.
1	TSENS is in free running mode and a new measurement will be initiated when the previous measurement completes.

#### Bits 2:0 – WINMODE[2:0]: Window Monitor Mode

These bits enable and define the window monitor mode.

Value	Name	Description
0x0	DISABLE	No window mode (default)
0x1	ABOVE	VALUE > WINLT
0x2	BELOW	VALUE < WINUT
0x3	INSIDE	WINLT < VALUE < WINUT
0x4	OUTSIDE	WINUT < VALUE < WINLT
0x5	HYST_ABOVE	VALUE > WINUT with hysteresis to WINLT
0x6	HYST_BELOW	VALUE < WINLT with hysteresis to WINUT
0x07		Reserved

#### 44.8.4. Event Control

**Name:** EVCTRL

**Offset:** 0x03

**Reset:** 0x00

**Property:** PAC Write-Protection, Enable-protected

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

##### Bit 2 – WINEO: 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 1 – STARTINV: Start Conversion Event Invert Enable

Value	Description
0	start event input source is not inverted.
1	start event input source is inverted.

##### Bit 0 – STARTEI: Start Conversion Event Input Enable

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.

#### 44.8.5. Interrupt Enable Clear

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

**Name:** INTENCLR

**Offset:** 0x04

**Reset:** 0x00

**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
					OVF	WINMON	OVERRUN	RESRDY
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

##### Bit 3 – 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, which disables the corresponding interrupt request.

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 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, which disables 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, which disables 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, which disables the corresponding interrupt request.

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.

#### 44.8.6. Interrupt Enable Set

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

**Name:** INTENSET

**Offset:** 0x05

**Reset:** 0x00

**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
					OVF	WINMON	OVERRUN	RESRDY
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

##### Bit 3 – 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, which enables the Overflow interrupt.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow 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, which enables 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 Enable bit, which enables the corresponding interrupt request.

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, which enables the Result Ready interrupt.

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

#### 44.8.7. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** –

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

##### Bit 3 – OVF: Overflow

This flag is cleared by writing a one to the flag.

This flag is set when the conversion result requires more than 16 bits and overflows the VALUE register, 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 2 – WINMON: Window Monitor

This flag is cleared by writing a one to the flag or by reading the VALUE register.

This flag is set on the next 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 a valid VALUE is updated before the previous valid value has been read by the CPU, and an interrupt 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 VALUE register.

This flag is set when the conversion result is available, and an interrupt will be generated if INTENCLR/SET.RESRDY is one.

This flag will not set if an overflow occurs during the conversion.

Writing a zero to this bit has no effect.

Writing a one to this bit clears the Result Ready interrupt flag.

#### 44.8.8. Status

**Name:** STATUS

**Offset:** 0x07

**Reset:** 0x00

**Property:** –

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

##### **Bit 0 – OVF: Result Overflow**

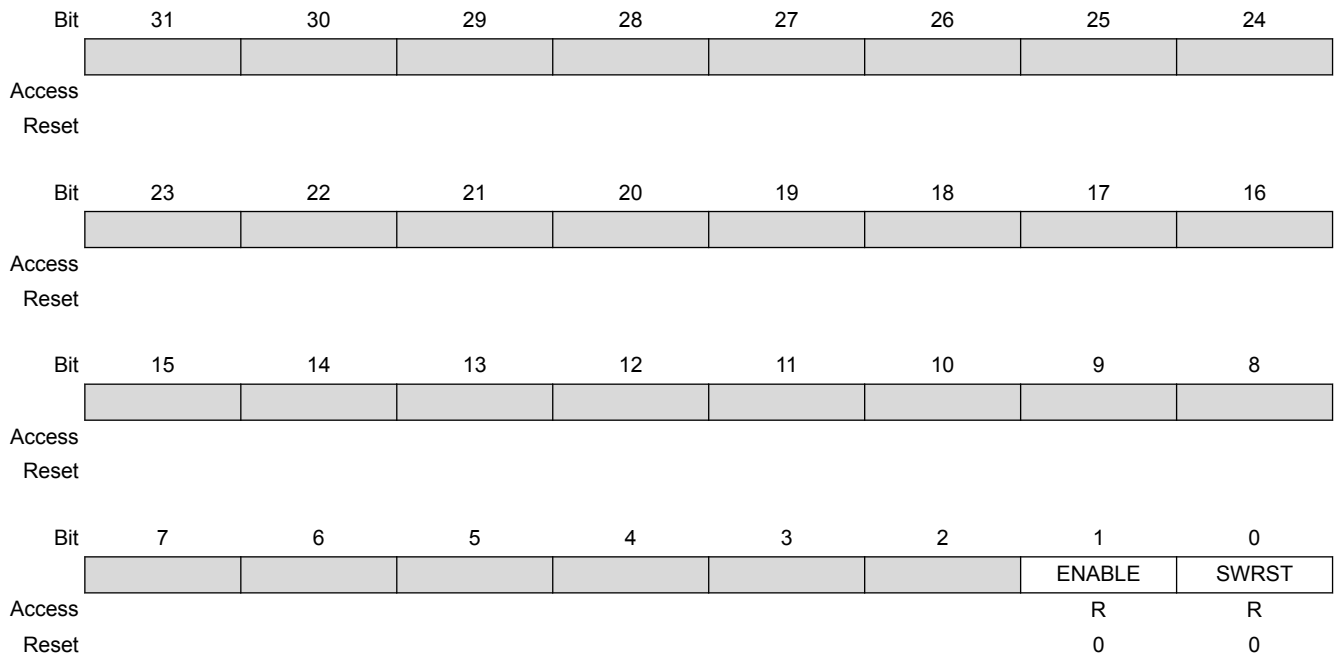
Writing a zero to this bit has no effect.

Writing a one to this bit has no effect.

Value	Description
0	No overflow in the VALUE register has occurred. The result is valid.
1	An overflow occurred in the VALUE register. The result is not valid.

#### 44.8.9. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** –



**Bit 1 – ENABLE: Enable Busy**

This bit is cleared when the synchronization of CTRLA.ENABLE is complete.

This bit is set when the synchronization of CTRLA.ENABLE is started.

**Bit 0 – SWRST: Software Reset Busy**

This bit is cleared when the synchronization of CTRLA.SWRST is complete.

This bit is set when the synchronization of CTRLA.SWRST is started.

#### 44.8.10. Value

**Name:** VALUE  
**Offset:** 0x0C  
**Reset:** 0x0000  
**Property:** –

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	VALUE[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
	VALUE[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
	VALUE[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### **Bits 23:0 – VALUE[23:0]: Measurement Value**

Result from measurement. This VALUE is in two's complement format.

#### 44.8.11. Window Monitor Lower Threshold

**Name:** WINLT  
**Offset:** 0x10  
**Reset:** 0x0000  
**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
	WINLT[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
	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 23:0 – WINLT[23:0]: Window Lower Threshold

If the window monitor is enabled, these bits define the lower threshold value. This WINLT value is in two's complement format.



#### 44.8.12. Window Monitor Upper Threshold

**Name:** WINUT  
**Offset:** 0x14  
**Reset:** 0x0000  
**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
	WINUT[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
	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 23:0 – WINUT[23:0]: Window Upper Threshold

If the window monitor is enabled, these bits define the upper threshold value. This WINUT value is in two's complement format.

### 44.8.13. Gain

**Name:** GAIN

**Offset:** 0x18

**Reset:** 0x0000

**Property:** Enable-Protected, PAC Write-Protection, not reset by a software reset

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	GAIN[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
	GAIN[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
	GAIN[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 – GAIN[23:0]: Time Amplifier Gain**

This bitfield is loaded from NVM after a reset. The bitfield can also be written by CPU.

The GAIN value defines the number of GCLK\_TSENS periods that will be used for a measurement cycle.

#### 44.8.14. Offset

**Name:** OFFSET

**Offset:** 0x1C

**Reset:** 0x0000

**Property:** Enable-Protected, PAC Write-Protection, not reset by a software reset

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	OFFSETC[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
	OFFSETC[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
	OFFSETC[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 – OFFSETC[23:0]: Offset Correction

This bitfield is loaded from NVM after a reset. The bitfield can also be written by CPU.

These bits define how the TSENS measurement result is compensated for offset error before being written to the VALUE register. This OFFSET value is in two's complement format.

#### 44.8.15. Calibration

**Name:** CAL

**Offset:** 0x20

**Reset:** 0x00000000

**Property:** Enable-Protected, PAC Write-Protection, not reset by a software reset

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			TCAL[5:0]					
Reset			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
Access			FCAL[5:0]					
Reset			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

##### Bits 13:8 – TCAL[5:0]: Temperature Calibration

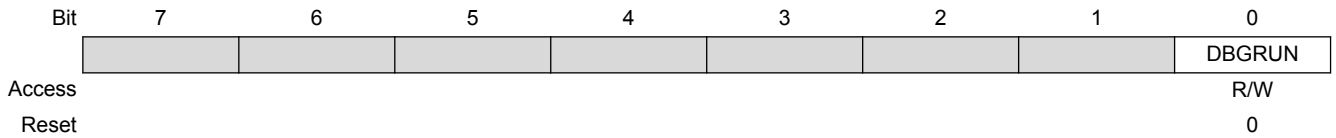
This value from production test must be loaded from the NVM software calibration row into the CAL register by software to achieve the specified accuracy. The value must be copied only, and must not be changed.

##### Bits 5:0 – FCAL[5:0]: Frequency Calibration

This value from production test must be loaded from the NVM software calibration row into the CAL register by software to achieve the specified accuracy. The value must be copied only, and must not be changed.

#### 44.8.16. Debug Control

**Name:** DBGCTRL  
**Offset:** 0x24  
**Reset:** 0x00  
**Property:** –



##### 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 TSENS is halted when the CPU is halted by an external debugger. Any on-going measurement will complete.
1	The TSENS continues normal operation when the CPU is halted by an external debugger.

## 45. FREQM – Frequency Meter

### 45.1. Overview

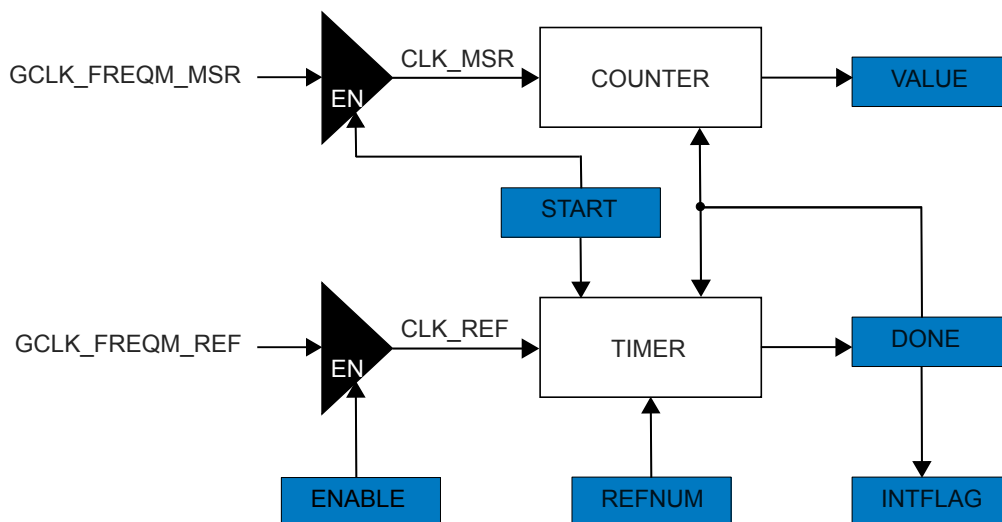
The Frequency Meter (FREQM) can be used to accurately measure the frequency of a clock by comparing it to a known reference clock.

### 45.2. Features

- Accurately measures a clock frequency
- A selectable reference clock from GCLK\_FREQM\_REF sources
- A selectable clock from GCLK\_FREQM\_MSR sources can be measured
- Ratio can be measured with 24-bit accuracy

### 45.3. Block Diagram

Figure 45-1. FREQM Block Diagram.



### 45.4. Signal Description

Not applicable.

### 45.5. Product Dependencies

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

#### 45.5.1. I/O Lines

Not applicable.

#### 45.5.2. Power Management

The FREQM will continue to operate in any sleep mode where the selected source clock is running. The FREQM's interrupts can be used to wake up the device from sleep modes. Refer to the Power Manager chapter for details on the different sleep modes.

#### 45.5.3. Clocks

The clock for the FREQM bus interface (CLK\_APB\_FREQM) is enabled and disabled by the Main Clock Controller, the default state of CLK\_APB\_FREQM can be found in the Peripheral Clock Masking section.

Two generic clocks are used by the FREQM (GCLK\_FREQM\_REF and GCLK\_FREQM\_MSR). The reference clock (GCLK\_FREQM\_REF) is required to clock the internal reference timer while operating as a frequency reference, while the measurement clock (GCLK\_FREQM\_MSR) is required to clock a ripple counter for frequency measurement. These clocks must be configured and enabled in the generic clock controller before using the FREQM.

##### Related Links

[MCLK – Main Clock](#) on page 131

[Peripheral Clock Masking](#) on page 135

[GCLK - Generic Clock Controller](#) on page 114

#### 45.5.4. DMA

Not applicable.

#### 45.5.5. Interrupts

The interrupt request line is connected to the interrupt controller. Using FREQM interrupt requires the interrupt controller to be configured first.

##### Related Links

[Nested Vector Interrupt Controller](#) on page 41

#### 45.5.6. Events

Not applicable

#### 45.5.7. Debug Operation

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

#### 45.5.8. Register Access Protection

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

- Control B register (CTRLB)
- 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.

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

## Related Links

[PAC - Peripheral Access Controller](#) on page 46

## 45.6. Functional Description

### 45.6.1. Principle of Operation

During a period of  $\text{REFNUM}/f_{\text{CLK\_REF}}$ , the FREQM is counting the number of periods of the clock to be measured, VALUE. Here, REFNUM is the Number of Reference Clock Cycles selected in the Configuration A register (CFG.A.REFNUM), VALUE is the Measurement result stored to the Value register (VALUE.VALUE), and  $f_{\text{CLK\_REF}}$  is the frequency of the reference clock.

The frequency of the measured clock,  $f_{\text{CLK\_MSR}}$ , is calculated by

$$f_{\text{CLK\_MSR}} = \left( \frac{\text{VALUE}}{\text{REFNUM}} \right) f_{\text{CLK\_REF}}$$

### 45.6.2. Basic Operation

#### 45.6.2.1. Initialization

Before enabling FREQM, the device and peripheral must be configured:

- Each of the generic clocks (GCLK\_FREQM\_REF and GCLK\_FREQM\_MSR) should be configured and enabled. Note that the reference clock should be slower than the measurement clock.
- The Number of Reference Clock Cycles value in the Configuration A register (CFG.A.REFNUM) must be written to a value greater than 0x00.

The following register is enable-protected, meaning that it can only be written when the FREQM is disabled (CTRLA.ENABLE is zero):

- Configuration A register (CFG.A)

Enable-protection is denoted by the "Enable-Protected" property in the register description.

## Related Links

[GCLK - Generic Clock Controller](#) on page 114

#### 45.6.2.2. Enabling, Disabling and Resetting

The FREQM is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The peripheral is disabled by writing CTRLA.ENABLE=0.

The FREQM is reset by writing a '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the FREQM will be reset to their initial state, and the FREQM will be disabled.

#### 45.6.2.3. Measurement

In the Configuration A register, the Number of Reference Clock Cycles field (CFG.A.REFNUM) selects the duration of the measurement. The measurement is given in number of GCLK\_FREQM\_REF periods.

**Note:** This field must be written before the FREQM is enabled.

After the FREQM is enabled, writing a '1' to the START bit in the Control B register (CTRLB.START) starts the measurement. The BUSY bit in Status register (STATUS.BUSY) is cleared when the measurement is done.

There is also an interrupt request for Measurement Done: When the Measurement Done bit in Interrupt Enable Set register (INTENSET.DONE) is '1' and a measurement is finished, the Measurement Done bit in the Interrupt Flag Status and Clear register (INTFLAG.DONE) will be set and an interrupt request is generated.



The result of the measurement can be read from the Value register ([VALUE.VALUE](#)). The frequency of the measured clock GCLK\_FREQM\_MSR is then:

$$f_{\text{CLK\_MSR}} = \left( \frac{\text{VALUE}}{\text{REFNUM}} \right) f_{\text{CLK\_REF}}$$

**Note:** In order to make sure the measurement result ([VALUE.VALUE](#)) is valid, the overflow status ([STATUS.OVF](#)) should be checked.

In case an overflow condition occurred, indicated by the Overflow bit in the STATUS register ([STATUS.OVF](#)), either the number of reference clock cycles must be reduced ([CFGA.REFNUM](#)), or a faster reference clock must be configured. Once the configuration is adjusted, clear the overflow status by writing a '1' to [STATUS.OVF](#). Then another measurement can be started by writing a '1' to [CTRLB.START](#).

### 45.6.3. DMA Operation

Not applicable.

### 45.6.4. Interrupts

The FREQM has one interrupt source:

- DONE: A frequency measurement is done

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear ([INTFLAG](#) on page 1101) 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](#) on page 1100) register, and disabled by writing a '1' to the corresponding bit in the Interrupt Enable Clear ([INTENCLR](#) on page 1099) 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 FREQM is reset. See [INTFLAG](#) on page 1101 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](#) on page 1101 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](#) on page 41

### 45.6.5. Events

Not applicable.

### 45.6.6. Sleep Mode Operation

The FREQM will continue to operate in any sleep mode where the selected source clock is running. The FREQM's interrupts can be used to wake up the device from sleep modes.

For lowest chip power consumption in sleep modes, FREQM should be disabled before entering a sleep mode.

#### Related Links

[PM – Power Manager](#) on page 162

### 45.6.7. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read. See the *Clock System Register Synchronization* for

details. An exception is the Channel Enable bit in the Peripheral Channel Control registers (PCHCTRLm.CHEN). When changing this bit, the bit value must be read-back to ensure the synchronization is complete and to assert glitch free internal operation. Note that changing the bit value under ongoing synchronization will *not* generate an error.

The following bits and registers are write-synchronized:

- Software Reset bit in Control A register (CTRLA.SWRST)
- Enable bit in Control A register (CTRLA.ENABLE)

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

#### **Related Links**

[Register Synchronization](#) on page 110

## 45.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0						ENABLE	SWRST	
0x01	CTRLB	7:0							START	
0x02	CFGA	7:0	REFNUM[7:0]							
0x03		15:8								
0x04 ... 0x07	Reserved									
0x08	INTENCLR	7:0							DONE	
0x09	INTENSET	7:0							DONE	
0x0A	INTFLAG	7:0							DONE	
0x0B	STATUS	7:0						OVF	BUSY	
0x0C	SYNCBUSY	7:0						ENABLE	SWRST	
0x0D		15:8								
0x0E		23:16								
0x0F		31:24								
0x10	VALUE	7:0	VALUE[7:0]							
0x11		15:8	VALUE[15:8]							
0x12		23:16	VALUE[23:16]							
0x13		31:24								

## 45.8. 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.

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.

## 45.8.1. Control A

**Name:** CTRLA

**Offset:** 0x00

**Reset:** 0x00

**Property:** PAC Write-Protection, Write-Synchronized, Read-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 CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the ENABLE 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.
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 FREQM to their initial state, and the FREQM will be disabled. Writing a '1' to this bit 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.

This bit is not enable-protected.

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

## 45.8.2. Control B

**Name:** CTRLB

**Offset:** 0x01

**Reset:** 0x00

**Property:** –

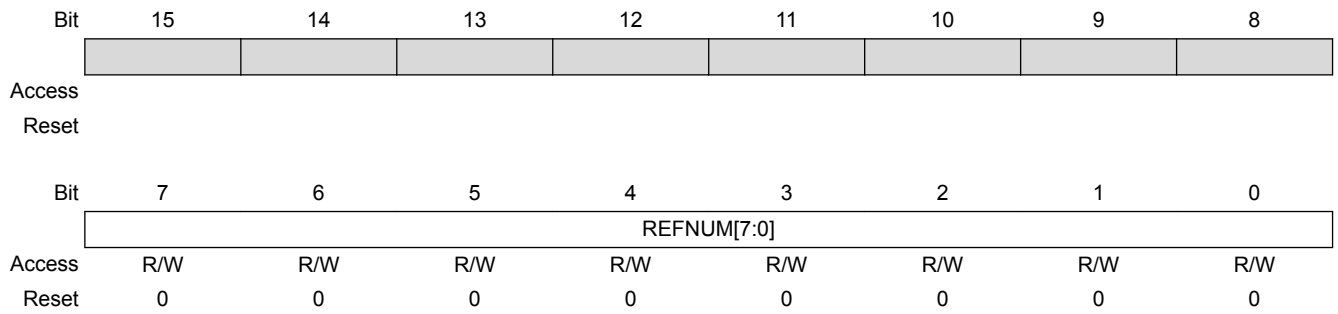
Bit	7	6	5	4	3	2	1	0
								START
Access								W
Reset								0

### Bit 0 – START: Start Measurement

Value	Description
0	Writing a '0' has no effect.
1	Writing a '1' starts a measurement.

### 45.8.3. Configuration A

**Name:** CFGA  
**Offset:** 0x02  
**Reset:** 0x0000  
**Property:** Write-Protected, Enable-protected



**Bits 7:0 – REFNUM[7:0]: Number of Reference Clock Cycles**

Selects the duration of a measurement in number of CLK\_FREQM\_REF cycles. This must be a non-zero value, i.e. 0x01 (one cycle) to 0xFF (255 cycles).

#### 45.8.4. Interrupt Enable Clear

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

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

##### **Bit 0 – DONE: Measurement Done Interrupt Enable**

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

Writing a '1' to this bit will clear the Measurement Done Interrupt Enable bit, which disables the Measurement Done interrupt.

Value	Description
0	The Measurement Done interrupt is disabled.
1	The Measurement Done interrupt is enabled.

### 45.8.5. Interrupt Enable Set

**Name:** INTENSET

**Offset:** 0x09

**Reset:** 0x00

**Property:** Write-Protected

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

#### Bit 0 – DONE: Measurement Done Interrupt Enable

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

Writing a '1' to this bit will set the Measurement Done Interrupt Enable bit, which enables the Measurement Done interrupt.

Value	Description
0	The Measurement Done interrupt is disabled.
1	The Measurement Done interrupt is enabled.



## 45.8.6. Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** –

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

### Bit 0 – DONE: Measurement Done

This flag is cleared by writing a '1' to it.

This flag is set when the STATUS.BUSY bit has a one-to-zero transition.

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

Writing a '1' to this bit will clear the DONE interrupt flag.

## 45.8.7. Status

**Name:** STATUS  
**Offset:** 0x0B  
**Reset:** 0x00  
**Property:** –

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

### Bit 1 – OVF: Sticky Count Value Overflow

This bit is cleared by writing a '1' to it.

This bit is set when an overflow condition occurs to the value counter.

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

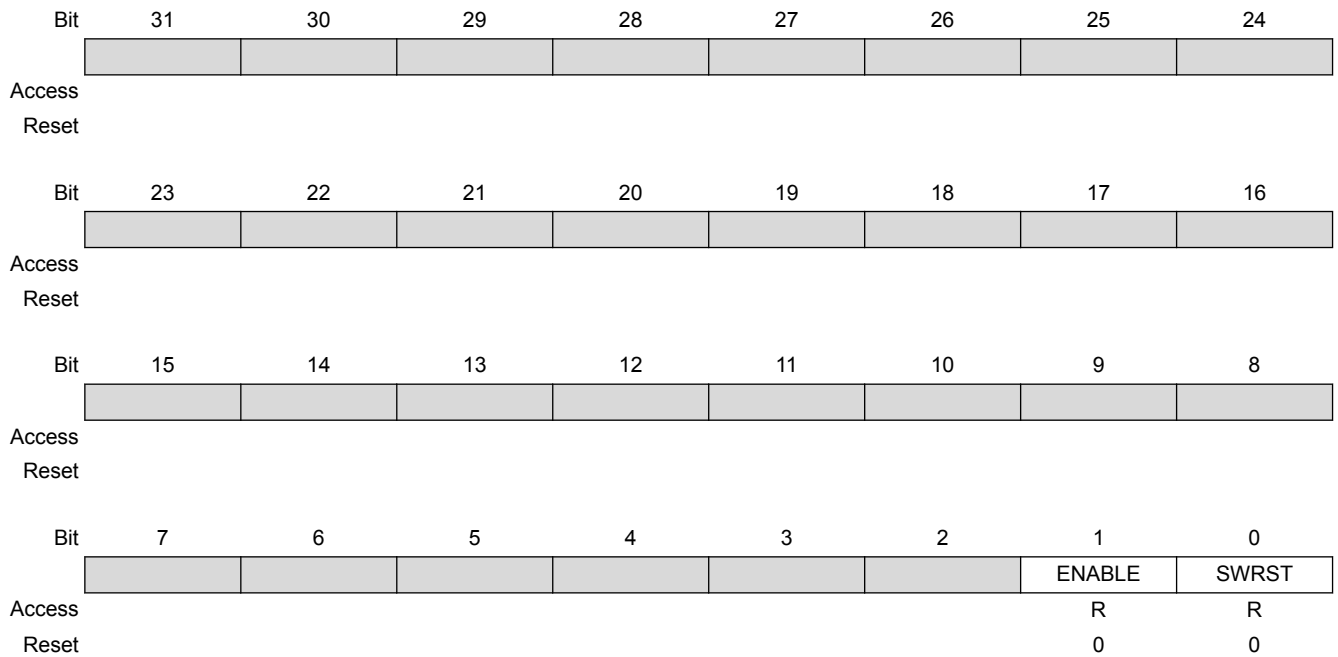
Writing a '1' to this bit will clear the OVF status.

### Bit 0 – BUSY: FREQM Status

Value	Description
0	No frequency measurement ongoing.
1	Frequency measurement is ongoing.

## 45.8.8. Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x0C  
**Reset:** 0x00000000  
**Property:** –



### Bit 1 – ENABLE: Enable

This bit is cleared when the synchronization of CTRLA.ENABLE is complete.

This bit is set when the synchronization of CTRLA.ENABLE is started.

### Bit 0 – SWRST: Synchronization Busy

This bit is cleared when the synchronization of CTRLA.SWRST is complete.

This bit is set when the synchronization of CTRLA.SWRST is started.

## 45.8.9. Value

**Name:** VALUE  
**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
	VALUE[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
	VALUE[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
	VALUE[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

**Bits 23:0 – VALUE[23:0]: Measurement Value**  
 Result from measurement.

## 46. Electrical Characteristics

### 46.1. Disclaimer

All values in this chapter are preliminary and subject to change without further notice.

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.

### 46.2. Absolute Maximum Ratings

Stresses beyond those listed in the below table 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 46-1. Absolute maximum ratings**

Symbol	Parameter	Condition	Min.	Max.	Units
$V_{DD}$	Power supply voltage		0	5.5	V
$I_{VDD}$	Current into a $V_{DD}$ pin		-	92	mA
$I_{GND}$	Current out of a GND pin		-	130	mA
$V_{PIN}$	Pin voltage with respect to GND and $V_{DD}$		GND-0.3V	$V_{DD} + 0.3V$	V
$T_{STORAGE}$	Storage temperature		-60	150	$^{\circ}\text{C}$

**Table 46-2. GPIO Clusters**

Package	Cluster	GPIO	Supplies Pin connected to the cluster	
64 pins	1	PB31 PB30 PA31 PA30 PA28 PA27	VDDIN (56)	GND (54)
	2	PB23 PB22	VDDIO (48)	GND (54+47)
	3	PA25 PA24 PA23 PA22 PA21 PA20 PB17 PB16 PA19 PA18 PA17 PA16	VDDIO (48+34)	GND (47+33)
	4	PA15 PA14 PA13 PA12 PB15 PB14 PB13 PB12 PB11 PB10	VDDIO (34+21)	GND (33+22)
	5	PA11 PA10 PA08 PA09	VDDIO (21)	GND (22)
	6	PA07 PA06 PA05 PA04 PB09 PB08 PB07 PB06 PB05 PB04 PA03 PA02 PA01 PA00 PB03 PB02 PB01 PB00	VDDANA (8)	GNDANA (7)

Package	Cluster	GPIO	Supplies Pin connected to the cluster	
48 pins	1	PA31 PA30 PA28 PA27	VDDIN (44)	GND (42)
	2	PB23 PB22	VDDIO (36)	GND (42+35)
	3	PA25 PA24 PA23 PA22 PA21 PA20 PA19 PA18 PA17 PA16 PA15 PA14 PA13 PA12 PB11 PB10	VDDIO (36+17)	GND (35+18)
	4	PA11 PA10 PA08 PA09	VDDIO (17)	GND (18)
	5	PA07 PA06 PA05 PA04 PB09 PB08 PA03 PA02 PA01 PA00 PB03 PB02	VDDANA (6)	GNDANA (5)
32 pins	1	PA31 PA30 PA28 PA27	VDDIN (30)	GND (28)
	2	PA25 PA24 PA23 PA22 PA19 PA18 PA17 PA16 PA15 PA14 PA11 PA10 PA08 PA09	VDDIO (9)	GND (28+10)
	3	PA07 PA06 PA05 PA04 PA03 PA02 PA01 PA00	VDDANA (9)	GND (28+10)

### 46.3. General Operating Ratings

The device must operate within the ratings listed in below in order for all other electrical characteristics and typical characteristics of the device to be valid.

**Table 46-3. General operating conditions**

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V <sub>DDIN</sub>	Power supply voltage		2.7 <sup>(1)</sup>	5.0	5.5	V
V <sub>DDANA</sub>	Analog supply voltage		2.7 <sup>(1)</sup>	5.0	5.5	V
V <sub>DDIO</sub>	IO supply voltage		2.7 <sup>(1)</sup>	5.0	5.5	V
T <sub>A</sub>	Temperature range		-40	25	105	°C
T <sub>J</sub>	Junction temperature		-	-	125	°C

1. With BODVDD disabled. If the BODVDD is enabled, check [Table 46-14 BOD LEVEL Values\(1\)\(2\)](#) on page 1116

### 46.4. Supply Characteristics

The following characteristics are applicable to the operating temperature range: T<sub>A</sub> = -40°C to 105°C, unless otherwise specified and are valid for a junction temperature up to T<sub>J</sub> = 125°C.

**Table 46-4. Supply Characteristics**

Symbol	Conditions	Voltage		
		Min.	Max.	Units
V <sub>DDIO</sub> V <sub>DDIN</sub> V <sub>DDANA</sub>	Full Voltage Range	2.7	5.5	V

**Table 46-5. Supply Rise Rates**

Symbol	Parameter	Rise Rate	Units
		Max.	
V <sub>DDIO</sub>	DC supply peripheral I/Os, internal regulator and analog supply	0.1	V/μs
V <sub>DDIN</sub>		0.1	
V <sub>DDANA</sub>		0.1	

## 46.5. Maximum Clock Frequencies

**Table 46-6. Maximum GCLK Generator Output Frequencies**

Symbol	Condition	Max.	Units
f <sub>GCLKGEN0</sub> / f <sub>GCLK_MAIN</sub>	Undivided	96	MHz
f <sub>GCLKGEN1</sub>			
f <sub>GCLKGEN2</sub>			
f <sub>GCLKGEN3</sub>	Divided	66	MHz
f <sub>GCLKGEN4</sub>			
f <sub>GCLKGEN5</sub>			
f <sub>GCLKGEN6</sub>			
f <sub>GCLKGEN7</sub>			
f <sub>GCLKGEN8</sub>			

**Table 46-7. Maximum Peripheral Clock Frequencies**

Symbol	Description	Max.	Units
f <sub>CPU</sub>	CPU clock frequency	48	MHz
f <sub>AHB</sub>	AHB clock frequency	48	MHz
f <sub>APBA</sub>	APBA clock frequency	48	MHz
f <sub>APBB</sub>	APBB clock frequency	48	MHz
f <sub>APBC</sub>	APBC clock frequency	48	MHz
f <sub>GCLK_DPLL</sub>	FDPLL96M Reference clock frequency	2	MHz

Symbol	Description	Max.	Units
f <sub>GCLK_DPLL_32K</sub>	FDPLL96M 32k Reference clock frequency	32	kHz
f <sub>GCLK_EIC</sub>	EIC input clock frequency	48	MHz
f <sub>GCLK_FREQM_MSR</sub>	FREQM Measure	48	MHz
f <sub>GCLK_FREQM_REF</sub>	FREQM Reference	48	MHz
f <sub>GCLK_TSENS</sub>	TSENS input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_0</sub>	EVSYS channel 0 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_1</sub>	EVSYS channel 1 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_2</sub>	EVSYS channel 2 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_3</sub>	EVSYS channel 3 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_4</sub>	EVSYS channel 4 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_5</sub>	EVSYS channel 5 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_6</sub>	EVSYS channel 6 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_7</sub>	EVSYS channel 7 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_8</sub>	EVSYS channel 8 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_9</sub>	EVSYS channel 9 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_10</sub>	EVSYS channel 10 input clock frequency	48	MHz
f <sub>GCLK_EVSYS_CHANNEL_11</sub>	EVSYS channel 11 input clock frequency	48	MHz
f <sub>GCLK_SERCOMx_SLOW</sub>	Common SERCOM slow input clock frequency	5	MHz
f <sub>GCLK_SERCOM0_CORE</sub>	SERCOM0 input clock frequency	48	MHz
f <sub>GCLK_SERCOM1_CORE</sub>	SERCOM1 input clock frequency	48	MHz
f <sub>GCLK_SERCOM2_CORE</sub>	SERCOM2 input clock frequency	48	MHz
f <sub>GCLK_SERCOM3_CORE</sub>	SERCOM3 input clock frequency	48	MHz
f <sub>GCLK_SERCOM4_CORE</sub>	SERCOM4 input clock frequency	48	MHz
f <sub>GCLK_SERCOM5_CORE</sub>	SERCOM5 input clock frequency	48	MHz
f <sub>GCLK_CANn</sub>	CAN input clock frequency	48	MHz
f <sub>GCLK_TCC0, 1</sub>	TCCn input clock frequency	96	MHz
f <sub>GCLK_TCC2</sub>	TCC2 input clock frequency	48	MHz
f <sub>GCLK_TCn</sub>	TCn input clock frequency	48	MHz
f <sub>GCLK_ADCn</sub>	ADCn input clock frequency	48	MHz
f <sub>GCLK_SDADC</sub>	SDADC input clock frequency	48	MHz
f <sub>GCLK_DAC</sub>	DAC input clock frequency	48	kHz
f <sub>GCLK_PTC</sub>	PTC input clock frequency	48	MHz



Symbol	Description	Max.	Units
f <sub>GCLK_CCL</sub>	CCL input clock frequency	48	MHz
f <sub>GCLK_AC</sub>	AC digital input clock frequency	48	MHz

## 46.6. Power Consumption

The values in [Table 46-8 Current Consumption](#) on page 1110 are measured values of power consumption under the following conditions, except where noted:

- Operating conditions
  - V<sub>VDDIN</sub> = 3.3 V, 5.0V
- Wake up time from sleep mode is measured from the edge of the wakeup signal to the first instruction fetched in flash.
- Oscillators
  - XOSC (crystal oscillator) stopped
  - XOSC32K (32 kHz crystal oscillator) running with external 32kHz crystal
  - FDPLL using XOSC32K as reference and running at 48 MHz
- Clocks
  - FDPLL used as main clock source, except otherwise specified
  - CPU, AHB clocks undivided
  - APBA clock divided by 4
  - APBB and APBC bridges off
  - The AHB module clocks are running
    - NVMCTRL, APBA bridge
    - All other AHB clocks are stopped
  - The following peripheral clocks are running
    - PM, SYSCTRL, RTC
    - All other peripheral clocks are stopped
- I/Os are inactive with internal pull-up
- CPU is running on flash with 1 wait state
- NVMCTRL cache enabled
- BODVDD disabled

**Table 46-8. Current Consumption**

Mode	Condition	T <sub>A</sub>	V <sub>CC</sub>	Typ.	Max.	Units
ACTIVE	CPU running a While 1 algorithm	25°C	5.0V	3.4		mA
		85°C	5.0V			
	CPU running a While 1 algorithm	25°C	3.0V	3.4		mA
		85°C	3.0V			
	CPU running a While 1 algorithm, with GCLKIN as reference	25°C	5.0V			μA (with freq in MHz)
		85°C	5.0V			
	CPU running a Fibonacci algorithm	25°C	5.0V	5.0		mA
		85°C	5.0V			
	CPU running a Fibonacci algorithm	25°C	3.0V	5.0		mA
		85°C	3.0V			
	CPU running a Fibonacci algorithm, with GCLKIN as reference	25°C	5.0V			μA (with freq in MHz)
		85°C	5.0V			
	CPU running a CoreMark algorithm	25°C	5.0V	6.2		mA
		85°C	5.0V			
CPU running a CoreMark algorithm	25°C	3.0V	5.4		mA	
	85°C	3.0V				
CPU running a CoreMark algorithm, with GCLKIN as reference	25°C	5.0V			μA (with freq in MHz)	
	85°C	5.0V				
IDLE0		25°C	5.0V	1.9		mA
		85°C	5.0V			
IDLE1		25°C	5.0V	1.8		
		85°C	5.0V			
IDLE2		25°C	5.0V	1.8		
		85°C	5.0V			

## 46.7. I/O Pin Characteristics

### 46.7.1. Normal I/O Pins

Table 46-9. Normal I/O Pins Characteristics<sup>(1)</sup>

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
RPULL	Pull-up - Pull-down resistance		20	40	60	kΩ
VIL	Input low-level voltage	VDD=2.7-4.5V	-	-	0.3 * VDD	V
		VDD=4.5V-5.5V	-	-	0.3 * VDD	
VIH	Input high-level voltage	VDD=2.7-4.5V	0.7*VDD	-	-	
		VDD=4.5V-5.5V	0.7*VDD	-	-	
VOL	Output low-level voltage	VDD>2.7V, IOL max	-		0.2 * VDD	
VOH	Output high-level voltage	VDD>2.7V, IOH max	0.8*VDD		-	
IOL	Output low-level current	VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=0	-	-	2.5	mA
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=0	-	-	5	
		VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=1	-	-	5	
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=1	-	-	10	
IOH	Output high-level current	VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=0	-	-	1.5	
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=0	-	-	3	
		VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=1	-	-	3	
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=1	-	-	6	
tRISE	Rise time	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1	-	-	TBD	ns
		load = 5pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0	-	-	TBD	
tFALL	Fall time	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1	-	-	TBD	
		load = 5pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0	-	-	TBD	

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
FOmax	Maximum output frequency	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1			TBD	MHz
		load = 5pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0			TBD	
FINmax					TBD	MHz
ILEAK	Input leakage current	Pull-up resistors disabled	-1		1	µA

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

#### 46.7.2. High Sink Current I/O Pins

Table 46-10. High Sink Current I/O Pins Characteristics<sup>(1)</sup>

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
RPULL	Pull-up - Pull-down resistance		20	40	60	kΩ
VIL	Input low-level voltage	VDD=2.7-4.5V	-	-	0.3 * VDD	V
		VDD=4.5V-5.5V	-	-	0.3 * VDD	
VIH	Input high-level voltage	VDD=2.7-4.5V	0.7*VDD	-	-	
		VDD=4.5V-5.5V	0.7*VDD	-	-	
VOL	Output low-level voltage	VDD>2.7V, IOL max	-		0.2 * VDD	
VOH	Output high-level voltage	VDD>2.7V, IOH max	0.8*VDD		-	
IOL	Output low-level current	VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=0	-	-	5	mA
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=0	-	-	10	
		VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=1	-	-	10	
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=1	-	-	20	
IOH	Output high-level current	VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=0	-	-	3	
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=0	-	-	6	
		VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=1	-	-	6	
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=1	-	-	12	

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
tRISE	Rise time(1)	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1	-	-	TBD	ns
		load = 5pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0	-	-	TBD	
tFALL	Fall time(1)	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1	-	-	TBD	
		load = 5pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0	-	-	TBD	
FOmax	Maximum output frequency	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1			TBD	MHz
		load = 5pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0			TBD	
FINmax					TBD	MHz
ILEAK	Input leakage current	Pull-up resistors disabled	-1		1	μA

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

### 46.7.3. I<sup>2</sup>C Pins

Refer to I/O Multiplexing and Considerations to get the list of I<sup>2</sup>C pins.

**Table 46-11. I<sup>2</sup>C Pins Characteristics in I/O Configuration<sup>(1)</sup>**

Symbol	Parameter	Conditions	Min.	Typ.	Max,	Units
RPULL	Pull-up - Pull-down resistance					kΩ
VIL	Input low-level voltage	VDD=2.7-4.5V	-	-	0.3*VDD	V
		VDD=4.5V-5.5V	-	-	0.3*VDD	
VIH	Input high-level voltage	VDD=2.7-4.5V	0.7*VDD	-	-	
		VDD=4.5V-5.5V	0.7*VDD	-	-	
VOL	Output low-level voltage	VDD>2.7V, IOL max	-		0.2*VDD	
VOH	Output high-level voltage	VDD>2.7V, IOH max	0.8*VDD		-	

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
IOL	Output low-level current	VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=0	-	-	2.5	mA
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=0	-	-	5	
		VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=1	-	-	5	
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=1	-	-	10	
IOH	Output high-level current	VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=0	-	-	1.5	mA
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=0	-	-	3	
		VDD=2.7V-4.5V PORT.PINCFG.DRVSTR=1	-	-	3	
		VDD=4.5V-5.5V PORT.PINCFG.DRVSTR=1	-	-	6	
tRISE	Rise time	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1	-	-	TBD	ns
		load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0	-	-	TBD	
tFALL	Fall time	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1	-	-	TBD	
		load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0	-	-	TBD	
FOmax	Maximum output frequency	load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=1			TBD	MHz
		load = 20pF, VDD = 5.0V PORT.PINCFG.DRVSTR=0			TBD	
FINmax					TBD	MHz
ILEAK	Input leakage current	Pull-up resistors disabled	-1		1	μA

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

**Table 46-12. I2C Pins Characteristics in I2C Configuration<sup>(1)</sup>**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
RPULL	Pull-up - Pull-down resistance		20		60	kΩ
VIL	Input low-level voltage	VDD=2.7V-4.5V	-	-		V
		VDD=4.5V-5.5V	-	-		
VIH	Input high-level voltage	VDD=2.7V-4.5V		-	-	
		VDD=4.5V-5.5V		-	-	
VHYS	Hysteresis of Schmitt trigger inputs			-	-	
VOL	Output low-level voltage	VDD > 2.7V IOL=3mA	-	-	0.4	
CI	Capacitance for each I/O Pin	Standard & Fast Modes	-	-	400	pF
		HS Mode	100		250	
IOL	Output low-level current	VOL =0.4V Standard, Fast and HS Modes		-	3	mA
		VOL =0.4V Fast Mode +		-	20	
		VOL =0.6V		-	6	
fSCL	SCL clock frequency		-	-	3.4	MHz
RP	Value of pull-up resistor	fSCL ≤ 100kHz				Ω
		fSCL > 100kHz				

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

**Related Links**

[I/O Multiplexing and Considerations](#) on page 27

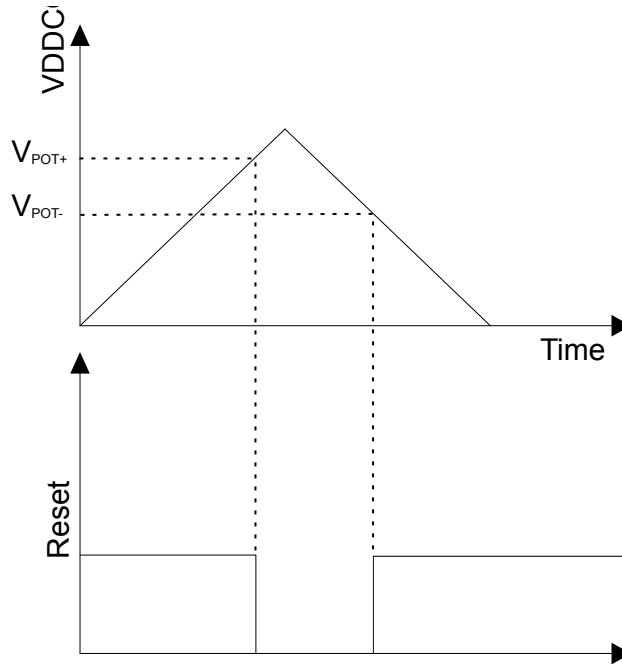
## 46.8. Analog Characteristics

### 46.8.1. Power-on Reset Characteristics

**Table 46-13. POR Characteristics**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
V <sub>POT+</sub>	Voltage threshold on V <sub>DDIN</sub> rising			2.3		V
V <sub>POT-</sub>	Voltage threshold on V <sub>DDIN</sub> falling	V <sub>DD</sub> falls faster than 1V/ms		2.27		V
		V <sub>DD</sub> falls at 1V/ms or slower		TBD		V

Figure 46-1. POR Operating Principle



## 46.8.2. Brown Out Detectors Characteristics

### 46.8.2.1. BODVDD

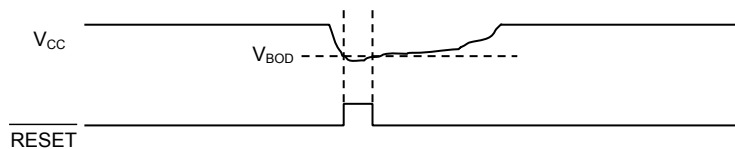
Table 46-14. BOD LEVEL Values<sup>(1)(2)</sup>

BODVDD.LEVEL	Condition	Min.	Typ.	Max.	Units
All levels	Hysteresis ON	2.45		5.5	V
4			2.7		
42			4.5		
All levels	Hysteresis ON or Hysteresis OFF	2.4		5.4	
4			2.6		
42			4.5		

1. See NVM User Row Mapping table for the BODVDD default value settings.

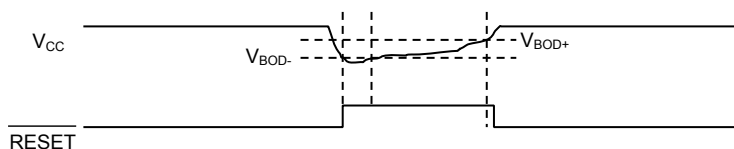
2. These values are based on simulation. These values are not covered by test limits in production or characterization.

Figure 46-2. BODVDD Hysteresis OFF





**Figure 46-3. BODVDD Hysteresis ON**



**Table 46-15. BODVDD Characteristics<sup>(1)</sup>**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
I	Step size, between adjacent values in BODVDDLEVEL			47		mV
V <sub>HYST</sub>	VBOD+ - VBOD-	Hysteresis ON	44		96	mV
t <sub>DET</sub>	Detection time	Time with V <sub>DDIN</sub> < V <sub>TH</sub> necessary to generate a reset signal		TBD		μs
I <sub>BOD33</sub>	Current consumption	Continuous mode		TBD		μA
		Sampling mode		TBD		μA
t <sub>STARTUP</sub>	Startup time				3.1	μs

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

**Related Links**

[NVM User Row Mapping](#) on page 36

**46.8.3. LDO Regulator Characteristics**

**Table 46-16. LDO Regulator Characteristics**

Symbol	Parameter	Min.	Typ.	Max.	Units
VDDIN	Input voltage range	2.7		5.5	V
VDDCORE	DC calibrated output voltage		1.23		V

**Table 46-17. Decoupling Requirements**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
C <sub>in</sub>	Input regulator capacitor	tantalum	-	1	-	μF
		ceramic dielectric	-	100	-	nF
C <sub>out</sub>	Output regulator capacitor	tantalum	-	1	-	μF
		ceramic dielectric	-	100	-	nF

#### 46.8.4. Analog-to-Digital (ADC) Characteristics

Table 46-18. Operation Conditions<sup>(1)</sup>

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
Res	Resolution			-	12	bits
	Conversion speed		10	-	1000	ksp/s
fs	Sampling clock		10	-	1000	kHz
clk	ADC Clock frequency	OFFCOMP=1 or R2R=1		fs*16		Hz
		OFFCOMP=0		fs*13		
Ts	Sampling time	OFFCOMP=1 or R2R=1	250	-	25000	ns
		OFFCOMP=0	76	-	7692	
	Conversion range	Diff mode	-VREF	-	+VREF	V
		Single-ended mode	0	-	VREF	
Vref	Reference input	REFCOMP=1	2	-	VDDANA-0.6	V
		REFCOMP=0	VDDANA	-	VDDANA	
Vin	Input channel range	-	0	-	VDDANA	V
Vcmin	Input common mode voltage	R2R=1	0.2	-	VREF-0.2	V
		R2R=0	VREF/2-0.2	-	VREF/2+0.2	
C <sub>SAMPLE</sub>	Input sampling capacitance		-	1.6	-	pF
R <sub>SAMPLE</sub>	Input channel source resistance	For a sampling rate at 1MSPS		1000		Ω
R <sub>ref</sub>	Reference input source resistance		0		1000	KΩ

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
IDD <sub>VDDANA</sub>	Differential mode	fs = 1 Msps / Reference buffer disabled / BIASREFBUF = '111', BIASREFCOMP = '111' Vcc=Vref=1.6V  Vcc=Vref= 3.6V		300		μA
		fs = 1 Msps / Reference buffer enabled / BIASREFBUF = '111', BIASREFCOMP = '111' Vcc=3.0V Vref=2,0V  Vcc=1.6V Vref=1.0V Vcc=3.6V Vref=3.0V		500		
		fs = 10 ksps / Reference buffer disabled / BIASREFBUF = '111', BIASREFCOMP = '111' Vcc=Vref=1.6V  Vcc=Vref= 3.6V		250		μA
		fs = 10 ksps / Reference buffer enabled / BIASREFBUF = '111', BIASREFCOMP = '111' Vcc=3,0V Vref=2,0V  Vcc=1.6V Vref=1.0V Vcc=3.6V Vref=3.0V		450		

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
IDD <sub>VDDANA</sub>	Single Ended mode	fs = 1 Msps / Reference buffer disabled Vcc=Vref=1.6V Vcc=Vref= 3.6V		300		μA
		fs = 1 Msps / Reference buffer enabled / BIASREFBUF = '111', BIASREFCOMP = '111' Vcc=3,0V Vref=2,0V Vcc=1.6V Vref=1.0V Vcc=3.6V Vref=3.0V		500		
		fs = 10 ksps / Reference buffer disabled / BIASREFBUF = '111', BIASREFCOMP = '111' Vcc=Vref=1.6V Vcc=Vref= 3.6V		250		μA
		fs = 10 ksps / Reference buffer enabled / BIASREFBUF = '111', BIASREFCOMP = '111' Vcc=3,0V Vref=2,0V Vcc=1.6V Vref=1.0V Vcc=3.6V Vref=3.0V		450		

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

**Table 46-19. Differential Mode<sup>(1)</sup>**

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
ENOB	Effective Number of bits	With gain compensation Vcc=3.0V / Vref =2.0V Vcc=1.6V/3.6V Vref=1.0V Vcc=Vref=1.6V Vcc=Vref=3.6V		9.5		bits
TUE	Total Unadjusted Error	Vcc=3.0V Vref=2.0V		-10	10	LSB
INL	Integral Non Linearity	Vcc=3.0V Vref=2.0V		-3	3	LSB
DNL	Differential Non Linearity	Vcc=3.0V Vref=2.0V		-1	1	LSB

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
	Gain Error	External Reference voltage			0.25	%
		Reference bandgap voltage			2	
		VDDANA2			2	
		VDDANA/1.6			2	
	Offset Error	External Reference voltage	-15	2	15	mV
		Reference bandgap voltage	-15	2	15	
		VDDANA/2	-15	2	15	
		VDDANA/1.6	-15	2	15	
SFDR	Spurious Free Dynamic Range	Fs = 1MHz / Fin = 13 kHz / Full range Input signal Vcc=3.0V Vref=2.0V		75		dB
SINAD	Signal to Noise and Distortion ratio			58		dB
SNR	Signal to Noise ratio			59		dB
THD				70		dB
	Noise RMS	External Reference voltage		2.0		mV

These values are based on simulation. These values are not covered by test limits in production or characterization.

**Table 46-20. Single-Ended Mode<sup>(1)</sup>**

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
ENOB	Effective Number of bits	With gain compensation Vcc=3.0V / Vref =2.0V  Vcc=1.6V/3.6V Vref=1.0V  Vcc=Vref=1.6/3.6V	8.6			bits
TUE	Total Unadjusted Error	Vcc=3.0V Vref=2.0V	-20		20	LSB
INL	Integral Non Linearity	Vcc=3.0V Vref=2.0V	-6		6	LSB
DNL	Differential Non Linearity	Vcc=3.0V Vref=2.0V	-1		2	LSB
	Gain Error	External Reference voltage	-0.25		0.25	%
		Reference bandgap voltage	-2		2	
		VDDANA/2	-2		2	
		VDDANA/1.6	-2		2	

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
	Offset Error	External Reference voltage	-15	2	15	mV
		Reference bandgap voltage	-15	2	15	
		VDDANA/2	-15	2	15	
		VDDANA/1.6	-15	2	15	
SFDR	Spurious Free Dynamic Range	Fs = 1MHz / Fin = 13 kHz / Full range Input signal Vcc=3.0V Vref=2.0V		75		dB
SINAD	Signal to Noise and Distortion ratio			52		dB
SNR	Signal to Noise ratio			53		dB
THD				70		dB
	Noise RMS	External Reference voltage		2		mV

These values are based on simulation. These values are not covered by test limits in production or characterization.

#### 46.8.5. Sigma-Delta Analog-to-Digital (SDADC) Characteristics

Table 46-21. Operating Conditions

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
Res	Resolution	Differential mode		16		bits
		Single-Ended mode		15		
CLK_SDADC	Sampling Clock Speed				6	MHz
CLK_SDADC_FS	Conversion rate				1.5	MHz
fs	Output Data Rate	Free running mode - SKPCNT = 0x0		CLK_SDADC_FS / OSR		MHz
		Single conversion mode - SKPCNT = 0x2		(CLK_SDADC_FS / OSR)*3		
OSR	Oversampling ratio	Diff mode	64	256	1024	Cycles

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
	Input Conversion range	Differential mode Gaincorr = 0x1	-VREF		VREF	V
		Single-Ended mode Gaincorr = 0x1	0		VREF	
Vcom	Common mode voltage	Differential	0		AVDD	V

**Table 46-22. SDADC DC Performance**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
INL	Integral Non Linearity	CLK_SDADC = 6MHz; VREF = 1.2V		5		LSB
		CLK_SDADC = 6MHz; INT VREF = 5.0V		10		
DNL	Differential Non Linearity	CLK_SDADC = 6MHz; VREF = 1.2V		3		LSB
		CLK_SDADC = 6MHz; INT VREF = 5.0V		10		
Offset Errors	Offset Errors	CLK_SDADC = 6MHz; VREF = 1.2V		-3		mV
		CLK_SDADC = 6MHz; INT VREF = 5.0V		-10		
Offset Drift	Offset Drift	CLK_SDADC = 6MHz; VREF = 1.2V	-30		+30	$\mu\text{V}/^\circ\text{C}$
Gain Errors	Gain Errors	CLK_SDADC = 6MHz; VREF = 1.2V		-1.0		%
		CLK_SDADC = 6MHz; INT VREF = 5.0V		-0.1		
Gain Drift	Gain Drift	CLK_SDADC = 6MHz; VREF = 1.2V		-5		$\text{ppm}/^\circ\text{C}$
Input noise rms	Input noise rms	OSR = 256		30		$\mu\text{Vrms}$
Idd	Power consumption	Int Ext – CTLSADC=0x0 SCLK_SDADC = 6 MHz		200		$\mu\text{A}$

**Table 46-23. SDADC AC Performance**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
SNR	Signal to Noise Ratio	Ext ref = 1.2V		74.9		dB
		Int Ref = 5.0V		79.5		
SINAD	Signal to Noise + Distortion Ratio	Ext ref = 1.2V		72.9		dB
		Int Ref = 5.0V		79.4		
THD	Total Harmonic Distortion	Ext ref = 1.2V		-77.2		dB
		Int Ref = 5.0V		-95.5		

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
ENOB	Effective Number Of Bits	Ext ref = 1.2V		12.6		dB
		Int Ref = 5.0V		12.9		
DR	Dynamic Range	Ext ref = 1.2V		86		dB
		Int Ref = 5.0V		92		

#### 46.8.6. Digital to Analog Converter (DAC) Characteristics

Table 46-24. Operating Conditions<sup>(1)</sup>

Symbol	Parameters	Conditions	Min.	Typ.	Max.	Units
VDDANA	Analog supply voltage		2.7	-	5.5	V
AVREF	External reference voltage		1	-	VDDANA - 0.6	V
	Internal reference voltage 1			1.024 2.048 4.096 <sup>(3)</sup>		V
	Internal reference voltage 2		-	VDDANA	-	V
	Linear output voltage range		0.05	-	VDDANA - 0.5	V
	Minimum resistive load		5	-	-	K $\Omega$
	Maximum capacitance load		-	-	100	pF
IDD	DC supply current <sup>(2)</sup>	Output buffer on		260		$\mu$ A
		Output buffer off		80		

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.
3. For VDDANA > 4.5V

Table 46-25. Clock and Timing<sup>(1)</sup>

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units	
	Conversion rate	Cload=100pF Rload > 5kW	Normal mode			350	ksps
			For DDATA=+/-1			1000	
	Startup time				3	$\mu$ s	

1. These values are based on simulation. These values are not covered by test limits in production or characterization.



**Table 46-26. Accuracy Characteristics<sup>(1)</sup>**

Symbol	Parameter	Conditions		Min.	Typ.	Max.	Units
RES	Input resolution					10	Bits
INL	Integral non-linearity	VREF= Ext 1.0V	VDD = 2.7V	1			LSB
			VDD = 5.5V	0.45			
		VREF = VDDANA	VDD = 2.7V	0.9			
			VDD = 5.5V	0.5			
		VREF= INT1V	VDD = 2.7V	TBD			
			VDD = 5.5V	TBD			
DNL	Differential non-linearity	VREF= Ext 1.0V	VDD = 2.7V	+/-0.3			LSB
			VDD = 5.5V	+/-0.4			
		VREF= VDDANA	VDD = 2.7V	+/-0.25			
			VDD = 5.5V	+/- 0.2			
		VREF= INT1V	VDD = 2.7V	TBD			
			VDD = 5.5V	TBD			
	Gain error	Ext. VREF		±5		mV	
	Offset error	Ext. VREF		±3		mV	

1. These values are based on simulation using a conversion rate of 350ksp/s

#### 46.8.7. Analog Comparator Characteristics

**Table 46-27. Analog Comparator Electrical and Timing Characteristics**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
	Positive input voltage range		0		VDDANA	V
	Negative input voltage range		0		VDDANA	
	Offset	Hysteresis = 0, Fast mode		0		mV
		Hysteresis = 0, Low power mode		0		mV
	Hysteresis	Hysteresis = 1, Fast mode		50		mV
		Hysteresis = 1, Low power mode		50		mV
	Propagation delay	Changes for VACM=VDDANA/2 100mV overdrive, Fast mode		35		ns
		Changes for VACM=VDDANA/2 100mV overdrive, Low power mode		125		ns

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
t <sub>STARTUP</sub>	Startup time	Enable to ready delay Fast mode		2		μs
		Enable to ready delay Low power mode		6		μs

## 46.8.8. Bandgap Reference Characteristics

Table 46-28. Internal Voltage Reference Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
ADC/DAC Ref	ADC0, ADC1, AC, DAC, SDADC Internal reference	After calibration at T= 25°C, over [-40, +85]°C. VREF.SEL = 0		1.024		V
		After calibration at T= 25°C, over [-40, +85]°C. VREF.SEL = 2		2.048		
		After calibration at T= 25°C, over [-40, +85]°C. VREF.SEL = 3		4.096		
		Over voltage at 25°C		TBD		

## 46.9. NVM Characteristics

Table 46-29. Maximum Operation Frequency<sup>(1)</sup>

VDD Range	NVM Wait States	Maximum Operating Frequency	Units
2.7V to 5.5V	0	16	MHz
	1	32	
	2	48	
	3	48	

1. Based on simulation.

Note that on this flash technology, a max number of 8 consecutive write is allowed per row. Once this number is reached, a row erase is mandatory.

## 46.10. Oscillator Characteristics

### 46.10.1. Crystal Oscillator (XOSC) Characteristics

#### 46.10.1.1. Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN.

**Table 46-30. Digital Clock Characteristics**

Symbol	Parameter	Condition	Min	Typ	Max	Units
f <sub>CPXIN</sub>	XIN clock frequency	Digital mode	-	-	48	MHz
DC <sub>XIN</sub>	XIN clock duty cycle	Digital mode	40	50	60	%

**46.10.1.2. Crystal Oscillator Characteristics**

The following table describes the characteristics for the oscillator when a crystal is connected between XIN and XOUT as shown in [Figure 46-4 Oscillator Connection](#) on page 1128. 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. The capacitance of the external capacitors (C<sub>LEXT</sub>) can then be computed as follows:

$$C_{LEXT} = 2(C_L + C_{STRAY} - C_{SHUNT})$$

where C<sub>STRAY</sub> is the capacitance of the pins and PCB, C<sub>SHUNT</sub> is the shunt capacitance of the crystal.

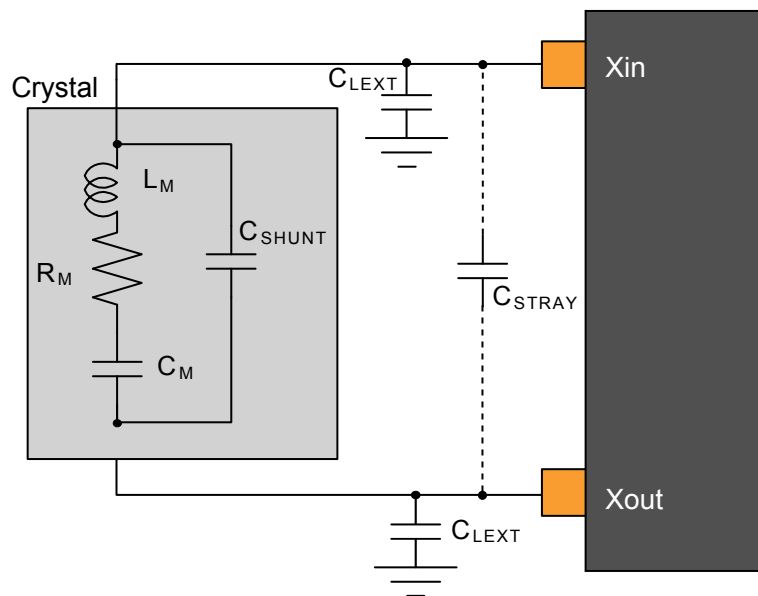
**Table 46-31. Crystal Oscillator Characteristics**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
Fout	Crystal oscillator frequency		0.4	-	32	MHz
ESR	Crystal Equivalent Series Resistance - SF = 3	f = 0.455 MHz, CL = 100pF XOSC.GAIN = 0	-	-	663 <sup>(1)</sup>	Ω
		F = 2MHz - CL=20 pF XOSC,GAIN=0	-	-	574 <sup>(1)</sup>	
		F = 4MHz - CL=20 pF XOSC,GAIN=1	-	-	327 <sup>(1)</sup>	
		F = 8MHz - CL=20 pF XOSC,GAIN=2	-	-	174 <sup>(1)</sup>	
		F = 16MHz - CL=20 pF XOSC,GAIN=3	-	-	100 <sup>(1)</sup>	
		F = 32MHz - CL=18 pF XOSC,GAIN=4	-	-	99 <sup>(1)</sup>	
Cxin	Parasitic load capacitor		-	7	-	pF
Cxout			-	7	-	
Tstart	Startup time	F = 2MHz - CL=20 pF XOSC,GAIN=0, ESR=600 Ohms		14K		Cycles
		F = 4MHz - CL=20 pF XOSC,GAIN=1, ESR=100 Ohms		6800		
		F = 8MHz - CL=20 pF XOSC,GAIN=2, ESR=35 Ohms		5550		
		F = 16MHz - CL=20 pF XOSC,GAIN=3, ESR=25 Ohms		7650		
		F = 32MHz - CL=18 pF XOSC,GAIN=4, ESR=40 Ohms		5.3K		

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
Ixosc	Current consumption	F = 2MHz - CL=20 pF XOSC,GAIN=0, AGC off	-	97		$\mu\text{A}$
		F = 2MHz - CL=20 pF XOSC,GAIN=0, AGC on	-	23		
		F = 4MHz - CL=20 pF XOSC,GAIN=1, AGC off	-	153		
		F = 4MHz - CL=20 pF XOSC,GAIN=1, AGC on	-	45		
		F = 8MHz - CL=20 pF XOSC,GAIN=2, AGC off	-	272		
		F = 8MHz - CL=20 pF XOSC,GAIN=2, AGC on	-	99		
		F = 16MHz - CL=20 pF XOSC,GAIN=3, AGC off	-	508		
		F = 16MHz - CL=20 pF XOSC,GAIN=3, AGC on	-	183		
		F = 32MHz - CL=18 pF XOSC,GAIN=4, AGC off	-	1069		
		F = 32MHz - CL=18 pF XOSC,GAIN=4, AGC on	-	716		

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

**Figure 46-4. Oscillator Connection**



## 46.10.2. External 32 kHz Crystal Oscillator (XOSC32K) Characteristics

### 46.10.2.1. Digital Clock Characteristics

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN32 pin.

**Table 46-32. Digital Clock Characteristics**

Symbol	Parameter	Condition	Min	Typ	Max	Units
$f_{CPXIN32}$	XIN32 clock frequency	Digital mode		32.768		MHz
$DC_{XIN32}$	XIN32 clock duty cycle	Digital mode		50		%

### 46.10.2.2. Crystal Oscillator Characteristics

Figure 46-4 Oscillator Connection on page 1128 and the equation above also applies to the 32 kHz oscillator connection. The user must choose a crystal oscillator where the crystal load capacitance  $C_L$  is within the range given in the table. The exact value of  $C_L$  can then be found in the crystal datasheet.

**Table 46-33. 32kHz Crystal Oscillator Characteristics**

Symbol	Parameter	Condition	Min	Typ	Max	Units
$f_{OUT}$	Crystal oscillator frequency			32768		Hz
$t_{STARTUP}$	Startup time	$R_m = 100\text{ k}\Omega$ , $C_L = 12.5\text{ pF}$		28K		cycles
$C_L$	Crystal load capacitance				12.5	pF
$C_{SHUNT}$	Crystal shunt capacitance			0.1		
$C_{XIN32}$	Parasitic capacitor load	TQFP64:48 package		3.1		
$C_{XOUT32}$	Parasitic capacitor load			3.3		
$I_{OSC32K}$	Current consumption			1.5		nA
ESR	Crystal equivalent series resistance $f=32.768\text{kHz}$ Safety Factor = 3	$C_L=12.5\text{pF}$		290		k $\Omega$

### 46.10.3. Digital Phase Locked Loop (DPLL) Characteristics

**Table 46-34. Fractional Digital Phase Locked Loop Characteristics**

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
$f_{IN}$	Input frequency		32		2000	KHz
$f_{OUT}$	Output frequency		48		96	MHz
IFDPLL96M	Current consumption	$f_{IN}= 32\text{ kHz}$ , $f_{OUT}= 48\text{ MHz}$		250 <sup>(1)</sup>		$\mu\text{A}$
		$f_{IN}= 32\text{ kHz}$ , $f_{OUT}= 96\text{ MHz}$		500 <sup>(1)</sup>		
$J_p$	Period jitter	$f_{IN}= 32\text{ kHz}$ , $f_{OUT}= 48\text{ MHz}$				%
		$f_{IN}= 32\text{ kHz}$ , $f_{OUT}= 96\text{ MHz}$				
		$f_{IN}= 2\text{ MHz}$ , $f_{OUT}= 48\text{ MHz}$				
		$f_{IN}= 2\text{ MHz}$ , $f_{OUT}= 96\text{ MHz}$				

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
t <sub>LOCK</sub>	Lock Time	After startup, time to get lock signal. f <sub>IN</sub> = 32 kHz, f <sub>OUT</sub> = 96 MHz				ms
		f <sub>IN</sub> = 2 MHz, f <sub>OUT</sub> = 96 MHz				μs
Duty	Duty cycle			50		%

1. These values are based on simulation. These values are not covered by test limits in production or characterization.

#### 46.10.4. 32.768kHz Internal oscillator (OSC32K) Characteristics

Table 46-35. 32 kHz RC Oscillator Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
f <sub>OUT</sub>	Output frequency	T =25°C, VDDANA = 5.0V		32.768		kHz
		T =25°C, over [2.7, 5.5]V		32.768		
		over [-40, 85]°C, over [2.7, 5.5]V		32.768		
I <sub>RC32K</sub>	Current consumption			0.8		μA
t <sub>STARTUP</sub>	Startup time			5		cycle
Duty	Duty Cycle			50		%

#### 46.10.5. Ultra Low Power Internal 32kHz RC Oscillator (OSCULP32K) Characteristics

Table 46-36. Ultra Low Power Internal 32 kHz RC Oscillator Electrical Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
f <sub>OUT</sub>	Output frequency	T =25°C, VDDANA = 5.0V		32.768		kHz
		T =25°C, over [2.7, 5.5]V		32.768		
		Over [-40, 85]°C, over [2.7, 5.5]V		32.768		
Duty	Duty Cycle			50		%

#### 46.10.6. 48MHz RC Oscillator (OSC48M) Characteristics

Table 46-37. Internal 48MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
Fout	Output frequency			48		MHz
TempCo	Freq vs temperature drift					%
SupplyCo	Freq vs supply drift					
Iosc48m	Current consumption			200		μA
Twup	Wake up time - 1st clock edge after enable				7	μs
Tstart	Startup time	95% of Fout			15	μs
Duty	Duty Cycle			50		%

## 46.11. Timing Characteristics

### 46.11.1. External Reset

Table 46-38. External Reset Characteristics

Symbol	Parameter	Min.	Typ.	Max.	Units
$t_{EXT}$	Minimum reset pulse width	10			ns

## 47. Packaging Information

### 47.1. Thermal Considerations

#### 47.1.1. Thermal Resistance Data

The [Table 47-1 Thermal Resistance Data](#) on page 1132 summarizes the thermal resistance data depending on the package.

**Table 47-1. Thermal Resistance Data**

Package Type	$\theta_{JA}$	$\theta_{JC}$
32-pin TQFP	68°C/W	25.8°C/W
48-pin TQFP	78.8°C/W	12.3°C/W
64-pin TQFP	66.7°C/W	11.9°C/W
32-pin QFN	37.2°C/W	3.1°C/W
48-pin QFN	33°C/W	11.4°C/W
64-pin QFN	33.5°C/W	11.2°C/W

#### Related Links

[Junction Temperature](#) on page 1132

#### 47.1.2. Junction Temperature

The average chip-junction temperature,  $T_J$ , in °C can be obtained from the following:

1.  $T_J = T_A + (P_D \times \theta_{JA})$
2.  $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

where:

- $\theta_{JA}$  = Package thermal resistance, Junction-to-ambient (°C/W), see Thermal Resistance Data
- $\theta_{JC}$  = Package thermal resistance, Junction-to-case thermal resistance (°C/W), see Thermal Resistance Data
- $\theta_{HEATSINK}$  = Thermal resistance (°C/W) specification of the external cooling device
- $P_D$  = Device power consumption (W)
- $T_A$  = Ambient temperature (°C)

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature  $T_J$  in °C.

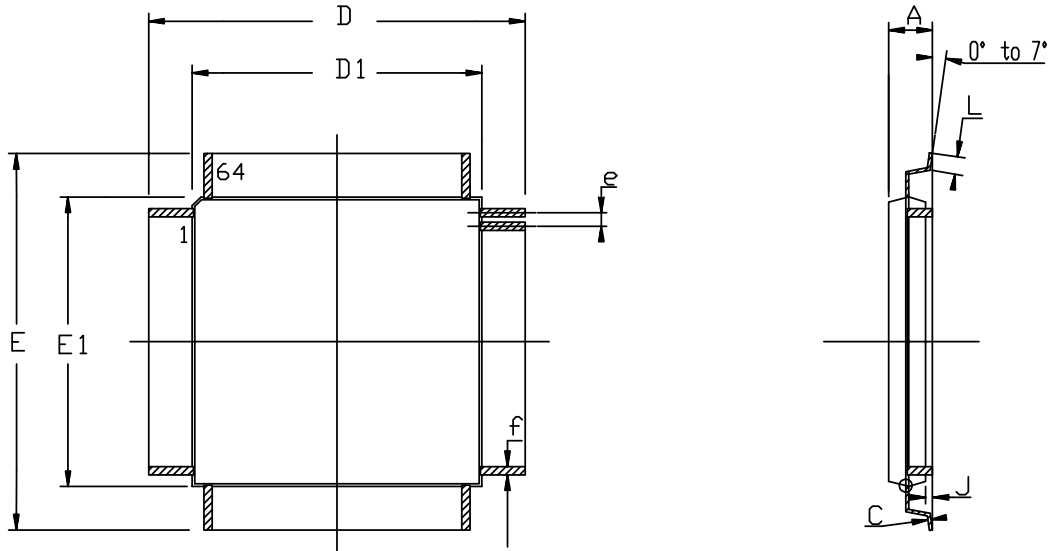
#### Related Links

[Thermal Resistance Data](#) on page 1132



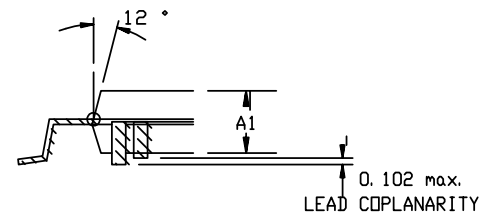
## 47.2. Package Drawings

### 47.2.1. 64 pin TQFP



COMMON DIMENSIONS IN MM

SYMBOL	Min	Max	NOTES
A	----	1.20	
A1	0.95	1.05	
C	0.09	0.20	
D	12.00 BSC		
D1	10.00 BSC		
E	12.00 BSC		
E1	10.00 BSC		
J	0.05	0.15	
L	0.45	0.75	
e	0.50 BSC		
f	0.17	0.27	



**Table 47-2. Device and Package Maximum Weight**

300	mg
-----	----

**Table 47-3. Package Characteristics**

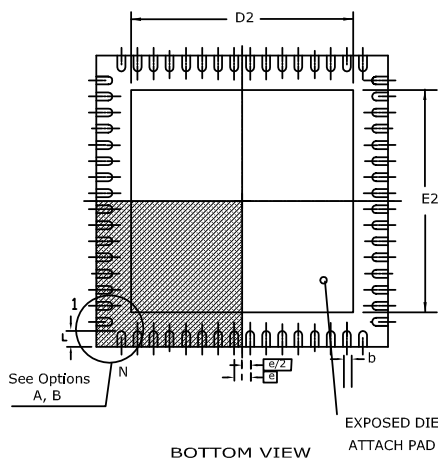
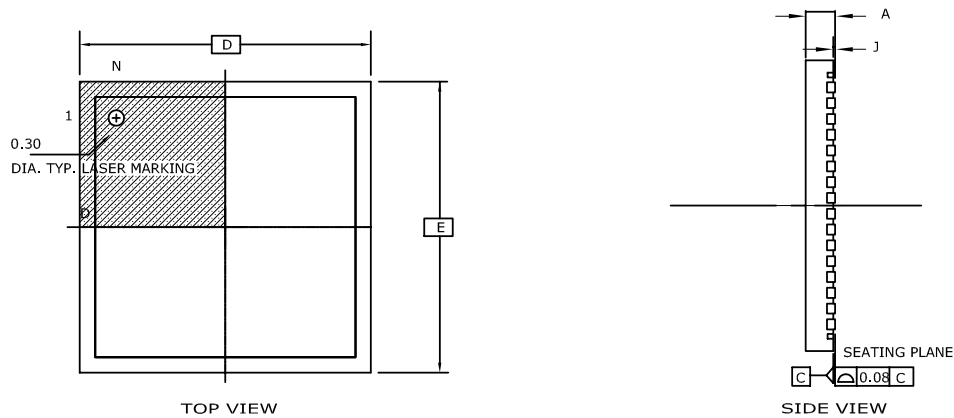
Moisture Sensitivity Level	MSL3
----------------------------	------

**Table 47-4. Package Reference**

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

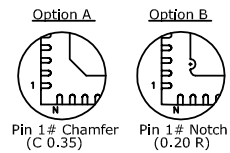
## 47.2.2. 64 pin QFN

DRAWINGS NOT SCALED



COMMON DIMENSIONS  
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	-----	1.00	
D/E	9.00 BSC			
D2/E2	4.60	4.70	4.80	
J	0.00	-----	0.05	
b	0.15	0.20	0.25	
e	0.50 BSC			
L	0.30	0.40	0.55	
N	64			



- Notes : 1. This drawing is for general information only. Refer to JEDEC Drawing MO-220  
 2. Dimension b applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip.  
 If the terminal has the optical radius on the other end of the terminal, the dimension should not be measured in that radius area.

**Note:** The exposed die attach pad is not connected electrically inside the device.

**Table 47-5. Device and Package Maximum Weight**

200	mg
-----	----

**Table 47-6. Package Characteristics**

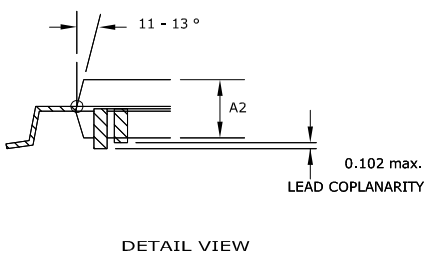
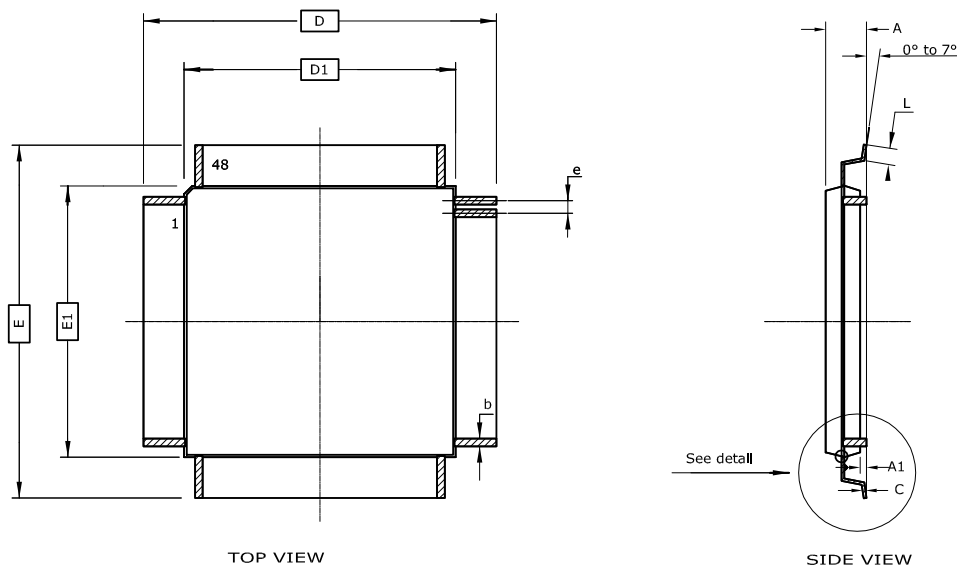
Moisture Sensitivity Level	MSL3
----------------------------	------

**Table 47-7. Package Reference**

JEDEC Drawing Reference	MO-220
JESD97 Classification	E3

### 47.2.3. 48 pin TQFP

DRAWINGS NOT SCALED



COMMON DIMENSIONS  
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	-----	-----	1.20	
A1	0.05	-----	0.15	
A2	0.95	-----	1.05	
C	0.09	-----	0.20	
D/E	9.00 BSC			
D1/E1	7.00 BSC			
L	0.45	-----	0.75	
b	0.17	-----	0.27	
e	0.50 BSC			

- Notes :
1. This drawing is for general information only. Refer to JEDEC Drawing MS-026, Variation ABC.
  2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
  3. Lead coplanarity is 0.10mm maximum.

**Table 47-8. Device and Package Maximum Weight**

140	mg
-----	----

**Table 47-9. Package Characteristics**

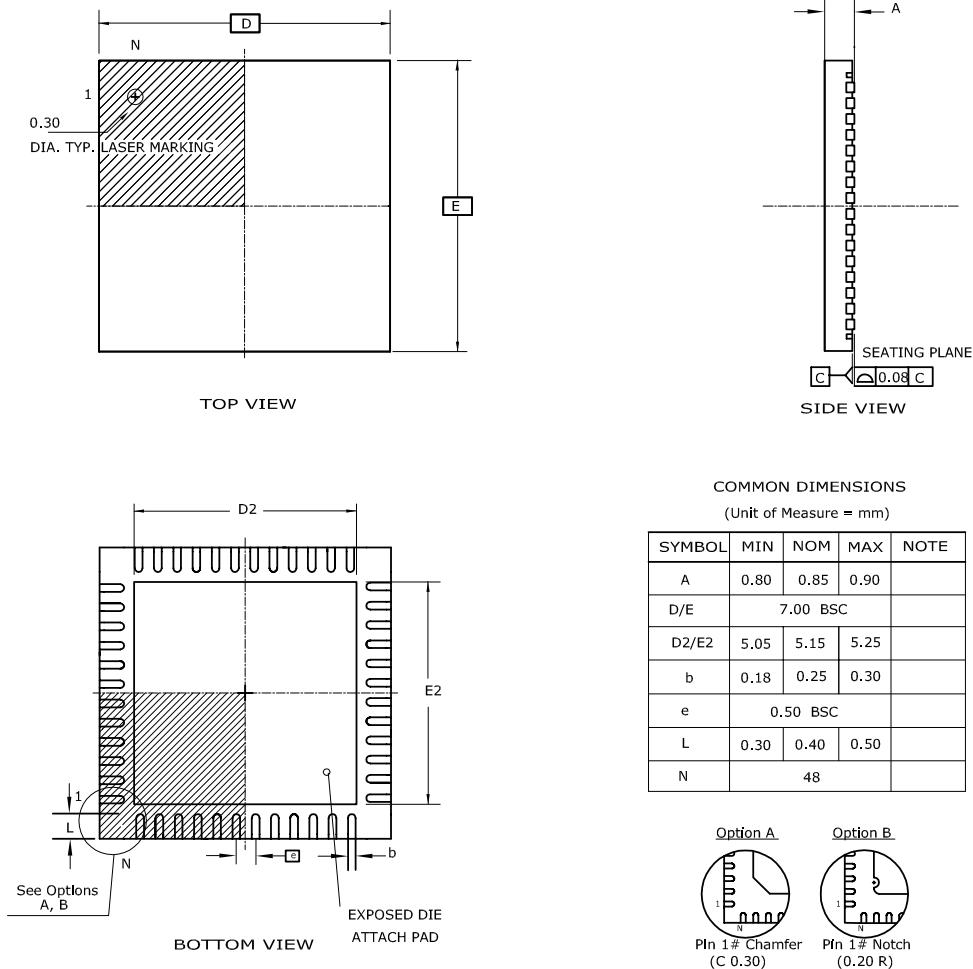
Moisture Sensitivity Level	MSL3
----------------------------	------

**Table 47-10. Package Reference**

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

## 47.2.4. 48 pin QFN

DRAWINGS NOT SCALED



Notes : 1. This drawing is for general information only. Refer to JEDEC Drawing MO-220, Variation VKKD-4, for proper dimensions, tolerances, datums, etc.  
2. Dimension b applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip.  
If the terminal has the optical radius on the other end of the terminal, the dimension should not be measured in that radius area.

**Table 47-11. Device and Package Maximum Weight**

140	mg
-----	----

**Table 47-12. Package Characteristics**

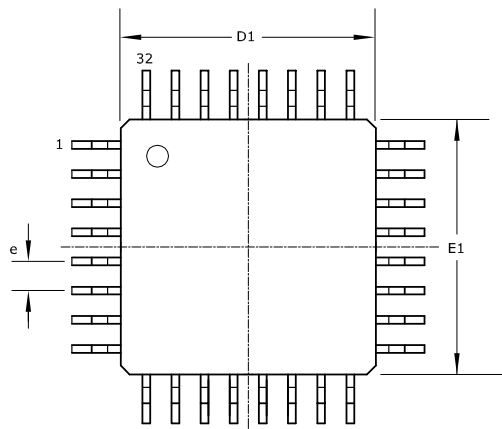
Moisture Sensitivity Level	MSL3
----------------------------	------

**Table 47-13. Package Reference**

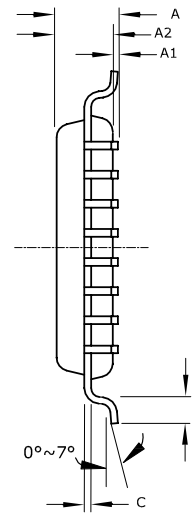
JEDEC Drawing Reference	MO-220
JESD97 Classification	E3

## 47.2.5. 32 pin TQFP

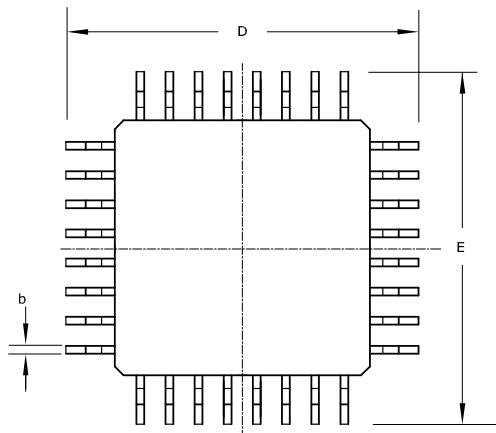
DRAWINGS NOT SCALED



TOP VIEW



SIDE VIEW



BOTTOM VIEW

COMMON DIMENSIONS  
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	----	----	1.20	
A1	0.05	----	0.15	
A2	0.95	1.00	1.05	
D/E	8.75	9.00	9.25	
D1/E1	6.90	7.00	7.10	2
C	0.09	----	0.20	
L	0.45	----	0.75	
b	0.30	----	0.45	
e	0.80 TYP			
n	32			

- Notes :
1. This drawing is for general information only. Refer to JEDEC Drawing MS-026, Variation ABA.
  2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
  3. Lead coplanarity is 0.10mm maximum.

**Table 47-14. Device and Package Maximum Weight**

100	mg
-----	----

**Table 47-15. Package Characteristics**

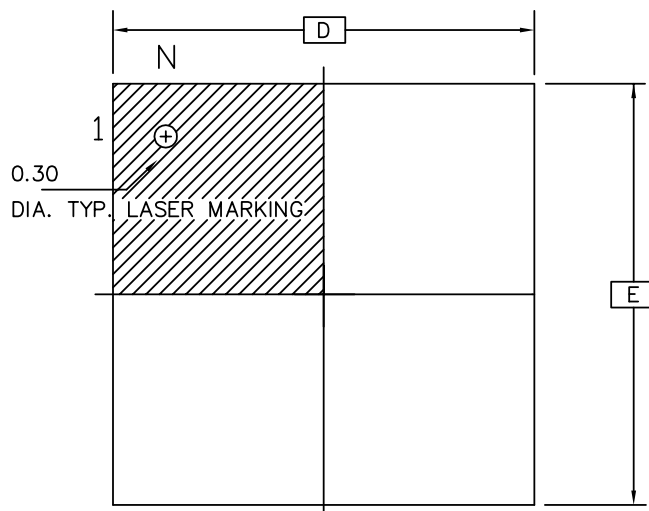
Moisture Sensitivity Level	MSL3
----------------------------	------

**Table 47-16. Package Reference**

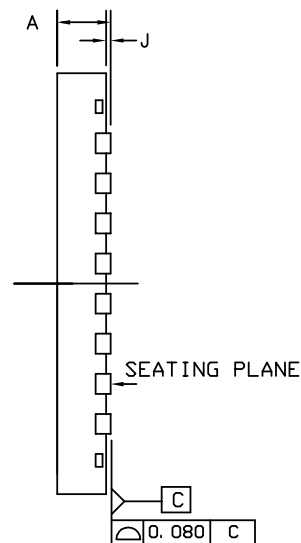
JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

## 47.2.6. 32 pin QFN

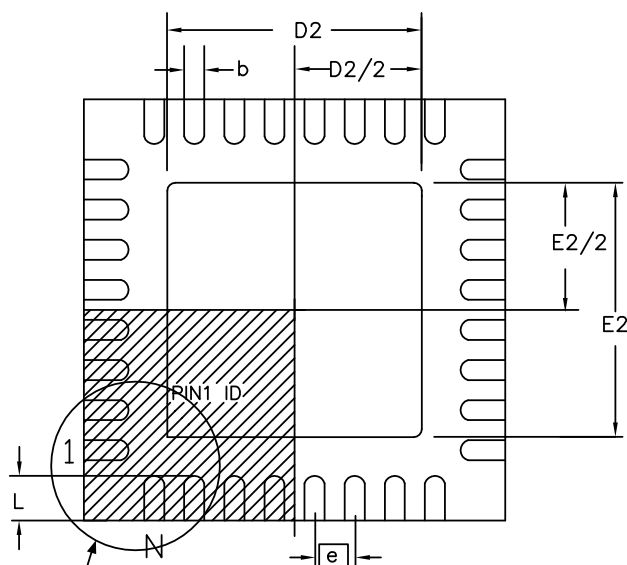
DRAWINGS NOT SCALED



TOP VIEW



SIDE VIEW

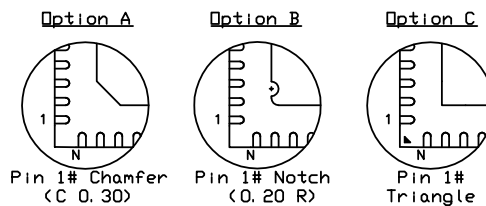


BOTTOM VIEW

See Options A, B, C

COMMON DIMENSIONS IN MM

SYMBOL	MIN.	NOM.	MAX.	NOTES
A	0.80	----	1.00	
J	0.00	----	0.05	
D/E	5.00		BSC	
D2/E2	3.50	3.60	3.70	
N	32			
e	0.50		BSC	
L	0.30	0.40	0.50	
b	0.18	0.25	0.30	



**Note:** The exposed die attach pad is connected inside the device to GND and GNDANA.

Table 47-17. Device and Package Maximum Weight

90	mg
----	----

Table 47-18. Package Characteristics

Moisture Sensitivity Level	MSL3
----------------------------	------

**Table 47-19. Package Reference**

JEDEC Drawing Reference	MO-220
JESD97 Classification	E3

### 47.3. Soldering Profile

The following table gives the recommended soldering profile from J-STD-20.

**Table 47-20.**

Profile Feature	Green Package
Average Ramp-up Rate (217°C to peak)	3°C/s max.
Preheat Temperature 175°C ±25°C	150-200°C
Time Maintained Above 217°C	60-150s
Time within 5°C of Actual Peak Temperature	30s
Peak Temperature Range	260°C
Ramp-down Rate	6°C/s max.
Time 25°C to Peak Temperature	8 minutes max.

A maximum of three reflow passes is allowed per component.

## 48. Schematic Checklist

### 48.1. Introduction

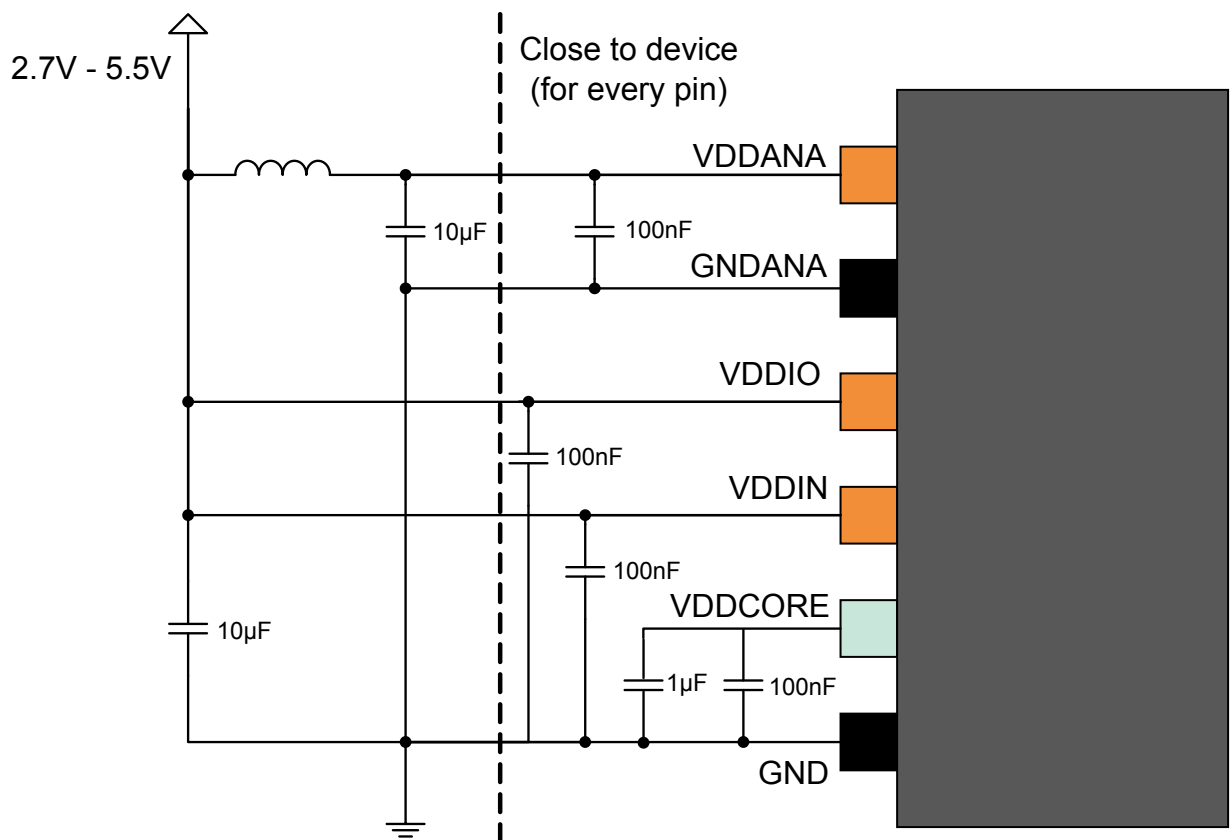
This chapter describes a common checklist which should be used when starting and reviewing the schematics for a SAM C21 design. This chapter illustrates recommended power supply connections, how to connect external analog references, programmer, debugger, oscillator and crystal.

### 48.2. Power Supply

The SAM C21 supports a single power supply or dual power supplies from 2.7 to 5.5V.

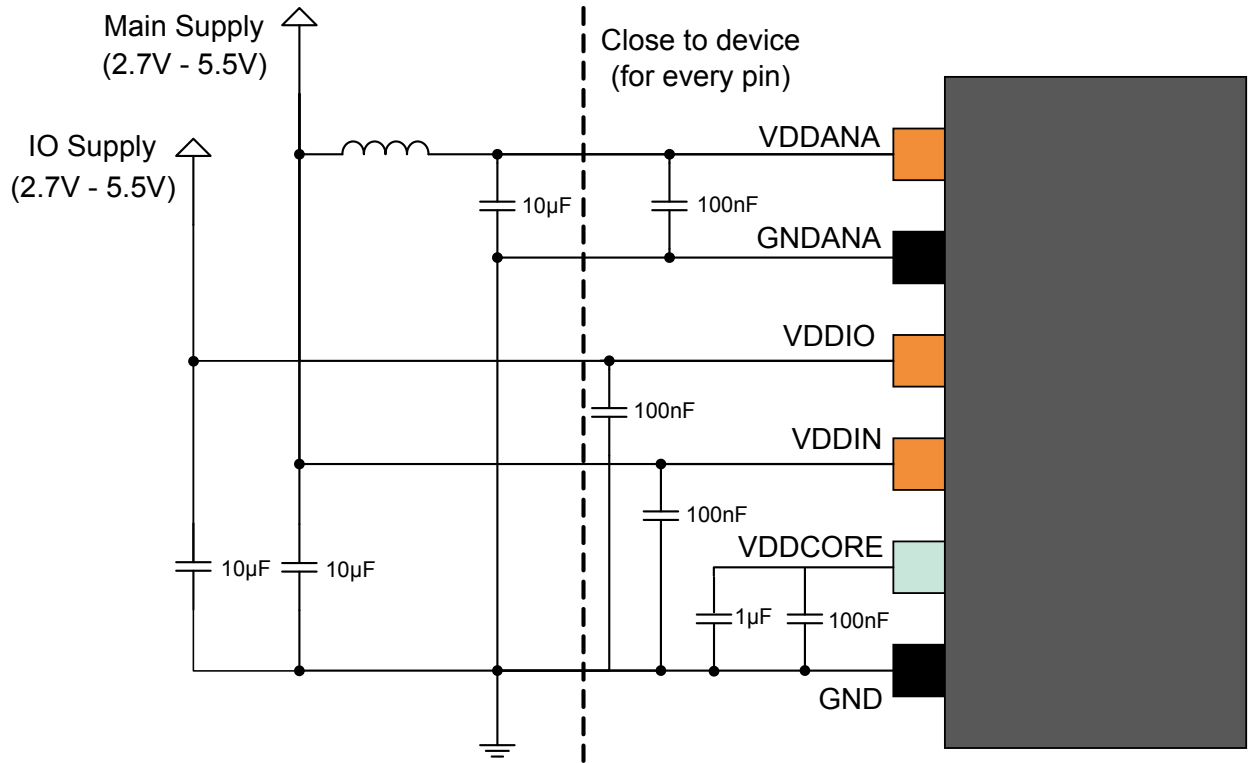
#### 48.2.1. Power Supply Connections

Figure 48-1. Single Power Supply Schematic





**Figure 48-2. Dual Power Supply Schematic**



**Table 48-1. Power Supply Connections,  $V_{DDCORE}$  From Internal Regulator**

Signal Name	Recommended Pin Connection	Description
$V_{DDIO}$	2.7V to 5.5V Decoupling/filtering capacitors 100nF <sup>(1)(2)</sup> and 10µF <sup>(1)</sup> Decoupling/filtering inductor 10µH <sup>(1)(3)</sup>	I/O supply voltage
$V_{DDANA}$	2.7V to 5.5V Decoupling/filtering capacitors 100nF <sup>(1)(2)</sup> and 10µF <sup>(1)</sup> Ferrite bead <sup>(4)</sup> prevents the $V_{DD}$ noise interfering with $V_{DDANA}$	Analog supply voltage
$V_{DDIN}$	2.7V to 5.5V Decoupling/filtering capacitors 100nF <sup>(1)(2)</sup> and 10µF <sup>(1)</sup> Decoupling/filtering inductor 10µH <sup>(1)(3)</sup>	Digital supply voltage
$V_{DDCORE}$	1.1V to 1.3V typical Decoupling/filtering capacitors 100nF <sup>(1)(2)</sup> and 1µF <sup>(1)</sup>	Core supply voltage / external decoupling pin
GND		Ground
$GND_{ANA}$		Ground for the analog power domain

1. These values are only given as a typical example.

2. Decoupling capacitors should be placed close to the device for each supply pin pair in the signal group, low ESR capacitors should be used for better decoupling.
3. An inductor should be added between the external power and the  $V_{DD}$  for power filtering.
4. A ferrite bead has better filtering performance compared to standard inductor at high frequencies. A ferrite bead can be added between the main power supply ( $V_{DD}$ ) and  $V_{DDANA}$  to prevent 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) to separate the digital and analog power domains. Make 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.

### 48.3. External Analog Reference Connections

The following schematic checklist is only necessary if the application is using one or more of the external analog references. If the internal references are used instead, the following circuits in xxx and xxx are not necessary.

**Figure 48-3. External Analog Reference Schematic With Two References**

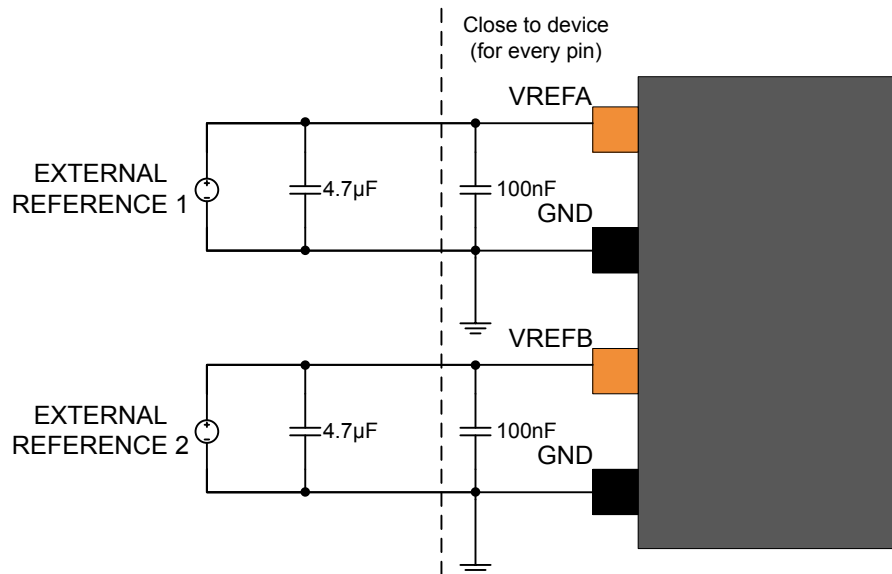


Figure 48-4. External Analog Reference Schematic With One Reference

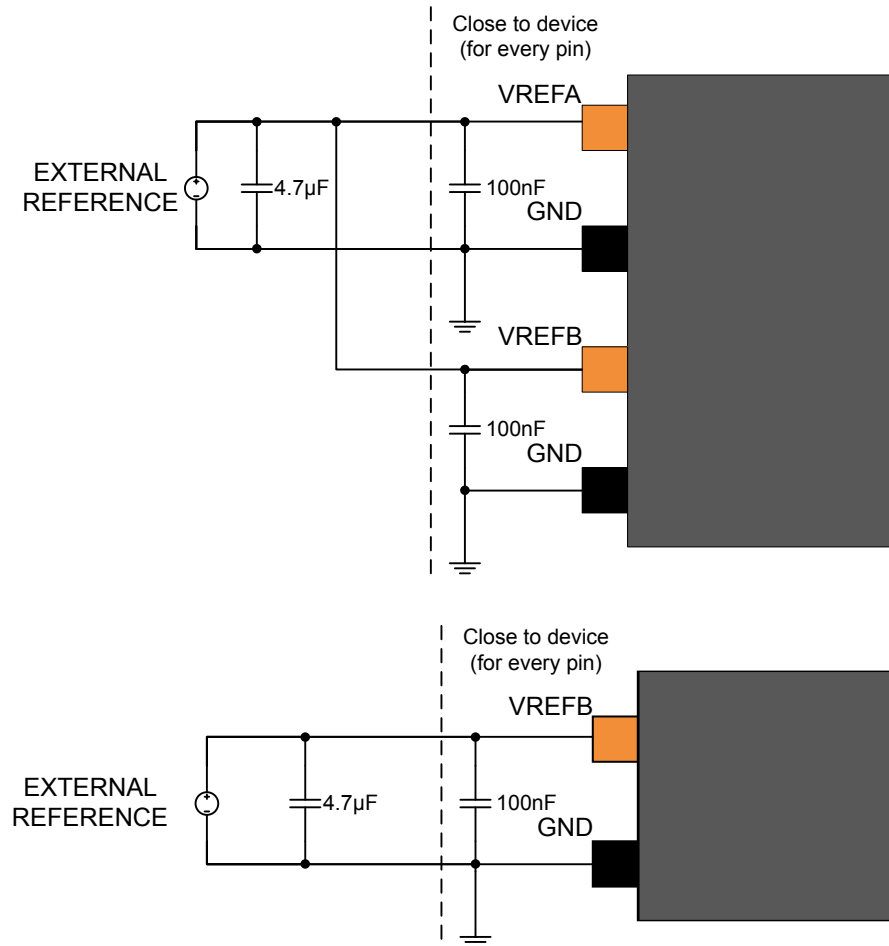


Table 48-2. External Analog Reference Connections

Signal Name	Recommended Pin Connection	Description
VREFA	1.0V to $V_{DDANA} - 0.6V$ for ADC 1.0V to $V_{DDANA} - 0.6V$ for DAC Decoupling/filtering capacitors: 100nF <sup>(1)(2)</sup> and 4.7µF <sup>(1)</sup>	External reference from VREFA pin on the analog port.
VREFB	1.0V to $V_{DDANA} - 0.6V$ for SDADC Decoupling/filtering capacitors: 100nF <sup>(1)(2)</sup> and 4.7µF <sup>(1)</sup>	External reference from VREFB pin on the analog port.
GND		Ground

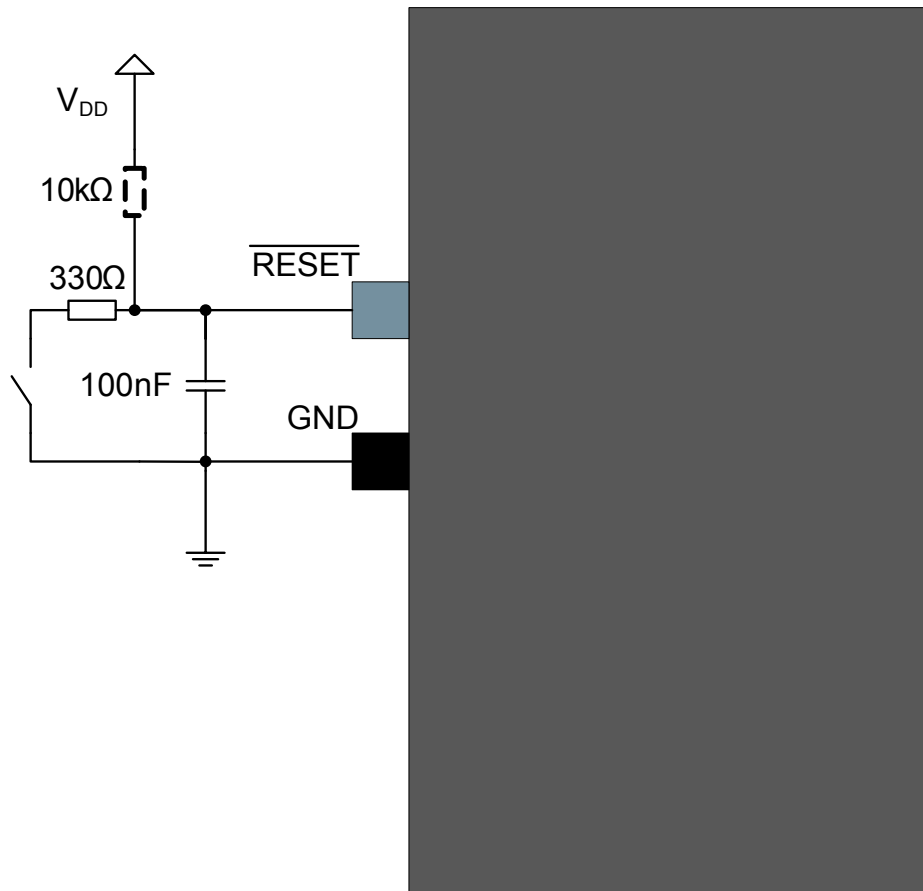
**Note:**

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.

#### 48.4. External Reset Circuit

The external reset circuit should be connected to the  $\overline{\text{RESET}}$  pin when the external reset function is used. If the external reset function has been disabled, the circuit is not necessary. The reset switch can also be removed, if the manual reset is not necessary. The  $\overline{\text{RESET}}$  pin itself has an internal pull-up resistor, hence it is optional to add any external pull-up resistor.

Figure 48-5. External Reset Circuit Schematic



A pull-up resistor makes sure that the reset does not go low and unintentionally causing a device reset. An additional resistor has been added in series with the switch to safely discharge the filtering capacitor, i.e. preventing a current surge when shorting the filtering capacitor which again can cause a noise spike that can have a negative effect on the system.

**Table 48-3. Reset Circuit Connections**

Signal Name	Recommended Pin Connection	Description
$\overline{\text{RESET}}$	Reset low level threshold voltage $V_{\text{DDIO}} = 1.6\text{V} - 2.0\text{V}$ : Below $0.33 * V_{\text{DDIO}}$ $V_{\text{DDIO}} = 2.7\text{V} - 3.6\text{V}$ : Below $0.36 * V_{\text{DDIO}}$ Decoupling/filter capacitor 100nF <sup>(1)</sup> Pull-up resistor 10k $\Omega$ <sup>(1)(2)</sup> Resistor in series with the switch 330 $\Omega$ <sup>(1)</sup>	Reset pin

1. These values are only given as a typical example.
2. The SAM C21 features an internal pull-up resistor on the  $\overline{\text{RESET}}$  pin, hence an external pull-up is optional.

### 48.5. Unused or Unconnected Pins

Unused or unconnected pins (unless marked as NC where applicable) should not be left unconnected and floating. Floating pins will add to the overall power consumption of the device. To prevent this one should always draw the pin voltage towards a given level, either VDD or GND, through a pull up/down resistor. External or internal pull up/down resistors can be used, e.g. the pins can be configured in pull-up or pull-down mode eliminating the need for external components. There are no obvious benefit in choosing external vs. internal pull resistors.

#### Related Links

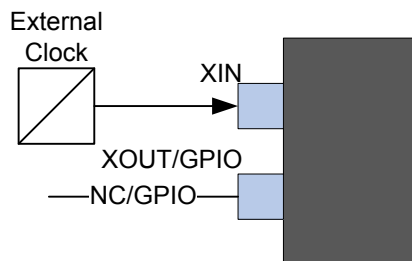
[PORT - I/O Pin Controller](#) on page 444

### 48.6. Clocks and Crystal Oscillators

The SAM C21 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).

#### 48.6.1. External Clock Source

**Figure 48-6. External Clock Source Schematic**

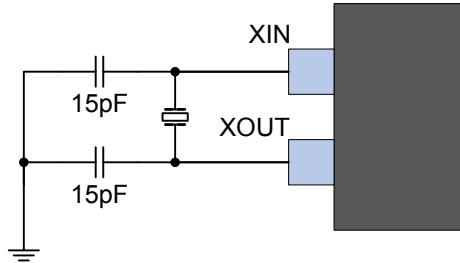


**Table 48-4. External Clock Source Connections**

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	

### 48.6.2. Crystal Oscillator

**Figure 48-7. Crystal Oscillator Schematic**



The crystal should be located as close to the device as possible. Long signal lines may cause too high load to operate the crystal, and cause crosstalk to other parts of the system.

**Table 48-5. 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 only given as a typical example.
2. Decoupling capacitor should be placed close to the device for each supply pin pair in the signal group.

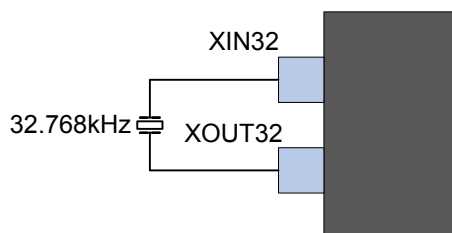
### 48.6.3. External Real Time Oscillator

The low frequency crystal oscillator is optimized for use with a 32.768kHz watch crystal. When selecting crystals, load capacitance and the crystal's Equivalent Series Resistance (ESR) must be taken into consideration. Both values are specified by the crystal vendor.

SAM C21 oscillator is optimized for very low power consumption, hence close attention should be made when selecting crystals.

The Low-frequency Crystal Oscillator provides an internal load capacitance of typical values available in the Electrical Characteristics section. This internal load capacitance and PCB capacitance can allow using a crystal inferior to 12.5pF load capacitance without external capacitors as shown in [Figure 48-8 External Real Time Oscillator without Load Capacitor](#) on page 1147.

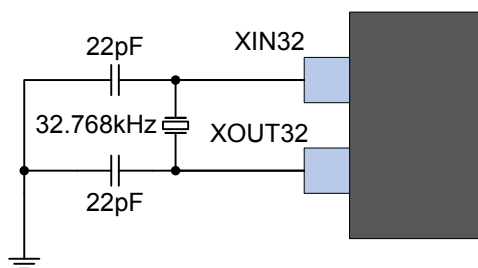
**Figure 48-8. External Real Time Oscillator without Load Capacitor**



To improve Crystal accuracy and Safety Factor, the crystal datasheet can recommend adding external capacitors as shown in [Figure 48-9 External Real Time Oscillator with Load Capacitor](#) on page 1147.

To find suitable load capacitance for a 32.768kHz crystal, consult the crystal datasheet.

**Figure 48-9. External Real Time Oscillator with Load Capacitor**



**Table 48-6. External Real Time Oscillator Checklist**

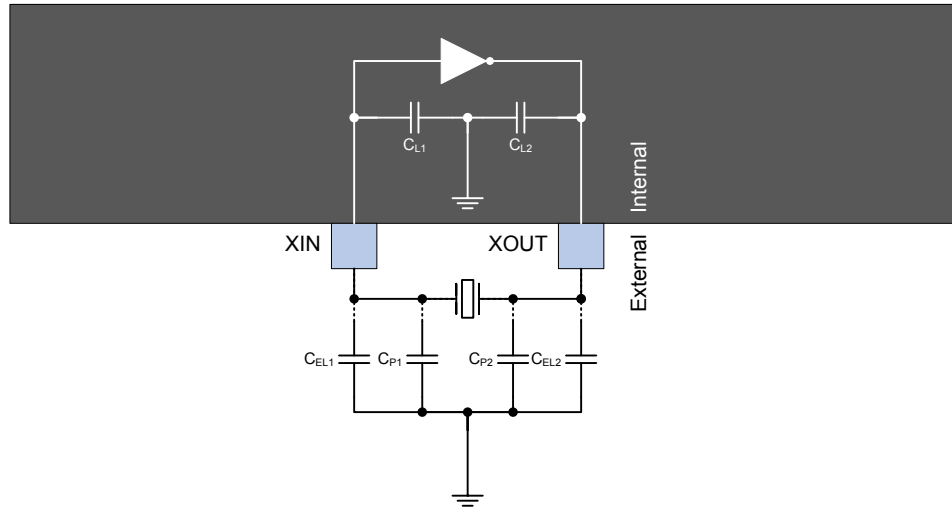
Signal Name	Recommended Pin Connection	Description
XIN32	Load capacitor 22pF <sup>(1)(2)</sup>	Timer oscillator input
XOUT32	Load capacitor 22pF <sup>(1)(2)</sup>	Timer oscillator output

1. These values are only given as typical examples.
2. Decoupling capacitor should be placed close to the device for each supply pin pair in the signal group.

#### 48.6.4. Calculating the Correct Crystal Decoupling Capacitor

The model shown in [Figure 48-10 Crystal Circuit With Internal, External and Parasitic Capacitance](#) on page 1148 can be used to calculate correct load capacitor for a given crystal. This model includes internal capacitors  $C_{Ln}$ , external parasitic capacitance  $C_{ELn}$  and external load capacitance  $C_{Pn}$ .

**Figure 48-10. 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:

$$\sum C_{\text{tot}} = \frac{(C_{L1} + C_{P1} + C_{EL1})(C_{L2} + C_{P2} + C_{EL2})}{C_{L1} + C_{P1} + C_{EL1} + C_{L2} + C_{P2} + C_{EL2}}$$

where  $C_{\text{tot}}$  is the total load capacitance seen by the crystal. This value should be equal to the load capacitance value found in the crystal manufacturer datasheet.

The parasitic capacitance  $C_{ELn}$  can in most applications be disregarded as these are usually very small. If accounted for, these values are dependent on the PCB material and PCB layout.

For some crystal the internal capacitive load provided by the device itself can be enough. To calculate the total load capacitance in this case,  $C_{ELn}$  and  $C_{Pn}$  are both zero,  $C_{L1} = C_{L2} = C_L$ , and the equation reduces to the following:

$$\sum C_{\text{tot}} = \frac{C_L}{2}$$

**Table 48-7. Equivalent Internal Pin Capacitance**

Symbol	Value	Description
$C_{XIN32}$	3.05pF	Equivalent internal pin capacitance
$C_{XOUT32}$	3.29pF	Equivalent internal pin capacitance

## 48.7. Programming and Debug Ports

For programming and/or debugging the SAM C21 the device should be connected using the Serial Wire Debug (SWD) interface. Currently the SWD interface is supported by several Atmel and third party programmers and debuggers, like the SAM-ICE, JTAGICE3 or SAM C21 Xplained Pro (SAM C21 evaluation kit) Embedded Debugger.

Refer to the SAM-ICE, JTAGICE3 or SAM C21 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.

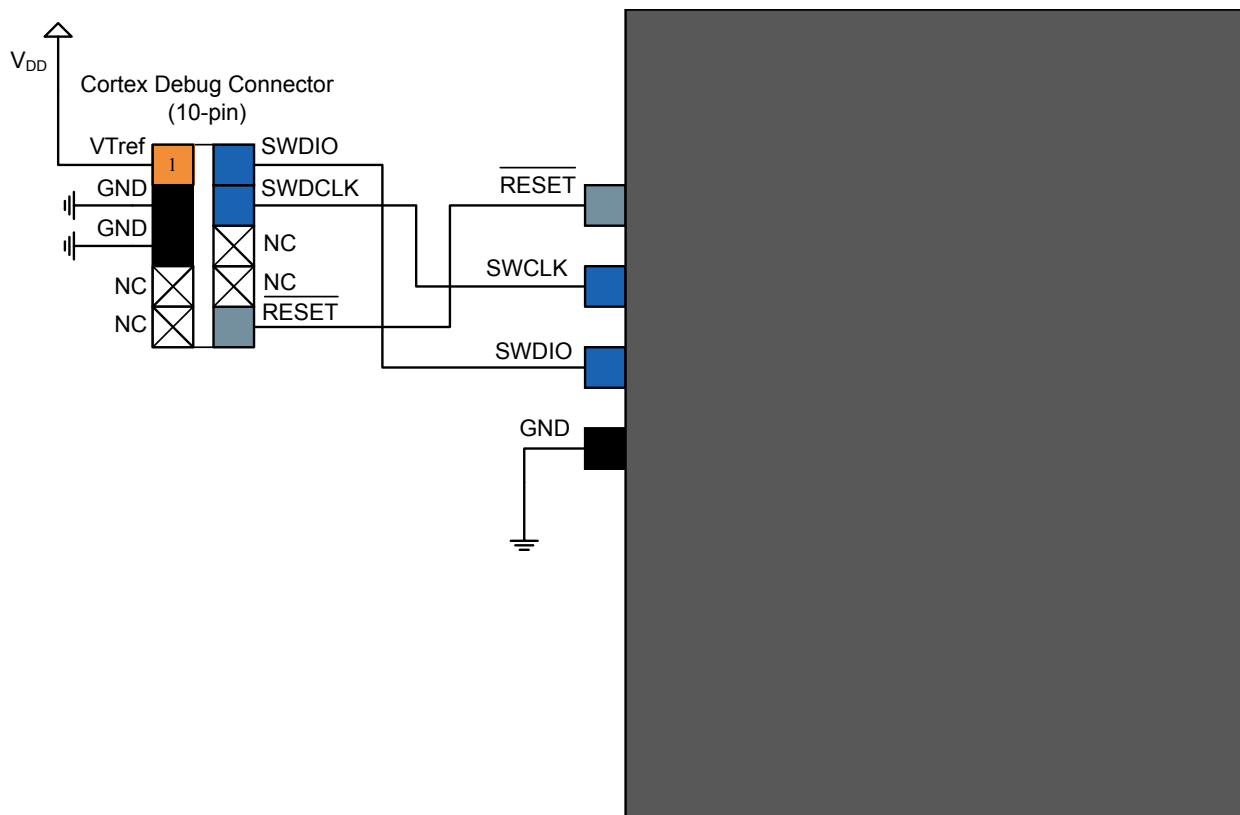


The SAM C21 Xplained Pro evaluation board for the SAM C21 supports programming and debugging through the onboard embedded debugger so no external programmer or debugger is needed.

#### 48.7.1. 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 [Figure 48-11 Cortex Debug Connector \(10-pin\)](#) on page 1149 with details described in [Table 48-8 Cortex Debug Connector \(10-pin\)](#) on page 1149.

**Figure 48-11. Cortex Debug Connector (10-pin)**



**Table 48-8. Cortex Debug Connector (10-pin)**

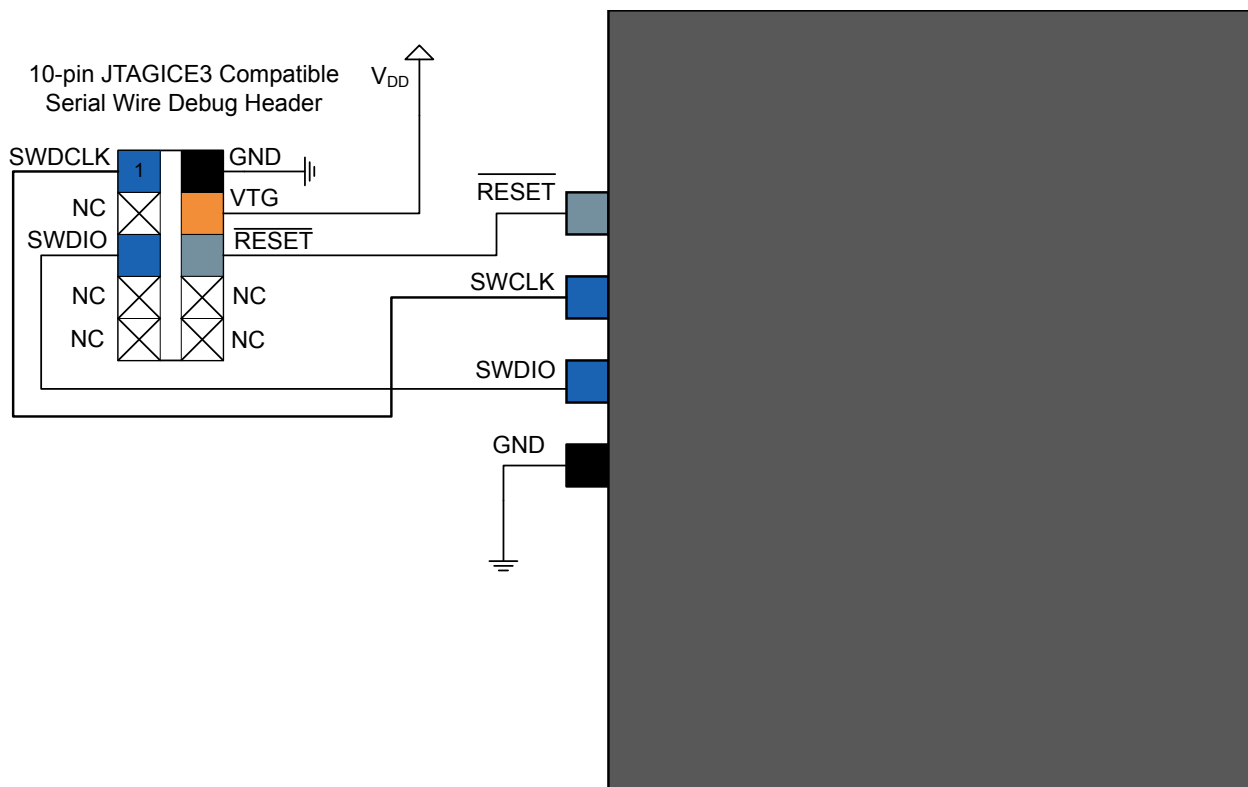
Header Signal Name	Description
SWDCLK	Serial wire clock pin
SWDIO	Serial wire bidirectional data pin
RESET	Target device reset pin, active low
VTref	Target voltage sense, should be connected to the device $V_{DD}$
GND	Ground

#### 48.7.2. 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 SAM C21 to the JTAGICE3, alternatively one can use the JTAGICE3 squid cable and manually match the signals between the JTAGICE3 and SAM C21. [Figure 48-12 10-pin JTAGICE3 Compatible Serial Wire Debug Interface](#) on page 1150 describes how to connect a 10-pin header that support connecting the JTAGICE3 directly to the SAM C21 without the need for a squid cable. This can also be used for the Atmel-ICE AVR connector port.

The JTAGICE3 squid cable or the JTACICE3 50mil cable can be used to connect the JTAGICE3 programmer and debugger to the SAM C21. [10-pin JTAGICE3 Compatible Serial Wire Debug Interface](#) on page 1149 illustrates the correct pinout for the JTAGICE3 50 mil, and details are given in [Table 48-9 10-pin JTAGICE3 Compatible Serial Wire Debug Interface](#) on page 1150.

**Figure 48-12. 10-pin JTAGICE3 Compatible Serial Wire Debug Interface**



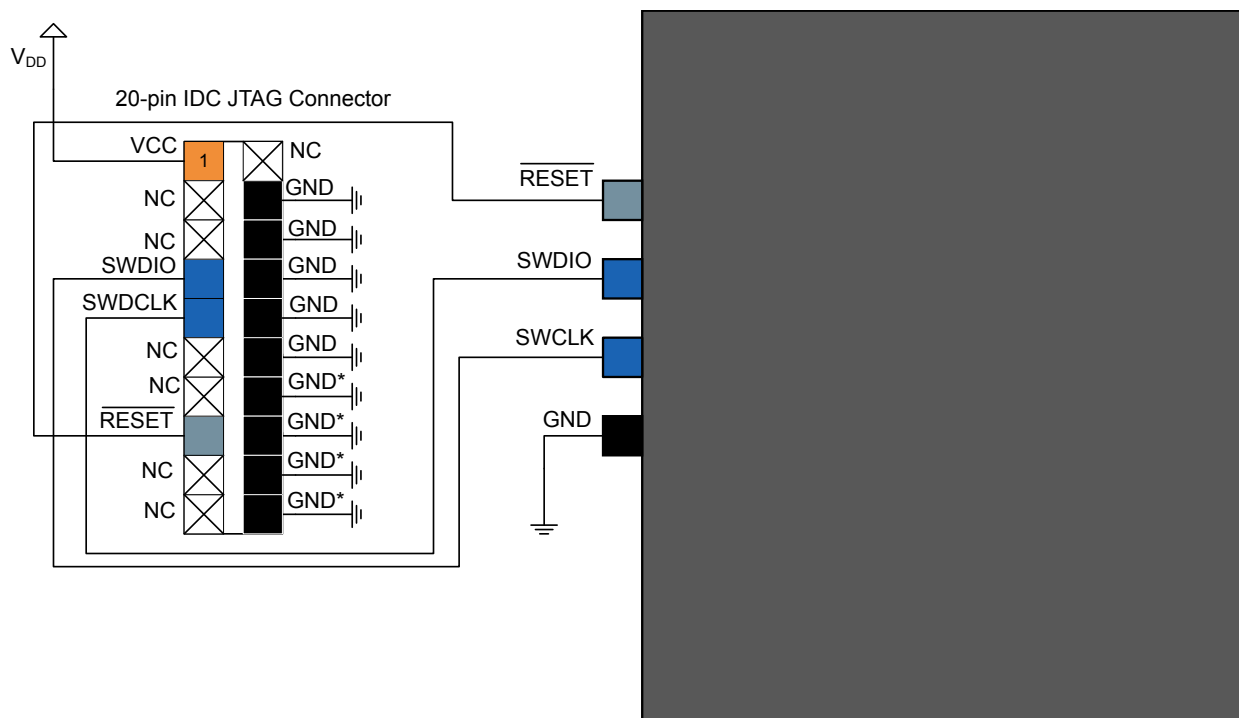
**Table 48-9. 10-pin JTAGICE3 Compatible Serial Wire Debug Interface**

Header Signal Name	Description
SWDCLK	Serial wire clock pin
SWDIO	Serial wire bidirectional data pin
RESET	Target device reset pin, active low
VTG	Target voltage sense, should be connected to the device $V_{DD}$
GND	Ground

### 48.7.3. 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 [Figure 48-13 20-pin IDC JTAG Connector](#) on page 1151 with details described in [Table 48-10 20-pin IDC JTAG Connector](#) on page 1151.

**Figure 48-13. 20-pin IDC JTAG Connector**



**Table 48-10. 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 <sub>DD</sub>
GND	Ground
GND*	These pins are reserved for firmware extension purposes. They can be left unconnected or connected to GND in normal debug environment. They are not essential for SWD in general.

## 49. Errata

### 49.1. Revision A

Not sampled.

### 49.2. Revision B

#### 49.2.1. PORT

**1 – When the PORT is defined as EVSYS.USER in a synch/resynch path, the first event is transmitted to the PORT but the acknowledgement coming from the PORT is not released. So next coming events are treated as overrun by EVSYS. Errata reference: 14317**

**Fix/Workaround:**

None.

Do not use the synch/resynch path, only use asynchronous path.

#### 49.2.2. OSC48M

**1 – When a System Reset is applied, the OSC48MDIV register is reset, but the value is not synchronized. This may result in the system clock running too fast. Errata reference: 14497**

**Fix/Workaround:**

Do not write OSC48MDIV to lower than 0xB. Do not run the device faster than 4MHz when running from external oscillators

#### 49.2.3. TC

**1 – A capture overflow can occur without INTFLAG.ERR being set if a new capture occurs within 3 APB clock periods + 3 generic clock periods after a previous capture. Errata reference: 13367**

**Fix/Workaround:**

The delay between two capture events must be longer than 3 APB clock periods + 3 generic clock periods.

**2 – The input capture on IO pins does not work. Errata reference: 14024**

**Fix/Workaround:**

Use the input capture through TC event and use the EIC or CCL as event generators.

#### 49.2.4. RTC

**1 – The COUNTSYNC/CLOCKSYNC bit of the RTC.CTRLA register has no effect. Read synchronization of the COUNT/CLOCK register is always enabled. Errata reference: 13714**

**Fix/Workaround:**

None

## 49.2.5. OSC32KCTRL

**1 – At start-up and in case of clock failure detection(CFD), the auto switch by the CFD does not work if the XOSC32K is requested by the GCLK. Errata reference: 14039**

**Errata/Workaround**

**Manually change clock from XOSC32K to another 32K source.**

## 49.2.6. Device

**1 – VREGSMOD bits have no effect in PM.STDBYCFG register. The power domain controller always operates in automatic regulator mode.**

**Errata reference: 14498**

**Fix/Workaround:**

None

**2 – The SYSTICK calibration value is incorrect. Errata reference: 14156**

**Fix/Workaround:**

The correct SYSTICK calibration value is 0x40000000. This value should not be used to initialize the SysTick RELOAD value register, which should be initialized instead with a value depending on the main clock frequency and on the tick period required by the application. For a detailed description of the SYSTICK module, refer to the official ARM Cortex-M0+ documentation.

**3 – In IDLE sleep mode, the APB and AHB clocks are not stopped if the FDPLL is running as a GCLK clock source. Errata reference: 13401**

**Fix/Workaround:**

Disable the FDPLL before entering IDLE sleep mode.

**4 – The Analog Comparators and ADC1 use the same generic clock configuration. GCLK\_ADC1 must be used to configure the clock for AC as GCLK\_AC is not functional. Errata reference: 13404**

**Fix/Workaround:**

None

**5 – Increased power consumption in standby sleep mode. Errata reference: 14100**

**Fix/Workaround:**

None

**6 – In I2C Slave mode, writing the CTRLB register when in the AMATCH or DRDY interrupt service routines can cause the state machine to reset. Errata reference: 13574**

**Fix/Workaround:**

Write CTRLB.ACKACT to 0 using the following sequence:

```
// If higher priority interrupts exist, then disable so that the
```

```
// following two writes are atomic.
```

```
SERCOM - STATUS.reg = 0;
```

```
SERCOM - CTRLB.reg = 0;
```

```
// Re-enable interrupts if applicable.
```

Write CTRLB.ACKACT to 1 using the following sequence:

```
// If higher priority interrupts exist, then disable so that the
```

```
// following two writes are atomic.
```

```
SERCOM - STATUS.reg = 0;
```

```
SERCOM - CTRLB.reg = SERCOM_I2CS_CTRLB_ACKACT;
```

// Re-enable interrupts if applicable.  
Otherwise, only write to CTRLB in the AMATCH or DRDY interrupts if it is to close out a transaction.  
When not closing a transaction, clear the AMATCH interrupt by writing a 1 to its bit position instead of using CTRLB.CMD. The DRDY interrupt is automatically cleared by reading/writing to the DATA register in smart mode. If not in smart mode, DRDY should be cleared by writing a 1 to its bit position.

Code replacements examples:

Current:

```
SERCOM - CTRLB.reg |= SERCOM_I2CS_CTRLB_ACKACT;
```

Change to:

```
// If higher priority interrupts exist, then disable so that the
```

```
// following two writes are atomic.
```

```
SERCOM - STATUS.reg = 0;
```

```
SERCOM - CTRLB.reg = SERCOM_I2CS_CTRLB_ACKACT;
```

```
// Re-enable interrupts if applicable.
```

Current:

```
SERCOM - CTRLB.reg &= ~SERCOM_I2CS_CTRLB_ACKACT;
```

Change to:

```
// If higher priority interrupts exist, then disable so that the
```

```
// following two writes are atomic.
```

```
SERCOM - STATUS.reg = 0;
```

```
SERCOM - CTRLB.reg = 0;
```

```
// Re-enable interrupts if applicable.
```

Current:

```
/* ACK or NACK address */
```

```
SERCOM - CTRLB.reg |= SERCOM_I2CS_CTRLB_CMD(0x3);
```

Change to:

```
// CMD=0x3 clears all interrupts, so to keep the result similar,
```

```
// PREC is cleared if it was set.
```

```
if (SERCOM - INTFLAG.bit.PREC) SERCOM - INTFLAG.reg =
```

```
SERCOM_I2CS_INTFLAG_PREC;
```

```
SERCOM - INTFLAG.reg = SERCOM_I2CS_INTFLAG_AMATCH;
```

**7 – The default TC selection as CCL input is not TC0, but TC4. Thus the TC selection for the CCL is TC4/TC0/TC1/TC2 instead of TC0/TC1/TC2/TC3. The TC alternate selection is TC0/TC1/TC2/TC3 instead of TC1/TC2/TC3/TC4. Errata reference: 13449**

**Fix/Workaround:**

Use the TC input mapping described above.

## 49.2.7. CAN

**1 – Description: Errata reference: 13525**

**(1) When a message is transmitted while CCCR.DAR = '1' (automatic re-transmission disabled for messages not transmitted successfully), the Event Type of the corresponding Tx Event FIFO element is ET = ""01"" instead of ET = ""10"". (2) When multiple messages are transmitted sequentially using the same Tx Buffer while CCCR.DAR = '1', it may happen that a newly requested transmission is not started when it is requested in the time window starting at the successful completion of**

the previous message and ending at the end of the intermission phase before the bus is idle again. This message is then treated as if it had lost arbitration.

**Scope:**

The errata is limited to message transmission when DAR mode is configured. Normal CAN / CAN FD operation is not affected

**Effects:**

(1) The Event Type of the associated Tx Event FIFO element is not correct. (2) When a message was transmitted successfully from a specific Tx Buffer, a following transmission using the same Tx Buffer and requested in the described time window will not be started.

**Fix/Workaround:**

Do not use the same Tx Buffer for consecutive DAR transmissions or wait at least for 4 CAN bit times after successful transmission before requesting the next transmission from the same Tx Buffer.

**2 – Description: Errata reference: 13511**

When a CAN 2.0 frame is transmitted while CAN FD operation is enabled, a recessive stuff bit following the first reserved bit will cause a shift in the DLC for specific identifiers with the result, that a frame with faulty DLC and faulty number of data bytes is transmitted.

**Scope:**

The erratum is limited to the case when a CAN 2.0 frame is transmitted while CAN FD operation is enabled (CCCR.CME ≠ ""00""). The problem does not occur when CAN FD frames are transmitted or when CAN FD operation is disabled.

**Effects:**

In case the identifier of a transmit message ends with two dominant bits (11-bit ID) or three dominant bits (29-bit ID), bit stuffing causes the DLC to be shifted by one bit to the right. This results in transmission of a message with faulty DLC and therefore faulty number of data bytes.

**Fix/Workaround:**

No workaround needed in CAN 2.0 networks, CAN Conformance Test passed. No workaround needed when only CAN FD messages are transmitted. For mixed operation (CAN 2.0 and CAN FD frames) the problematic identifiers may not be used for the transmission of CAN 2.0 frames.

**3 – Description: Errata reference: 13524**

When CCCR.INIT is set while the M\_CAN is receiving a frame, the next received frame after resetting CCCR.INIT will cause IR.MRAF to be set.

**Scope:**

The errata is limited to the case when CCCR.INIT is set / reset while the M\_CAN is receiving a frame.

**Effects:**

IR.MRAF is set when the first frame after resetting CCCR.INIT is received although that frame is received correctly.

**Fix/Workaround:**

If CCCR.INIT shall be set during operation proceed as follows: (1) Issue a clock stop request by setting bit CCCR.CSR. (2) Wait until the M\_CAN sets CCCR.INIT and CCCR.CSA to one.

**4 – Description: Errata reference: 13521**

When a CAN frame is received with bit FDF and the following res bit both recessive, the protocol controller correctly detects a Protocol Exception Event. Reception of the disturbed message is not finished, the message is discarded. If this happened, two cases have to be distinguished: (1) Message reception directly after Protocol Exception Event => When the next frame is received interrupt flag IR.MRAF is set to '1' although the frame has been received correctly. (2) Message transmission directly after Protocol Exception Event => When a frame is transmitted directly after a Protocol Exception Event, that frame is transmitted with faulty frame format. In this case interrupt flag IR.MRAF is not set. The frame will cause an error frame. Only the first message after a Protocol Exception Event is affected, all following messages (received or transmitted) have no problem.

**Scope:**

The errata is limited to the case when the reserved bit res after the FDF bit in CAN FD frames is received recessive.

**Effects:**

Reception directly after Protocol Exception Event => Interrupt flag IR.MRAF is set although there was no problem in accessing the Message RAM. The Message is received correctly. Transmission directly after Protocol Exception Event => Transmission of a frame with faulty frame format.

**Fix/Workaround:**

None.

**5 – Description: Errata reference: 13518**

When `CCCR.CME` ≠ `""00""` and a change of the CAN operation mode is requested by writing to `CCCR.CMR` while frame transmission/reception is ongoing, this request may be ignored and the `M_CAN` remains in its previous operation mode.

**Scope:**

The errata is limited to the case when a change of the CAN operation mode from/to CAN FD operation is requested while frame transmission/reception is ongoing.

**Effects:**

In case one of the affected CAN operation mode changes is requested by writing `CCCR.CMR` while a frame transmission/reception is ongoing, the request is acknowledged by resetting `CCCR.CMR` to `""00""` but the `M_CAN` remains in its previous operation mode.

**Fix/Workaround:**

No workaround needed in CAN 2.0 networks, CAN Conformance Test passed. No workaround needed for switching between CAN operation according to ISO11898-1 and CAN FD operation with bit rate switching. In all other cases check whether the requested CAN operation mode change has been executed by reading `CCCR.FDO` and `CCCR.FDBS`. If not, repeat command until requested mode change is signalled by `CCCR.FDO` and `CCCR.FDBS`.

**6 – The CAN is not compatible with an on-demand clock source. Errata reference: 14406**

**Fix/Workaround:**



Clear the ONDEMAND bit to zero for the oscillator source that provides the GCLK to the CAN.

**7 – The CAN-FD frame format implements Bosch CAN FD Specification V1.0 and is not compatible with ISO11898-1. The CCR.NISO bit has no effect. Errata reference: 13757**

**Fix/Workaround:**

Connect only to CAN-FD networks that support Bosch CAN FD Specification V1.0

**8 – Description: Errata reference: 13523**

**After detecting a Message RAM Access Failure during frame transmission, interrupt flag IR.MRAF is set and the M\_CAN enters Restricted Operation Mode (CCCR.ASM = '1'). When the Restricted Operation Mode is left by writing CCCR.ASM = '0', it may happen, that the first frame transmitted is send out with unexpected identifier and control field. If this is a valid frame, it may happen that it is accepted and acknowledged by a receiver.**

**Scope:**

The errata is limited to the case when the M\_CAN has entered Restricted Operation Mode due to a Message RAM Access Failure, signaled by interrupt flag IR.MRAF.

**Effects:**

**With the next transmission after leaving Restricted Operation Mode by resetting CCCR.ASM, a frame with unexpected identifier and control field is transmitted which accidentally might be accepted and acknowledged by a receiver.**

**Fix/Workaround:**

To recover from Restricted Operation Mode proceed as follows: (1) Cancel all pending transmission requests by writing 0hFFFF FFFF to register TXBCR. (2) Issue a clock stop request by setting bit CCCR.CSR. (3) Wait until the M\_CAN sets CCCR.INIT and CCCR.CSA to one. (4) First reset CCCR.CSR. (5) Then reset CCCR.INIT. (6) Wait until CCCR.INIT is read as zero. (7) Issue a second clock stop request by setting bit CCCR.CSR. (8) Wait until the M\_CAN sets CCCR.INIT and CCCR.CSA to one. (9) Set CCCR.CCE, reset CCCR.CSR, and reset CCCR.ASM. (10) Restart M\_CAN by writing CCCR.INIT = '0'. (11) Configure the CAN operation mode by writing to CCCR.CMR. (12) Request the transmissions cancelled by step one.

**9 – Description: Errata reference: 13526**

**When CCCR.CCE is set while the M\_CAN Tx Handler is scanning the Message RAM for Tx Buffers with pending transmission requests (bits TXBRP.TRPNn set), register TXBRP is reset and the Tx Handler FSM is halted. After CCCR.INIT and CCCR.CCE have been reset by the Host, the M\_CAN is unable to transmit messages. When the Host requests a transmission by writing to register TXBAR, the respective Tx Buffer Request Pending bit in register TXBRP is set, but the Tx Handler will not start the requested transmission.**

**Scope:**

The errata is limited to the case when CCCR.CCE is set while the M\_CAN Tx Handler is scanning the Message RAM.

**Effects:**

**When CCCR.CCE is set while a Tx scan is in progress, the Tx Handler FSM stops. After CCCR.INIT and CCCR.CCE are reset, the Tx Handler FSM does not execute transmission requests.**

**Fix/Workaround:**

(1) Cancel all pending transmission requests by writing 0hFFFF FFFF to register TXBCR. (2) Issue a clock stop request by setting bit CCCR.CSR. (3) Wait until the M\_CAN sets CCCR.INIT and CCCR.CSA to one. (4) First reset CCCR.CSR. (5) Then reset CCCR.INIT. (6) Wait until CCCR.INIT is read as zero. (7) Issue a second clock stop request by setting bit CCCR.CSR. (8) Wait until the M\_CAN sets CCCR.INIT and CCCR.CSA to one. (9) Set CCCR.CCE and reset CCCR.CSR.

**10 – Description: Errata reference: 13519**

**When a CAN 2.0 frame with a recessive stuff bit following the first reserved bit is received while CAN FD operation is enabled and a transmission is pending, the M\_CAN will internally overwrites the received arbitration bits with the pending transmission's arbitration bits.**

**Scope:**

The erratum is limited to the case when CAN 2.0 frames with specific identifiers causing the described stuff bit are received while CAN FD operation is enabled (CCCR.CME ≠ ""00""). The problem does not occur when CAN FD operation is disabled.

**Effects:**

In case the identifier of a received data frame ends with two dominant bits (11-bit ID) or three dominant bits (29-bit ID), there will be a recessive stuff bit after the first reserved bit. This causes the falsification of the received arbitration bits if a transmission is pending. In case the pending transmission is a remote frame, the received data frame is treated as a received remote frame which will cause a format or CRC error resulting in an error frame. In case the pending transmission is a data frame, the incoming frame is received and is presented to the receive message handler with the identifier of the pending transmit message. Depending on the configuration of the acceptance filtering, the frame may be stored in an Rx Buffer or Rx FIFO.

**Fix/Workaround:**

No workaround needed in CAN 2.0 networks, CAN Conformance Test passed. No workaround needed when only CAN FD frames are received. For mixed operation (CAN 2.0 and CAN FD frames) the problematic identifiers may not be used.

**11 – Description: Errata reference: 13520**

**When BTP.TSEG2 and BTP.BRP are both zero and the M\_CAN transmits a frame, the FDF bit in CAN FD format (reserved bit in Classic CAN format) in the control field may be falsified. The effect is different for frames to be transmitted in Classic CAN format and for frames to be transmitted in CAN FD format. Transmission of Classic CAN Frame => When BTP.TSEG2 and BTP.BRP are both zero and the M\_CAN transmits a Classic CAN frame (CCCR.CME = ""00"" ) with a 29-bit identifier where the MSB (ID28) is '1', the reserved bit following the RTR bit will be transmitted recessive instead of dominant while the rest of**

the frame is transmitted in Classic CAN format. Transmission of CAN FD Frame => When BTP.TSEG2 and BTP.BRP are both zero and the M\_CAN transmits a CAN FD frame with a 29-bit identifier where the MSB (ID28) is '0' or a CAN FD frame with 11-bit identifier, the FDF bit of the frame is transmitted dominant instead of recessive, the rest of the frame is transmitted in Classic CAN format with a falsified DLC.

**Scope:**

The erratum is limited to the case when in the bit time configuration for Classic CAN operation and the Arbitration Phase in CAN FD operation BTP.TSEG2 and BTP.BRP are both zero. This configures the time segment after the sample point to the length of one time quantum and the length of the time quantum to one clock period. This is an unusual configuration.

**Effects:**

Transmission of Classic CAN Frame => When a Classic CAN frame is received by a CAN FD enabled receiving node it will interpret the falsified reserved bit as FDF bit. If this bit is recessive instead of dominant, the frame will be interpreted as CAN FD frame. In this case the receiving node will respond with an error frame when it detects that the rest of the frame is not in CAN FD format. A strictly Classic CAN receiving node will interpret the recessive FDF bit as reserved bit, ignore its actual value and will receive this frame correctly without detecting an error. Transmission of CAN FD Frame => When the M\_CAN wants to transmit a CAN FD frame, it transmits the FDF bit dominant instead of recessive and the rest of the frame in Classic CAN format with a falsified DLC.

**Fix/Workaround:**

Do not use bit timing configurations where BTP.TSEG2 and BTP.BRP are both zero for CAN FD communication.

**12 – Description: Errata reference: 13522**

When CCCR.CMR is changed during start of transmission, the following may happen: (1) Classic CAN -> CAN FD with bit rate switching => When the Tx Event FIFO is used, bits EDL and BRS of the related Tx Event FIFO element do not match with the transmitted frame type. They signal a CAN FD frame with bit rate switching (both set to one) while a Classic CAN frame was transmitted. (2) Classic CAN -> CAN FD without bit rate switching => When the Tx Event FIFO is used, bit EDL of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame while a Classic CAN frame was transmitted. (3) CAN FD with bit rate switching -> CAN FD without bit rate switching => When the Tx Event FIFO is used, bit BRS of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame without bit rate switching while a CAN FD frame with bit rate switching was transmitted. (4) CAN FD without bit rate switching -> CAN FD with bit rate switching => When the Tx Event FIFO is used, bit BRS of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame with bit rate switching while a CAN FD frame without bit rate switching was transmitted. (5) CAN FD with/without bit rate switching -> Classic CAN => IR.MRAF is set, the

**M\_CAN switches to Restricted Operation Mode, and the transmission is aborted.**

**Scope:**

**The errata is limited to the case when the CAN operation mode is changed during start of transmission.**

**Effects:**

**Tx Event FIFO element faulty (cases 1,2,3,4) or interrupt flag IR.MRAF set, Restricted Operation Mode entered, and transmission aborted (case 5).**

**Fix/Workaround:**

**Do not change the CAN operation mode by writing to CCCR.CMR as long as there are pending transmission requests (TXBRP.TRPnn = '1').**

#### **49.2.8. FDPLL**

**1 – When entering standby mode, the FDPLL is still running even if not requested by any module causing extra consumption. Errata reference: 12244**

**Fix/Workaround:**

**FDPLL must be disabled before entering in standby mode and re-enabled after wake-up**

#### **49.2.9. 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**

**2 – If the SERCOM is enabled in SPI mode with SSL detection enabled (CTRLB.SSDE) and CTRLB.RXEN=1, an erroneous slave select low interrupt (INTFLAG.SSL) can be generated. Errata reference: 13369**

**Fix/Workaround:**

**Enable the SERCOM first with CTRLB.RXEN=0. In a subsequent write, set CTRLB.RXEN=1.**

#### **49.2.10. 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 – A capture overflow can occur without INTFLAG.ERR being set if a new capture occurs within 3 APB clocks + 3 generic Clock periods from a previous capture. Errata reference: 13366**

**Fix/Workaround:**

**The delay between two capture events must be longer than 3 APB clock periods + 3 generic clock periods.**

**3 – When the circular buffer is enabled, an APB clock is requested to update the corresponding APB register. If all masters in the system**

(CPU, DMA) are disabled, the APB clock is never provided to the TCC, making the circular buffer feature not functional in standby sleep mode. Errata reference: 12269

**Fix/Workaround:**

Keep a master enabled in the system (enable DMA, or do not enable standby sleep mode when circular buffer is enabled).

**4 – In RAMP 2 mode with Fault keep, qualified and restart: Errata reference: 13262**

If a fault occurred at the end of the period during the qualified state, the switch to the next ramp can have two restarts.

**Fix/Workaround:**

Avoid faults few cycles before the end or the beginning of a ramp.

#### 49.2.11. PTC

**1 – The PTC generic clock is always requested during standby when RUNSTDBY is set to one. Power consumption will be higher if the PTC is enabled during standby sleep mode even if no conversion is on-going. Errata reference: 14370**

**Fix/workaround**

Disable PTC in standby mode to reduce power consumption

#### 49.2.12. DMAC

**1 – If data is written to CRCDATAIN in two consecutive instructions, the CRC computation may be incorrect. Errata reference: 13507**

**Fix/Workaround:**

Add a NOP instruction between each write to CRCDATAIN register.

#### 49.2.13. CCL

**1 – The reset of the RS latch is not functional. The latch can only be cleared by disabling the LUT. Errata reference: 14043**

**Fix/Workaround:**

None

#### 49.2.14. ADC

**1 – Once set, the ADC.SWTRIG.START will not be cleared until the Microcontroller is reset. Errata reference: 14094**

**Fix/Workaround:**

None

**2 – When window monitor is enabled and its output is 0, the ADC GCLK is kept running. Power consumption will be higher than expected in sleep modes Errata reference: 14449**

**Fix/Workaround:**

None

**3 – The LSB bit of ADC result is stuck at zero, in unipolar mode for 8-bit and 10-bit resolution. Errata reference: 14431**

**Fix/Workaround:**

None

**4 – If a synchronized event is received during an ADC conversion, the ADC will not acknowledge the event, causing a stall of the event channel. Errata reference: 14795**

**Fix/Workaround:**

When using events with the ADC, only the asynchronous path from the Event System must be used.

#### 49.2.15. EIC

**1 – The EIC ASYNCH register is not write protected. Errata reference: 13848**

**Fix/Workaround:**

None

**2 – If the NMI pin PORT config is INPUT+PULL-UP enabled and the NMI is configured to trigger on rising edge (or both edges), the NMI exception is triggered as soon as the NMI config is written. Errata reference: 13074**

**Fix/Workaround:**

Set the NMI pin PORT config, enable EIC in edge detection mode then disable EIC. Clear INTFLAG, then write NMI configuration.

#### 49.2.16. EVSYS

**1 – Using synchronous, spurious overrun can appear with generic clock for the channel always on. Errata reference: 14532**

**Fix/Workaround:**

- Request the generic clock on demand by setting the CHANNEL.ONDEMAND bit to one.
- No penalty is introduced.

#### 49.2.17. SDADC

**1 – The default value of zero in GAINCORR causes RESULT to be zero. The default value of zero in CTRLB.SKPCNT generates invalid data on the first two conversions. Errata reference: 14416**

**Fix/Workaround:**

Write GAINCORR to 1 before running any conversions. Write CTRLB.SKPCNT to 2 before running single conversions.

**2 – If the APB clock is not 2x or higher than the Generic Clock frequency, the first input conversion in a sequence will be invalid. Errata reference: 14367**

**Fix/Workaround:**

The APB clock must be twice the generic clock frequency or higher, or the prescaler must be configured to provide a similar ratio.

#### 49.2.18. TSENS

**1 – The magnitude of the temperature measurement value decreases with increasing temperature, i.e. it has a negative temperature coefficient. Errata reference: 14476**

**Fix/Workaround:**

Invert the VALUE register value by software to achieve a positive coefficient. For window mode, the WINLT and WINUT registers will need to respect the

negative coefficient, that is WINLT sets the high temp threshold and WINUT sets the low temp threshold.

#### 49.2.19. AC

**1 – Hysteresis is only present for a falling (1->0) transition of the comparator output. Errata reference: 13712**

**Fix/Workaround:**

None

## 50. Revision History

### 50.1. Rev E - 12/2015

<a href="#">Configuration Summary</a> on page 14	<ul style="list-style-type: none"> <li>Corrected memory sizes.</li> </ul>
<a href="#">Ordering Information</a> on page 16	<ul style="list-style-type: none"> <li>Introduced 105°C ordering codes.</li> <li>Corrected package type from QFN48 to TQFP48 for ATSAM C21G16A-AUT.</li> </ul>
<a href="#">DSU - Device Service Unit</a> on page 59	Bit CTRL.CRC is write-only.
<a href="#">NVMCTRL – Non-Volatile Memory Controller</a> on page 419	Updated description in <a href="#">NVM Write</a> on page 424: Removed reference to default MANW value. This is covered in the <a href="#">CTRLB.MANW</a> bit description.
<a href="#">DMAC – Direct Memory Access Controller</a> on page 334	Added note in <a href="#">Sleep Mode Operation</a> on page 358.
<a href="#">CCL – Configurable Custom Logic</a> on page 904	Removed oscillator related sub sections from <a href="#">Sequential Logic</a> on page 914.
<a href="#">SERCOM USART – SERCOM Universal Synchronous and Asynchronous Receiver and Transmitter</a> on page 505	<ul style="list-style-type: none"> <li>Added RS485 to the TXPO bit description in the <a href="#">CTRLA</a> on page 522register.</li> </ul>

### 50.2. Rev D - 09/2015

<a href="#">SERCOM USART – SERCOM Universal Synchronous and Asynchronous Receiver and Transmitter</a> on page 505	<ul style="list-style-type: none"> <li>Updated formula in the <a href="#">RXPL</a> on page 532register.</li> </ul>
<a href="#">Errata</a> on page 1152	<ul style="list-style-type: none"> <li>Reinserted errata section which was missing from datasheet rev C.</li> </ul>

### 50.3. Rev C - 09/2015

General	Editorial updates.
<a href="#">DMAC – Direct Memory Access Controller</a> on page 334	<ul style="list-style-type: none"> <li>Updated number of bits in the SWTRIGCTRL, INTSTATUS, BUSYCH and PENDCH registers (Related to number of DMA channels available).</li> </ul>
<a href="#">PORT - I/O Pin Controller</a> on page 444	<ul style="list-style-type: none"> <li>Functional Description: Overview diagram updated.</li> </ul>



ADC – Analog-to-Digital Converter on page 923	<ul style="list-style-type: none"> <li>Block Diagram: Renamed ADC input signals from ADC to AIN.</li> <li>Signal Description: Renamed ADC signal to AIN</li> </ul>
PTC - Peripheral Touch Controller on page 1058	<ul style="list-style-type: none"> <li>Block Diagram updated.</li> <li>Section Self-capacitance Sensor Arrangement updated.</li> </ul>

**50.4. Rev B - 06/2015**

Ordering Information on page 16	<ul style="list-style-type: none"> <li>Remove carrier type Tray option.</li> </ul>
---------------------------------	--

**50.5. Rev A - 04/2015**

Initial revision.
-------------------



**Atmel®** | Enabling Unlimited Possibilities®



**Atmel Corporation**    1600 Technology Drive, San Jose, CA 95110 USA    **T:** (+1)(408) 441.0311    **F:** (+1)(408) 436.4200    |    **www.atmel.com**

© 2015 Atmel Corporation. / Rev.: Atmel-42365E-SAM C21\_Datasheet\_Preliminary-12/2015

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected® logo, and others are the registered trademarks or trademarks of ARM Ltd. Other terms and product names may be trademarks of others.

**DISCLAIMER:** The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

**SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER:** Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.