# AVR128DB28/32/48/64

## AVR® DB Family

## Introduction

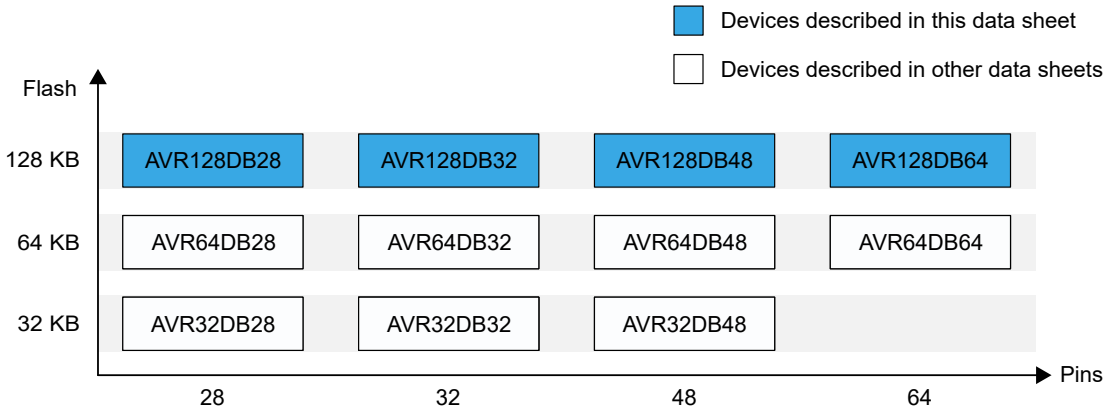
The AVR128DB28/32/48/64 microcontrollers of the AVR® DB family of microcontrollers are using the AVR® CPU with hardware multiplier running at clock speeds up to 24 MHz. They come with 128 KB of Flash, 16 KB of SRAM, and 512 bytes of EEPROM. The microcontrollers are available in 28-, 32-, 48- and 64- pin packages. The AVR® DB family uses the latest technologies from Microchip with a flexible and low-power architecture, including Event System, accurate analog subsystems, and advanced digital peripherals.

## AVR® DB Family Overview

The figure below shows the AVR DB devices, laying out pin count variants and memory sizes:

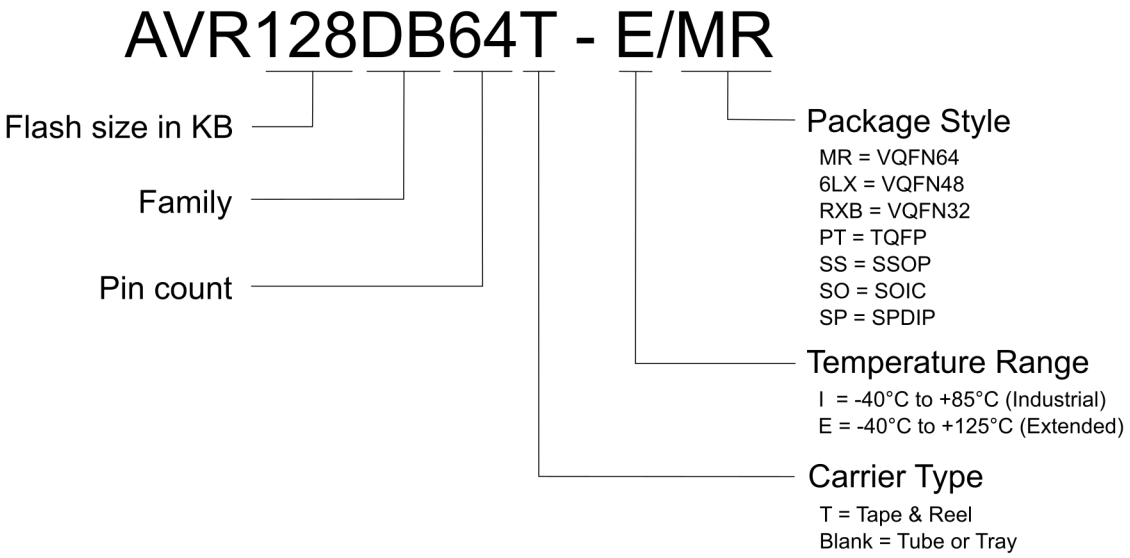
- Vertical migration is possible without code modification, as these devices are fully pin and feature compatible.
- Horizontal migration to the left reduces the pin count and therefore the available features.

**Figure 1. AVR® DB Family Overview**




The name of a device in the AVR DB family is decoded as follows:

**Figure 2. AVR® DB Device Designations**




## Memory Overview

The following table shows the memory overview of the entire family, but the further documentation describes only the AVR128DB28/32/48/64 devices.

**Table 1. Memory Overview**

| Devices | AVR32DB28<br>AVR32DB32<br>AVR32DB48 | AVR64DB28<br>AVR64DB32<br>AVR64DB48<br>AVR64DB64 | AVR128DB28<br>AVR128DB32<br>AVR128DB48<br>AVR128DB64 |
|---|---|---|---|
| Flash memory | 32 KB | 64 KB | 128 KB |
| SRAM | 4 KB | 8 KB | 16 KB |
| EEPROM | 512B | 512B | 512B |
| User row | 32B | 32B | 32B |

## Peripheral Overview

The following table shows the peripheral overview of the entire AVR DB family, but the further documentation describes only the AVR128DB28/32/48/64 devices.

**Table 2. Peripheral Overview**

| Feature | AVR32DB28<br>AVR64DB28<br>AVR128DB28 | AVR32DB32<br>AVR64DB32<br>AVR128DB32 | AVR32DB48<br>AVR64DB48<br>AVR128DB48 | AVR64DB64<br>AVR128DB64 |
|---|---|---|---|---|
| **Pins** | **28** | **32** | **48** | **64** |
| Max. frequency (MHz) | 24 | 24 | 24 | 24 |
| 16-bit Timer/Counter type A (TCA) | 1 | 1 | 2 | 2 |
| 16-bit Timer/Counter type B (TCB) | 3 | 3 | 4 | 5 |
| 12-bit Timer/Counter type D (TCD) | 1 | 1 | 1 | 1 |

.....continued

| Feature | AVR32DB28<br>AVR64DB28<br>AVR128DB28 | AVR32DB32<br>AVR64DB32<br>AVR128DB32 | AVR32DB48<br>AVR64DB48<br>AVR128DB48 | AVR64DB64<br>AVR128DB64 |
|---|---|---|---|---|
| **Pins** | **28** | **32** | **48** | **64** |
| Real-Time Counter (RTC) | 1 | 1 | 1 | 1 |
| USART | 3 | 3 | 5 | 6 |
| SPI | 2 | 2 | 2 | 2 |
| TWI/I$^2$C | 1[(1)] | 2[(1)] | 2[(1)] | 2[(1)] |
| 12-bit differential ADC (channels) | 1 (9) | 1 (13) | 1 (18) | 1 (22) |
| 10-bit DAC (outputs) | 1 (1) | 1 (1) | 1 (1) | 1 (1) |
| Analog Comparator (AC) | 3 | 3 | 3 | 3 |
| Zero-Cross Detector (ZCD) | 1 | 1 | 2 | 3 |
| Peripheral Touch Controller (PTC) | - | - | - | - |
| Op amp (OP) | 2 | 2 | 3 | 3 |
| Configurable Custom Logic Look-up Table (CCL LUT) | 4 | 4 | 6 | 6 |
| Watchdog Timer (WDT) | 1 | 1 | 1 | 1 |
| Event System channels (EVSYS) | 8 | 8 | 10 | 10 |
| General Purpose I/O[(2)] | 22/21[(2)] | 26/25[(2)] | 41/40[(2)] | 55/54[(2)] |
| PORT | PA[7:0], PC[3:0], PD[7:1], PF[6,1,0] | PA[7:0], PC[3:0], PD[7:1], PF[6:0] | PA[7:0], PB[5:0], PC[7:0], PD[7:0], PE[3:0], PF[6:0] | PA[7:0], PB[7:0], PC[7:0], PD[7:0], PE[7:0], PF[6:0], PG[7:0] |
| External Interrupts | 22 | 26 | 41 | 55 |
| CRCSCAN | 1 | 1 | 1 | 1 |
| Unified Program and Debug Interface (UPDI) | 1 | 1 | 1 | 1 |

**Notes:**

1. The TWI/I$^2$C can operate simultaneously as master and slave on different pins.
2. PF6/$\overline{\text{RESET}}$ pin is input only.

## Features

- AVR® CPU
  - Running at up to 24 MHz
  - Single-cycle I/O register access
  - Two-level interrupt controller
  - Two-cycle hardware multiplier
  - Supply voltage range: 1.8V to 5.5V
- Memories
  - 128 KB In-System self-programmable Flash memory
  - 512B EEPROM
  - 16 KB SRAM
  - 32B of user row in nonvolatile memory that can keep data during chip-erase and be programmed while the device is locked
  - Write/erase endurance
    - Flash: 10,000 cycles
    - EEPROM: 100,000 cycles
  - Data retention: 40 Years at 55°C
- System
  - Power-on Reset (POR) circuit
  - Brown-out Detector (BOD) with user-programmable levels
  - Voltage Level Monitor (VLM) with interrupt at a programmable level above the BOD level
  - Clock failure detection
  - Clock options
    - High-precision internal oscillator with selectable frequency up to 24 MHz (OSCHF)
      - Auto-tuning for improved internal oscillator accuracy
    - Internal PLL up to 48 MHz for high-frequency operation of Timer/Counter type D (PLL)
    - Internal ultra-low power 32.768 kHz oscillator (OSC32K)
    - External 32.768 kHz crystal oscillator (XOSC32K)
    - External clock input
    - External high-frequency crystal oscillator (XOSCHF) with clock failure detection
  - Single pin Unified Program and Debug Interface (UPDI)
  - Three sleep modes
    - Idle with all peripherals running for immediate wake-up
    - Standby
      - Configurable operation of selected peripherals
      - SleepWalking peripherals
    - Power-Down with full data retention
- Peripherals
  - Up to two 16-bit Timer/Counters type A (TCA) with three compare channels for PWM and waveform generation
  - Up to five 16-bit Timer/Counters type B (TCB) with input capture for capture and signal measurements
  - One 12-bit PWM Timer/Counter type D (TCD) optimized for power control
  - One 16-bit Real-Time Counter (RTC) that can run from external crystal or internal oscillator
  - Up to six USARTs
    - Operation modes: RS-485, LIN slave, master SPI, and IrDA
    - Fractional baud rate generator, auto-baud, and start-of-frame detection
  - Two SPIs with master/slave operation modes
  - Up to two Two-Wire Interface (TWI) with dual address match
    - Independent master and slave operation Dual mode)

    - Phillips $I^2C$ compatible
    - Standard mode (Sm, 100 kHz)
    - Fast mode (Fm, 400 kHz)
    - Fast mode plus (Fm+, 1 MHz)[(1)]
  - Event System for CPU-independent and predictable inter-peripherals signaling
  - Configurable Custom Logic (CCL) with up to six programmable Look-up Tables (LUTs)
  - One 12-bit 130 ksps differential Analog-to-Digital Converter (ADC)
  - Three Analog Comparators (ACs) with window compare functions
  - One 10-bit Digital-to-Analog Converter (DAC)
  - Up to three Zero-Cross Detectors (ZCDs)
  - Analog Signal Conditioning (OPAMP) peripheral with up to three op amps, each with an internal resistor ladder that allows for many useful configurations with no external components
  - Multiple voltage references (VREF)
    - 1.024V
    - 2.048V
    - 2.500V
    - 4.096V
    - External Voltage Reference (VREFA)
    - Supply Voltage ($V_{DD}$)
  - Automated Cyclic Redundancy Check (CRC) Flash program memory scan
  - Watchdog Timer (WDT) with Window mode, and separate on-chip oscillator
  - External interrupt on all general purpose pins
- I/O and Packages
  - Multi-Voltage I/O (MVIO) on I/O port C
  - Selectable input voltage threshold
  - Up to 55/54 programmable I/O pins
  - 28-pin SSOP, SOIC and SPDIP
  - 32-pin VQFN 5x5 mm and TQFP 7x7 mm
  - 48-pin VQFN 5x5 mm and TQFP 7x7 mm
  - 64-pin VQFN 9x9 mm and TQFP 10x10 mm
- Temperature Ranges
  - Industrial: -40°C to 85°C
  - Extended: -40°C to 125°C

**Note:**

1. $I^2C$ Fm+ is only supported for supply voltage $V_{DD}$ above 2.7 VDC.

# Table of Contents

Introduction....1

AVR® DB Family Overview.... 1

1. Memory Overview.... 2
2. Peripheral Overview....2

Features....4

1. Block Diagram....13

2. Pinout....14

2.1. 28-pin SSOP, SOIC and SPDIP.... 14
2.2. 32-pin VQFN and TQFP....15
2.3. 48-pin VQFN and TQFP....16
2.4. 64-pin VQFN and TQFP....17

3. I/O Multiplexing and Considerations.... 18

3.1. I/O Multiplexing....18

4. Hardware Guidelines....21

4.1. General Guidelines....21
4.2. Connection for Power Supply....21
4.3. Connection for $\overline{\text{RESET}}$....23
4.4. Connection for UPDI Programming....23
4.5. Connecting External Crystal Oscillators....24
4.6. Connection for External Voltage Reference.... 25

5. Power Supply....27

5.1. Power Domains....27
5.2. Voltage Regulator....27
5.3. Power-Up.... 28

6. Conventions.... 29

6.1. Numerical Notation....29
6.2. Memory Size and Type....29
6.3. Frequency and Time....29
6.4. Registers and Bits.... 30
6.5. ADC Parameter Definitions.... 31

7. AVR® CPU.... 34

7.1. Features.... 34
7.2. Overview.... 34
7.3. Architecture.... 34
7.4. Functional Description....36
7.5. Register Summary....40
7.6. Register Description....40

8. Memories.... 45

8.1. Overview ... 45
8.2. Memory Map ... 45
8.3. In-System Reprogrammable Flash Program Memory ... 45
8.4. SRAM Data Memory ... 46
8.5. EEPROM Data Memory ... 46
8.6. SIGROW - Signature Row ... 47
8.7. USERROW - User Row ... 51
8.8. FUSE - Configuration and User Fuses ... 51
8.9. LOCK - Memory Sections Access Protection ... 59
8.10. I/O Memory ... 62

9. GPR - General Purpose Registers ... 65
9.1. Register Summary ... 66
9.2. Register Description ... 66

10. Peripherals and Architecture ... 68
10.1. Peripheral Address Map ... 68
10.2. Interrupt Vector Mapping ... 70
10.3. SYSCFG - System Configuration ... 72

11. NVMCTRL - Nonvolatile Memory Controller ... 75
11.1. Features ... 75
11.2. Overview ... 75
11.3. Functional Description ... 76
11.4. Register Summary ... 84
11.5. Register Description ... 84

12. CLKCTRL - Clock Controller ... 92
12.1. Features ... 92
12.2. Overview ... 92
12.3. Functional Description ... 94
12.4. Register Summary ... 100
12.5. Register Description ... 100

13. SLPCTRL - Sleep Controller ... 115
13.1. Features ... 115
13.2. Overview ... 115
13.3. Functional Description ... 115
13.4. Register Summary ... 120
13.5. Register Description ... 120

14. RSTCTRL - Reset Controller ... 123
14.1. Features ... 123
14.2. Overview ... 123
14.3. Functional Description ... 124
14.4. Register Summary ... 127
14.5. Register Description ... 127

15. CPUINT - CPU Interrupt Controller ... 130

15.1. Features ..... 130
15.2. Overview ..... 130
15.3. Functional Description ..... 131
15.4. Register Summary ..... 136
15.5. Register Description ..... 136

16. EVSYS - Event System ..... 141
16.1. Features ..... 141
16.2. Overview ..... 141
16.3. Functional Description ..... 142
16.4. Register Summary ..... 148
16.5. Register Description ..... 148

17. PORTMUX - Port Multiplexer ..... 155
17.1. Overview ..... 155
17.2. Register Summary ..... 156
17.3. Register Description ..... 156

18. PORT - I/O Pin Configuration ..... 170
18.1. Features ..... 170
18.2. Overview ..... 170
18.3. Functional Description ..... 172
18.4. Register Summary - PORTx ..... 176
18.5. Register Description - PORTx ..... 176
18.6. Register Summary - VPORTx ..... 193
18.7. Register Description - VPORTx ..... 193

19. MVIO - Multi-Voltage I/O ..... 198
19.1. Features ..... 198
19.2. Overview ..... 198
19.3. Functional Description ..... 199
19.4. Register Summary ..... 202
19.5. Register Description ..... 202

20. BOD - Brown-out Detector ..... 206
20.1. Features ..... 206
20.2. Overview ..... 206
20.3. Functional Description ..... 207
20.4. Register Summary ..... 209
20.5. Register Description ..... 209

21. VREF - Voltage Reference ..... 216
21.1. Features ..... 216
21.2. Overview ..... 216
21.3. Functional Description ..... 216
21.4. Register Summary ..... 217
21.5. Register Description ..... 217

22. WDT - Watchdog Timer ..... 221

22.1. Features....221
22.2. Overview....221
22.3. Functional Description....221
22.4. Register Summary....225
22.5. Register Description....225

23. TCA - 16-bit Timer/Counter Type A....229
23.1. Features....229
23.2. Overview....229
23.3. Functional Description....232
23.4. Register Summary - Normal Mode....242
23.5. Register Description - Normal Mode....242
23.6. Register Summary - Split Mode....261
23.7. Register Description - Split Mode....261

24. TCB - 16-bit Timer/Counter Type B....277
24.1. Features....277
24.2. Overview....277
24.3. Functional Description....279
24.4. Register Summary....289
24.5. Register Description....289

25. TCD - 12-Bit Timer/Counter Type D....300
25.1. Features....300
25.2. Overview....300
25.3. Functional Description....302
25.4. Register Summary....325
25.5. Register Description....325

26. RTC - Real-Time Counter....350
26.1. Features....350
26.2. Overview....350
26.3. Clocks....351
26.4. RTC Functional Description....351
26.5. PIT Functional Description....352
26.6. Crystal Error Correction....353
26.7. Events....353
26.8. Interrupts....354
26.9. Sleep Mode Operation....355
26.10. Synchronization....355
26.11. Debug Operation....355
26.12. Register Summary....356
26.13. Register Description....356

27. USART - Universal Synchronous and Asynchronous Receiver and Transmitter....373
27.1. Features....373
27.2. Overview....373
27.3. Functional Description....374
27.4. Register Summary....389

27.5. Register Description....389

28. SPI - Serial Peripheral Interface....407
28.1. Features....407
28.2. Overview....407
28.3. Functional Description....408
28.4. Register Summary....415
28.5. Register Description....415

29. TWI - Two-Wire Interface....422
29.1. Features....422
29.2. Overview....422
29.3. Functional Description....423
29.4. Register Summary....434
29.5. Register Description....434

30. CRCSCAN - Cyclic Redundancy Check Memory Scan....452
30.1. Features....452
30.2. Overview....452
30.3. Functional Description....452
30.4. Register Summary....455
30.5. Register Description....455

31. CCL – Configurable Custom Logic....459
31.1. Features....459
31.2. Overview....459
31.3. Functional Description....461
31.4. Register Summary....469
31.5. Register Description....469

32. AC - Analog Comparator....482
32.1. Features....482
32.2. Overview....482
32.3. Functional Description....483
32.4. Register Summary....487
32.5. Register Description....487

33. ADC - Analog-to-Digital Converter....494
33.1. Features....494
33.2. Overview....494
33.3. Functional Description....495
33.4. Register Summary....506
33.5. Register Description....506

34. DAC - Digital-to-Analog Converter....524
34.1. Features....524
34.2. Overview....524
34.3. Functional Description....524
34.4. Register Summary....526

34.5. Register Description....526

35. OPAMP - Analog Signal Conditioning....529
35.1. Features....529
35.2. Overview....529
35.3. Functional Description....530
35.4. Register Summary....542
35.5. Register Description....542

36. ZCD - Zero-Cross Detector....553
36.1. Features....553
36.2. Overview....553
36.3. Functional Description....554
36.4. Register Summary....561
36.5. Register Description....561

37. UPDI - Unified Program and Debug Interface....565
37.1. Features....565
37.2. Overview....565
37.3. Functional Description....567
37.4. Register Summary....586
37.5. Register Description....586

38. Instruction Set Summary....597

39. Electrical Characteristics....598
39.1. Disclaimer....598
39.2. Absolute Maximum Ratings....598
39.3. Standard Operating Conditions....598
39.4. DC Characteristics....599
39.5. AC Characteristics....606

40. Typical Characteristics....619
40.1. OPAMP....619

41. Ordering Information....628

42. Package Drawings....630
42.1. Online Package Drawings....630
42.2. 28-Pin SPDIP....631
42.3. 28-Pin SOIC....632
42.4. 28-Pin SSOP....635
42.5. 32-Pin VQFN....638
42.6. 32-Pin TQFP....641
42.7. 48-Pin VQFN....644
42.8. 48-Pin TQFP....647
42.9. 64-Pin VQFN....650
42.10. 64-Pin TQFP....653

43. Data Sheet Revision History....656

43.1. Rev. A - 08/2020....656

The Microchip Website....657

Product Change Notification Service....657

Customer Support....657

Product Identification System....658

Microchip Devices Code Protection Feature....658

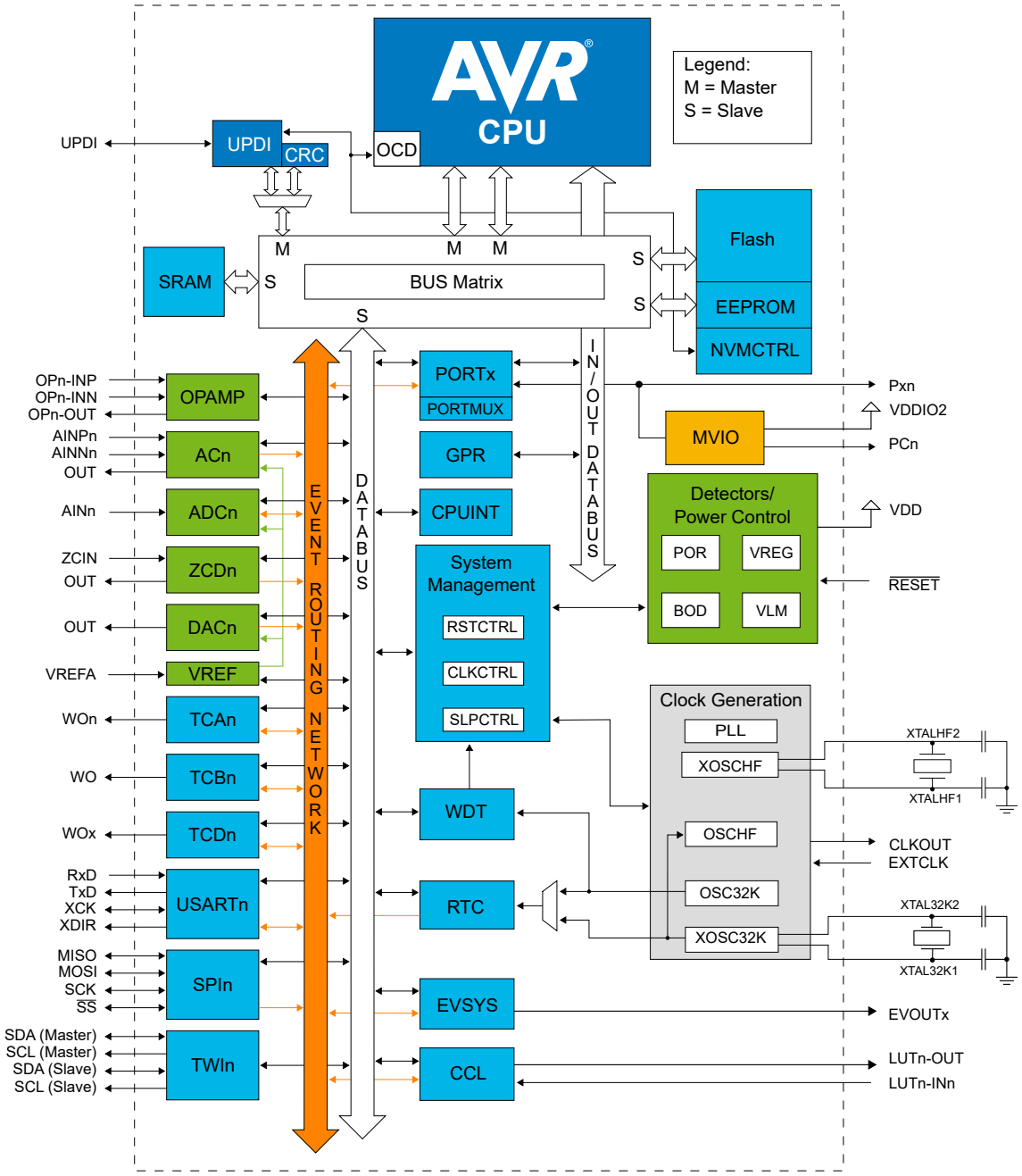
Legal Notice....658

Trademarks....659

Quality Management System....659

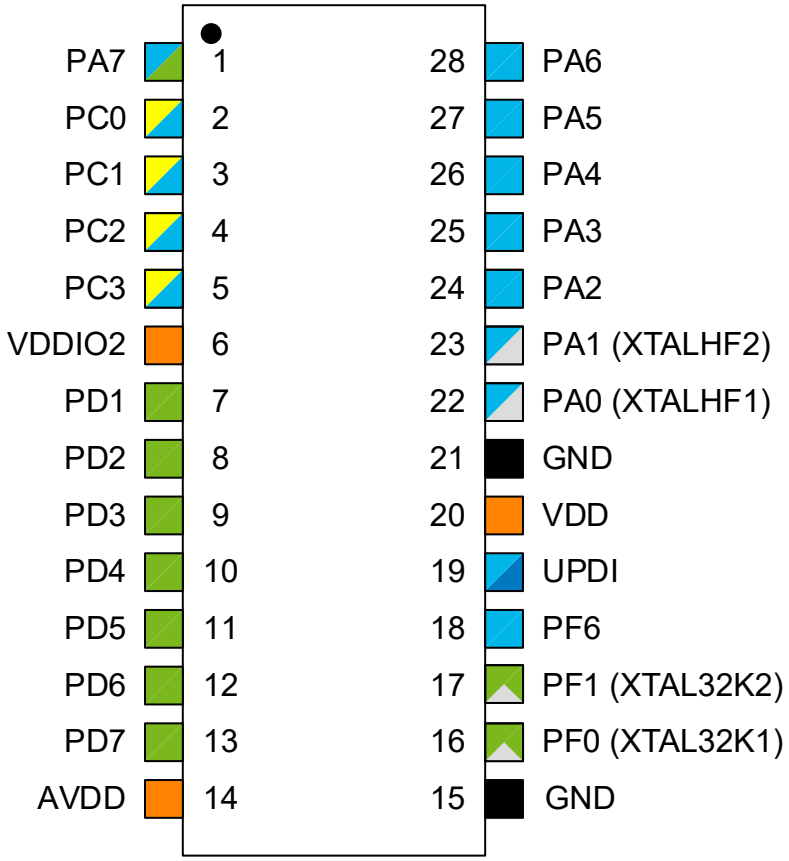
Worldwide Sales and Service....660

# 1. Block Diagram



# 2. Pinout

## 2.1 28-pin SSOP, SOIC and SPDIP




**Note:** For the AVR® DBFamily of devices, AVDD is internally connected to VDD (not separate power domains).

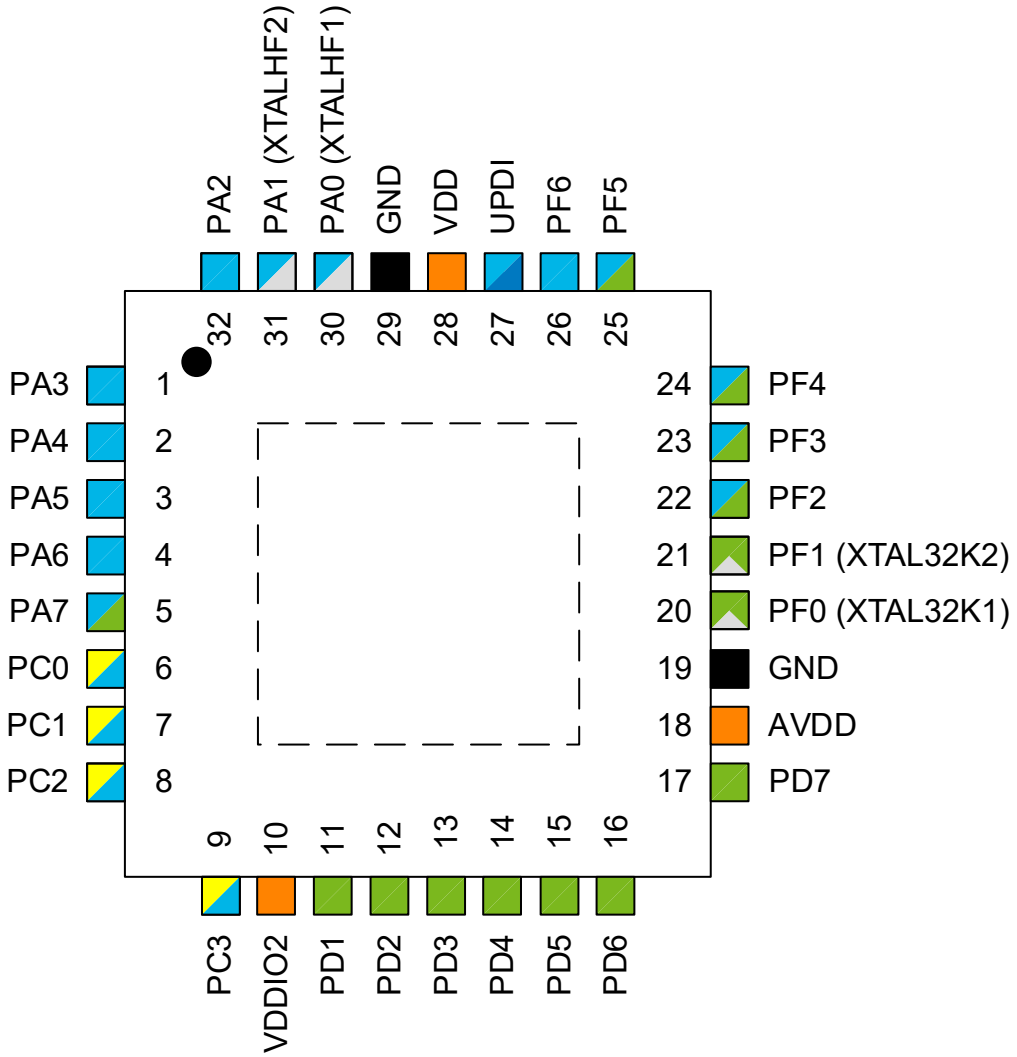
**Power**

- Power Supply
- Ground
- Pin on VDD Power Domain
- Pin on AVDD Power Domain
- Pin on VDDIO2 Power Domain

**Functionality**

- Programming/Debug
- Clock/Crystal
- Digital Function Only
- Analog Function

## 2.2 32-pin VQFN and TQFP




**Note:** For the AVR® DBFamily of devices, AVDD is internally connected to VDD (not separate power domains).

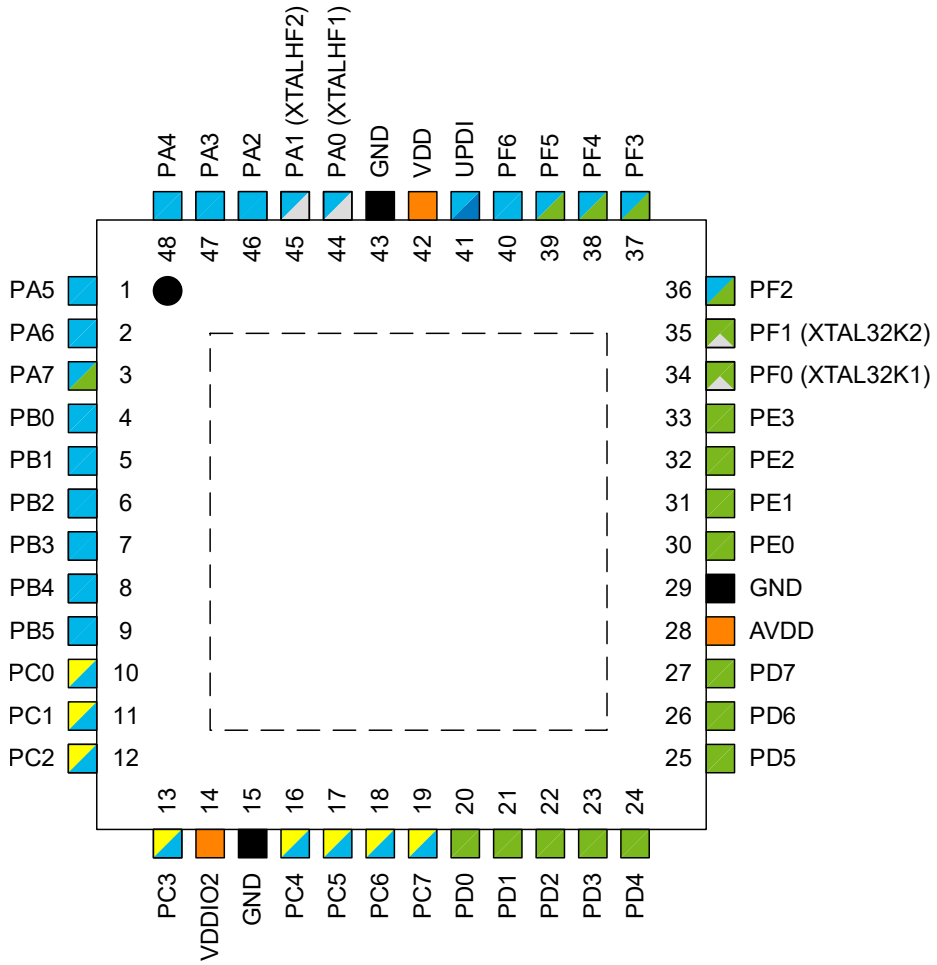
**Power**

- Power Supply
- Ground
- Pin on VDD Power Domain
- Pin on AVDD Power Domain
- Pin on VDDIO2 Power Domain

**Functionality**

- Programming/Debug
- Clock/Crystal
- Digital Function Only
- Analog Function

## 2.3 48-pin VQFN and TQFP




**Note:** For the AVR® DBFamily of devices, AVDD is internally connected to VDD (not separate power domains).

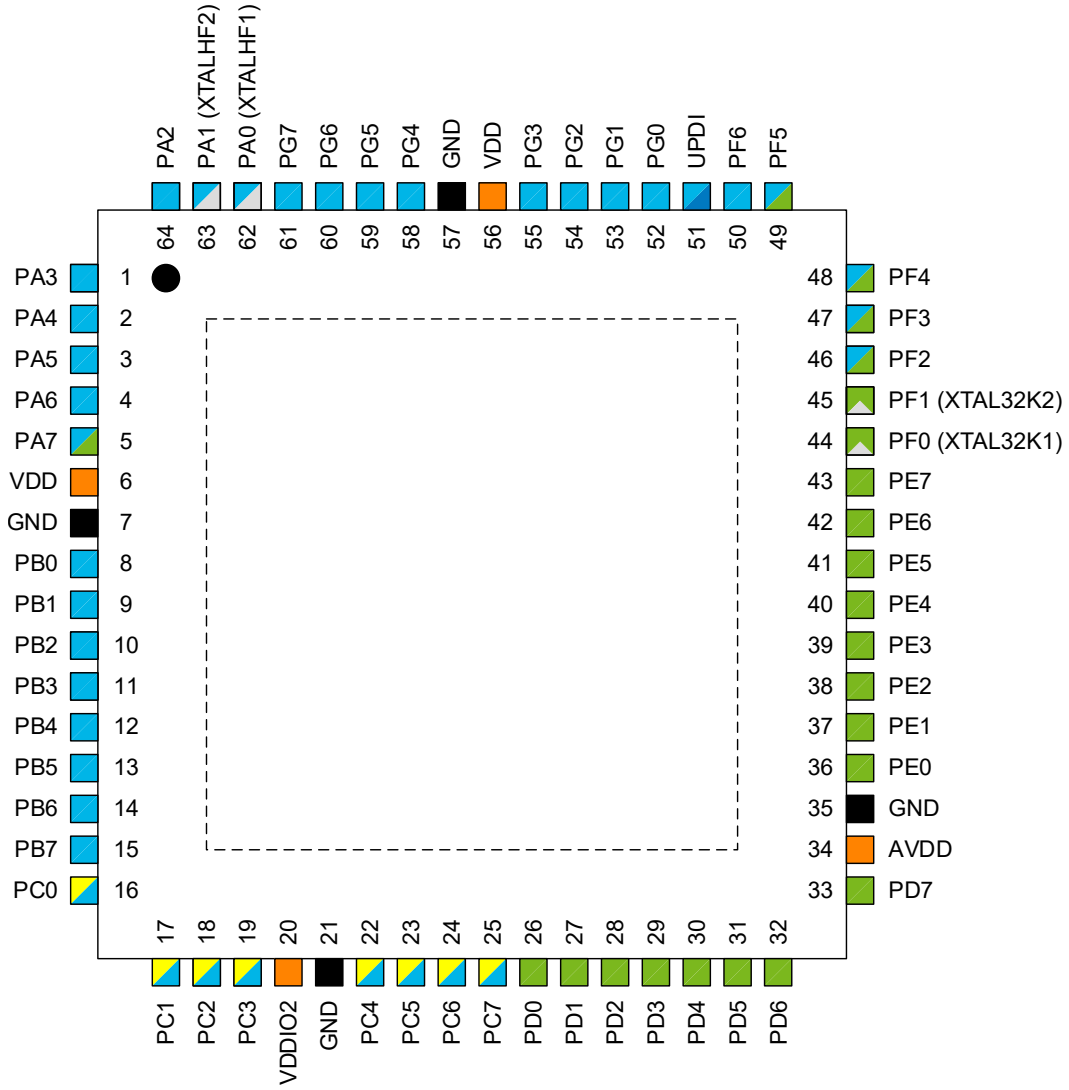
**Power**

- Power Supply
- Ground
- Pin on VDD Power Domain
- Pin on AVDD Power Domain
- Pin on VDDIO2 Power Domain

**Functionality**

- Programming/Debug
- Clock/Crystal
- Digital Function Only
- Analog Function

## 2.4 64-pin VQFN and TQFP




**Note:** For the AVR® DBFamily of devices, AVDD is internally connected to VDD (not separate power domains).

**Power**

- Power Supply
- Ground
- Pin on VDD Power Domain
- Pin on AVDD Power Domain
- Pin on VDDIO2 Power Domain

**Functionality**

- Programming/Debug
- Clock/Crystal
- Digital Function Only
- Analog Function

# 3. I/O Multiplexing and Considerations

## 3.1 I/O Multiplexing

| VQFN64/ TQFP64 | VQFN48/ TQFP48 | VQFN32/ TQFP32 | SOIC28/ SSOP28/SPDIP28 | Pin name(1,2) | Special | ADC0 | ACn | DAC0 | OPAMP | ZCDn | USARTn | SPIn | TWIn(4) | TCAn | TCBn | TCD0 | EVSYS | CCL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 62 | 44 | 30 | 22 | PA0 | XTALHF1<br>EXTCLK | | | | | | 0, TxD | | | 0, WO0 | | | | LUT0, IN0 |
| 63 | 45 | 31 | 23 | PA1 | XTALHF2 | | | | | | 0, RxD | | | 0, WO1 | | | | LUT0, IN1 |
| 64 | 46 | 32 | 24 | PA2 | TWI<br>Fm+ | | | | | | 0, XCK | | 0, SDA(MS) | 0, WO2 | 0, WO | | EVOUTA | LUT0, IN2 |
| 1 | 47 | 1 | 25 | PA3 | TWI<br>Fm+ | | | | | | 0, XDIR | | 0, SCL(MS) | 0, WO3 | 1, WO | | | LUT0, OUT |
| 2 | 48 | 2 | 26 | PA4 | | | | | | | 0, TxD(3) | 0, MOSI | | 0, WO4 | | WOA | | |
| 3 | 1 | 3 | 27 | PA5 | | | | | | | 0, RxD(3) | 0, MISO | | 0, WO5 | | WOB | | |
| 4 | 2 | 4 | 28 | PA6 | | | | | | | 0, XCK(3) | 0, SCK | | | | WOC | | LUT0, OUT(3) |
| 5 | 3 | 5 | 1 | PA7 | CLKOUT | | 0, OUT<br>1, OUT<br>2, OUT | | | 0, OUT<br>1, OUT<br>2, OUT | 0, XDIR(3) | 0, $\overline{SS}$ | | | | WOD | EVOUTA(3) | |
| 6 | | | | VDD | | | | | | | | | | | | | | |
| 7 | | | | GND | | | | | | | | | | | | | | |
| 8 | 4 | | | PB0 | | | | | | | 3, TxD | | | 0, WO0(3)<br>1, WO0 | | | | LUT4, IN0 |
| 9 | 5 | | | PB1 | | | | | | | 3, RxD | | | 0, WO1(3)<br>1, WO1 | | | | LUT4, IN1 |
| 10 | 6 | | | PB2 | TWI | | | | | | 3, XCK | | 1, SDA(MS)(3) | 0, WO2(3)<br>1, WO2 | | | EVOUTB | LUT4, IN2 |
| 11 | 7 | | | PB3 | TWI | | | | | | 3, XDIR | | 1, SCL(MS)(3) | 0, WO3(3)<br>1, WO3 | | | | LUT4, OUT |
| 12 | 8 | | | PB4 | | | | | | | 3, TxD(3) | 1, MOSI(3) | | 0, WO4(3)<br>1, WO4 | 2, WO(3) | WOA(3) | | |
| 13 | 9 | | | PB5 | | | | | | | 3, RxD(3) | 1, MISO(3) | | 0, WO5(3)<br>1, WO5 | 3, WO | WOB(3) | | |
| 14 | | | | PB6 | | | | | | | 3, XCK(3) | 1, SCK(3) | 1, SDA(S)(3) | | | WOC(3) | | LUT4, OUT(3) |
| 15 | | | | PB7 | | | | | | | 3, XDIR(3) | 1, $\overline{SS}$(3) | 1, SCL(S)(3) | | | WOD(3) | EVOUTB(3) | |
| 16 | 10 | 6 | 2 | PC0 | | | | | | | 1, TxD | 1, MOSI | | 0, WO0(3) | 2, WO | | | LUT1, IN0 |
| 17 | 11 | 7 | 3 | PC1 | | | | | | | 1, RxD | 1, MISO | | 0, WO1(3) | 3, WO(3) | | | LUT1, IN1 |
| 18 | 12 | 8 | 4 | PC2 | TWI<br>Fm+ | | | | | | 1, XCK | 1, SCK | 0, SDA(MS)(3) | 0, WO2(3) | | | EVOUTC | LUT1, IN2 |
| 19 | 13 | 9 | 5 | PC3 | TWI<br>Fm+ | | | | | | 1, XDIR | 1, $\overline{SS}$ | 0, SCL(MS)(3) | 0, WO3(3) | | | | LUT1, OUT |
| 20 | 14 | 10 | 6 | VDDIO2 | | | | | | | | | | | | | | |
| 21 | 15 | | | GND | | | | | | | | | | | | | | |
| 22 | 16 | | | PC4 | | | | | | | 1, TxD(3) | 1, MOSI(3) | | 0, WO4(3)<br>1, WO0(3) | | | | |
| 23 | 17 | | | PC5 | | | | | | | 1, RxD(3) | 1, MISO(3) | | 0, WO5(3)<br>1, WO1(3) | | | | |
| 24 | 18 | | | PC6 | | | 0, OUT(3)<br>1, OUT(3)<br>2, OUT(3) | | | | 1, XCK(3) | 1, SCK(3) | 0, SDA(S)(3) | 1, WO2(3) | | | | LUT1, OUT(3) |
| 25 | 19 | | | PC7 | | | | | | 0, OUT(3)<br>1, OUT(3)<br>2, OUT(3) | 1, XDIR(3) | 1, $\overline{SS}$(3) | 0, SCL(S)(3) | | | | EVOUTC(3) | |
| 26 | 20 | | | PD0 | | AIN0 | 0, AINN1<br>1, AINN1<br>2, AINN1 | | | | | | | 0, WO0(3) | | | | LUT2, IN0 |

.....continued

| VQFN64/ TQFP64 | VQFN48/ TQFP48 | VQFN32/ TQFP32 | SOIC28/ SSOP28/SPDIP28 | Pin name(1,2) | Special | ADC0 | ACn | DAC0 | OPAMP | ZCDn | USARTn | SPIn | TWIn(4) | TCAn | TCBn | TCD0 | EVSYS | CCL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 27 | 21 | 11 | 7 | PD1 | | AIN1 | | | OP0, INP | 0, ZCIN | | | | 0, WO1(3) | | | | LUT2, IN1 |
| 28 | 22 | 12 | 8 | PD2 | | AIN2 | 0, AINP0<br>1, AINP0<br>2, AINP0 | | OP0, OUT | | | | | 0, WO2(3) | | | EVOUTD | LUT2, IN2 |
| 29 | 23 | 13 | 9 | PD3 | | AIN3 | 0, AINN0<br>1, AINP1 | | OP0, INN | | | | | 0, WO3(3) | | | | LUT2, OUT |
| 30 | 24 | 14 | 10 | PD4 | | AIN4 | 1, AINP2<br>2, AINP1 | | OP1, INP | | | | | 0, WO4(3) | | | | |
| 31 | 25 | 15 | 11 | PD5 | | AIN5 | 1, AINN0 | | OP1, OUT | | | | | 0, WO5(3) | | | | |
| 32 | 26 | 16 | 12 | PD6 | | AIN6 | 0, AINP3<br>1, AINP3<br>2, AINP3 | OUT | | | | | | | | | | LUT2, OUT(3) |
| 33 | 27 | 17 | 13 | PD7 | VREFA | AIN7 | 0, AINN2<br>1, AINN2<br>2, AINN0/AINN2 | | OP1, INN | | | | | | | | EVOUTD(3) | |
| 34 | 28 | 18 | 14 | AVDD | | | | | | | | | | | | | | |
| 35 | 29 | 19 | 15 | AGND | | | | | | | | | | | | | | |
| 36 | 30 | | | PE0 | | AIN8 | 0, AINP1 | | | | 4, TxD | 0, MOSI(3) | | 0, WO0(3) | | | | |
| 37 | 31 | | | PE1 | | AIN9 | 2, AINP2 | | OP2, INP | | 4, RxD | 0, MISO(3) | | 0, WO1(3) | | | | |
| 38 | 32 | | | PE2 | | AIN10 | 0, AINP2 | | OP2, OUT | | 4, XCK | 0, SCK(3) | | 0, WO2(3) | | | EVOUTE | |
| 39 | 33 | | | PE3 | | AIN11 | | | OP2, INN | 1, ZCIN | 4, XDIR | 0, $\overline{SS}$(3) | | 0, WO3(3) | | | | |
| 40 | | | | PE4 | | AIN12 | | | | | 4, TxD(3) | | | 0, WO4(3)<br>1, WO0(3) | | | | |
| 41 | | | | PE5 | | AIN13 | | | | | 4, RxD(3) | | | 0, WO5(3)<br>1, WO1(3) | | | | |
| 42 | | | | PE6 | | AIN14 | | | | | 4, XCK(3) | | | 1, WO2(3) | | | | |
| 43 | | | | PE7 | | AIN15 | | | | 2, ZCIN | 4, XDIR(3) | | | | | | EVOUTE(3) | |
| 44 | 34 | 20 | 16 | PF0 | XTAL32K1 | AIN16(5) | | | | | 2, TxD | | | 0, WO0(3) | | WOA(3) | | LUT3, IN0 |
| 45 | 35 | 21 | 17 | PF1 | XTAL32K2 | AIN17(5) | | | | | 2, RxD | | | 0, WO1(3) | | WOB(3) | | LUT3, IN1 |
| 46 | 36 | 22 | | PF2 | TWI Fm+ | AIN18(5) | | | | | 2, XCK | | 1, SDA(MS) | 0, WO2(3) | | WOC(3) | EVOUTF | LUT3, IN2 |
| 47 | 37 | 23 | | PF3 | TWI Fm+ | AIN19(5) | | | | | 2, XDIR | | 1, SCL(MS) | 0, WO3(3) | | WOD(3) | | LUT3, OUT |
| 48 | 38 | 24 | | PF4 | | AIN20(5) | | | | | 2, TxD(3) | | | 0, WO4(3) | 0, WO(3) | | | |
| 49 | 39 | 25 | | PF5 | | AIN21(5) | | | | | 2, RxD(3) | | | 0, WO5(3) | 1, WO(3) | | | |
| 50 | 40 | 26 | 18 | PF6(6) | $\overline{RESET}$ | | | | | | | | | | | | | |
| 51 | 41 | 27 | 19 | UPDI | UPDI | | | | | | | | | | | | | |
| 52 | | | | PG0 | | | | | | | 5, TxD | | | 0, WO0(3)<br>1, WO0(3) | | | | LUT5, IN0 |
| 53 | | | | PG1 | | | | | | | 5, RxD | | | 0, WO1(3)<br>1, WO1(3) | | | | LUT5, IN1 |
| 54 | | | | PG2 | | | | | | | 5, XCK | | | 0, WO2(3)<br>1, WO2(3) | | | EVOUTG | LUT5, IN2 |
| 55 | | | | PG3 | | | | | | | 5, XDIR | | | 0, WO3(3)<br>1, WO3(3) | 4, WO | | | LUT5, OUT |
| 56 | 42 | 28 | 20 | VDD | | | | | | | | | | | | | | |
| 57 | 43 | 29 | 21 | GND | | | | | | | | | | | | | | |
| 58 | | | | PG4 | | | | | | | 5, TxD(3) | 0, MOSI(3) | | 0, WO4(3)<br>1, WO4(3) | | WOA(3) | | |
| 59 | | | | PG5 | | | | | | | 5, RxD(3) | 0, MISO(3) | | 0, WO5(3)<br>1, WO5(3) | | WOB(3) | | |
| 60 | | | | PG6 | | | | | | | 5, XCK(3) | 0, SCK(3) | | | | WOC(3) | | LUT5, OUT(3) |
| 61 | | | | PG7 | | | | | | | 5, XDIR(3) | 0, $\overline{SS}$(3) | | | | WOD(3) | EVOUTG(3) | |

**Notes:**

1. Pin names are of type P*xn*, with *x* being the PORT instance (A, B, C, ...) and *n* the pin number. Notation for signals is PORT*x*_PIN*n*. All pins can be used as event inputs.
2. All pins can be used for external interrupt, where pins P*x*2 and P*x*6 of each port have full asynchronous detection.
3. Alternative pin positions.
4. TWI pins are marked *MS* if they can be used as TWI Master or Slave pins, and *S* if they can only be used as TWI Slave pins.
5. AIN16 - AIN21 cannot be used as a negative ADC input for differential measurements.
6. Input only.

# 4. Hardware Guidelines

This section contains guidelines for designing or reviewing electrical schematics using AVR 8-bit microcontrollers. The information presented here is a brief overview of the most common topics. More detailed information can be found in application notes, listed in this section where applicable.

This section covers the following topics:

- General guidelines
- Connection for power supply
- Connection for $\overline{\text{RESET}}$
- Connection for UPDI (Unified Program and Debug Interface)
- Connection for external crystal oscillators
- Connection for VREF (external voltage reference)

## 4.1 General Guidelines

Unused pins must be soldered to their respective soldering pads. The soldering pads must not be connected to the circuit.

The PORT pins are in their default state after Reset. Follow the recommendations in the *PORT* section to reduce power consumption.

All values are given as typical values and serve only as a starting point for circuit design.

Refer to the following application notes for further information:

- *AVR040 - EMC Design Considerations*
- *AVR042 - AVR Hardware Design Considerations*

### 4.1.1 Special Consideration for Packages with Center Pad

Flat packages often come with an exposed pad located on the bottom, often referred to as the center pad or the thermal pad. This pad is not electrically connected to the internal circuit of the chip, but it is mechanically bonded to the internal substrate and serves as a thermal heat sink as well as providing added mechanical stability. This pad must be connected to GND since the ground plane is the best heat sink (largest copper area) of the printed circuit board (PCB).

## 4.2 Connection for Power Supply

The basics and details regarding the design of the power supply itself lie beyond the scope of these guidelines. For more detailed information about this subject, see the application notes mentioned at the beginning of this section.

A decoupling capacitor must be placed close to the microcontroller for each supply pin pair (VDD, AVDD, or other power supply pin and its corresponding GND pin). If the decoupling capacitor is placed too far from the microcontroller, a high-current loop might form that will result in increased noise and increased radiated emission.

Each supply pin pair (power input pin and ground pin) must have separate decoupling capacitors.

It is recommended to place the decoupling capacitor on the same side of the PCB as the microcontroller. If space does not allow it, the decoupling capacitor may be placed on the other side through a via, but make sure the distance to the supply pin is kept as short as possible.

If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the decoupling capacitor described above. Place this second capacitor next to the primary decoupling capacitor.

On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

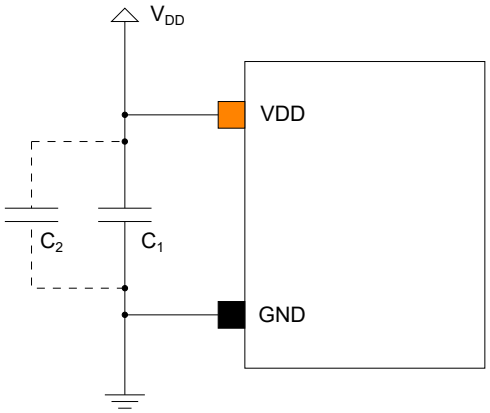
As mentioned at the beginning of this section, all values used in examples are typical values. The actual design may require other values.

### 4.2.1 Digital Power Supply

For larger pin count package types, there are several VDD and corresponding GND pins. All the VDD pins in the microcontroller are internally connected. The same voltage must be applied to each of the VDD pins.

The following figure shows the recommendation for connecting a power supply to the VDD pin(s) of the device.

**Figure 4-1. Recommended VDD Connection Circuit Schematic**



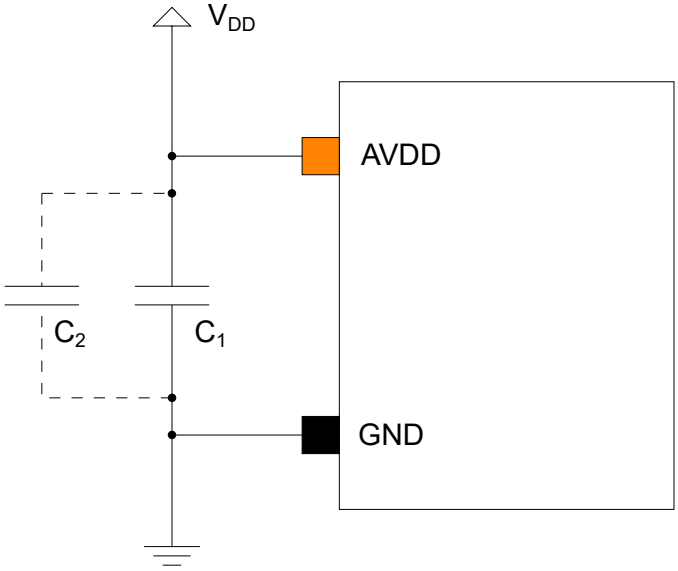

**Typical values (recommended):**
$C_1$: 1 µF (primary decoupling capacitor)
$C_2$: 10-100 nF (HF decoupling capacitor)

### 4.2.2 Analog Power Supply

These devices have a separate analog supply voltage pin, AVDD. This separate voltage supply pin is provided to make the analog circuits less exposed to the digital noise originating from the switching of the digital circuits.

The following figure shows the recommendation for connecting a power supply to the AVDD pin of the device.

**Figure 4-2. Recommended AVDD Connection Circuit Schematic**



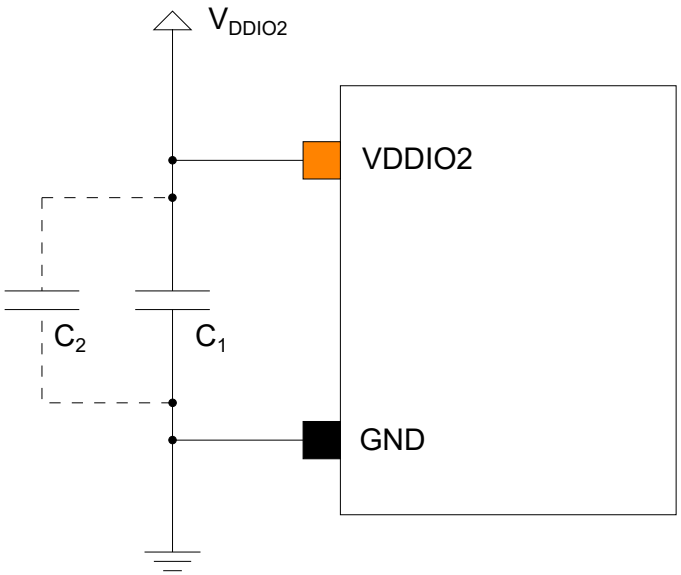

**Typical values (recommended):**
$C_1$: 1 µF (primary decoupling capacitor)
$C_2$: 10-100 nF (HF decoupling capacitor)

### 4.2.3 Multi-Voltage I/O

This additional Multi-Voltage I/O (MVIO) power supply input pin and corresponding grounding pin must be treated the same way as any other power supply pin pair: By connecting a separate decoupling capacitor at the shortest possible trace distance from pins. If there is more than one MVIO power supply pin, each supply pin and its corresponding ground pin must have a decoupling capacitor.

The following figure shows the recommendation for connecting a power supply to the VDDIO2 pin(s) of the device.

**Figure 4-3. Recommended VDDIO2 Connection Circuit Schematic**




**Typical values (recommended):**
$C_1$: 1 µF (primary decoupling capacitor)
$C_2$: 10-100 nF (HF decoupling capacitor)

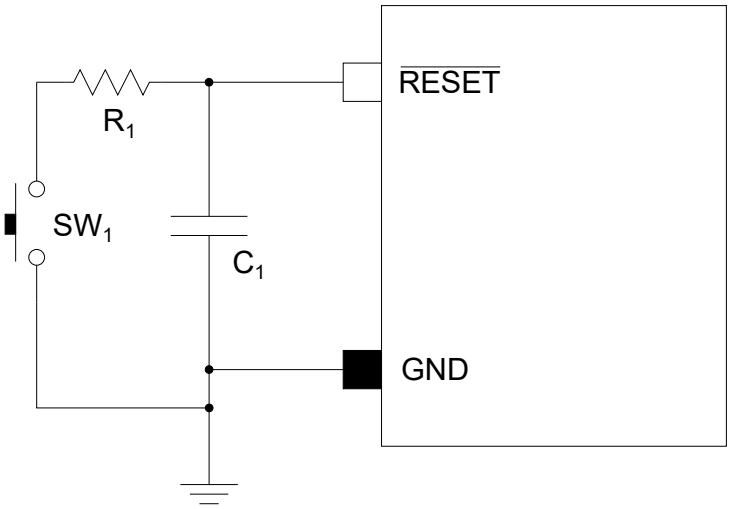
## 4.3 Connection for $\overline{\text{RESET}}$

The $\overline{\text{RESET}}$ pin on the device is active-low, and setting the pin low externally will result in a Reset of the device.

AVR devices feature an internal pull-up resistor on the $\overline{\text{RESET}}$ pin, and an external pull-up resistor is usually not required.

The following figure shows the recommendation for connecting an external Reset switch to the device.

**Figure 4-4. Recommended External Reset Circuit Schematic**




**Typical values (Recommended):**
$C_1$: 1 µF (Filtering capacitor)
$R_1$: 330Ω (Switch series resistance)

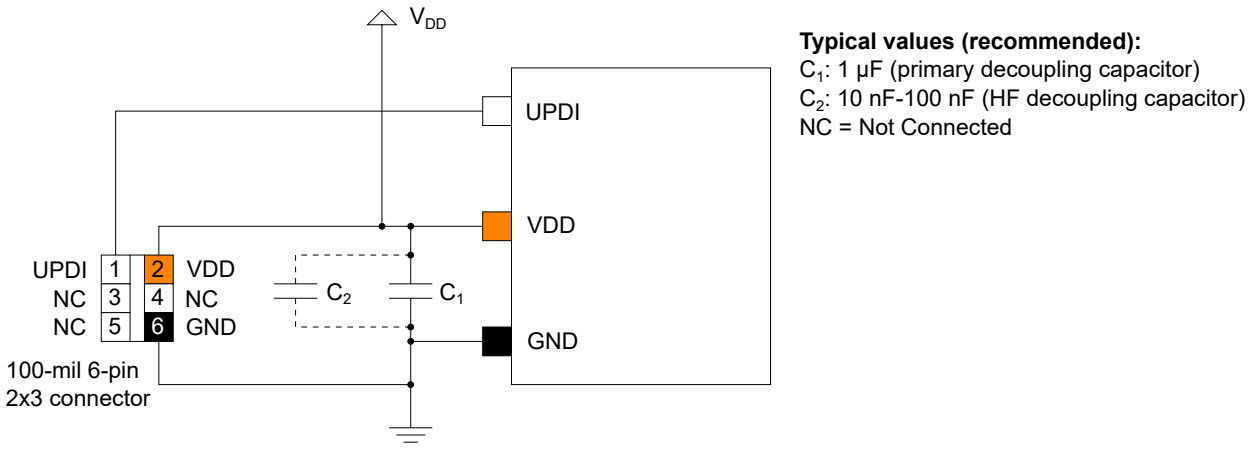
A resistor in series with the switch can safely discharge the filtering capacitor. This prevents a current surge when shorting the filtering capacitor, as this may cause a noise spike that can harm the system.

## 4.4 Connection for UPDI Programming

The standard connection for UPDI programming is a 100-mil 6-pin 2x3 header. Even though three pins are sufficient for programming most AVR devices, it is recommended to use a 2x3 header since most programming tools are delivered with 100-mil 6-pin 2x3 connectors.

The following figure shows the recommendation for connecting a UPDI connector to the device.

**Figure 4-5. Recommended UPDI Programming Circuit Schematic**




The decoupling capacitor between VDD and GND must be placed as close to the pin pair as possible. The decoupling capacitor must be included even if the UPDI connector is not included in the circuit.

## 4.5 Connecting External Crystal Oscillators

The use of external oscillators and the design of oscillator circuits are not trivial. This because there are many variables: $V_{DD}$, operating temperature range, crystal type and manufacture, loading capacitors, circuit layout, and PCB material. Presented here are some typical guidelines to help with the basic oscillator circuit design.

- Even the best performing oscillator circuits and high-quality crystals will not perform well if the layout and materials used during the assembly are not carefully considered
- The crystal circuit must be placed on the same side of the board as the device. Place the crystal circuit as close to the respective oscillator pins as possible and avoid long traces. This will reduce parasitic capacitance and increase immunity against noise and crosstalk. The load capacitors must be placed next to the crystal itself, on the same side of the board. Any kind of sockets must be avoided.
- Place a grounded copper area around the crystal circuit to isolate it from surrounding circuits. If the circuit board has two sides, the copper area on the bottom layer must be a solid area covering the crystal circuit. The copper area on the top layer must surround the crystal circuit and tie to the bottom layer area using via(s).
- Do not run any signal traces or power traces inside the grounded copper area. Avoid routing digital lines, especially clock lines, close to the crystal lines.
- If using a two-sided PCB, avoid any traces beneath the crystal. For a multilayer PCB, avoid routing signals below the crystal lines.
- Dust and humidity will increase parasitic capacitance and reduce signal isolation. A protective coating is recommended.
- Successful oscillator design requires good specifications of operating conditions, a component selection phase with initial testing, and testing in actual operating conditions to ensure that the oscillator performs as desired

For more detailed information about oscillators and oscillator circuit design, read the following application notes:

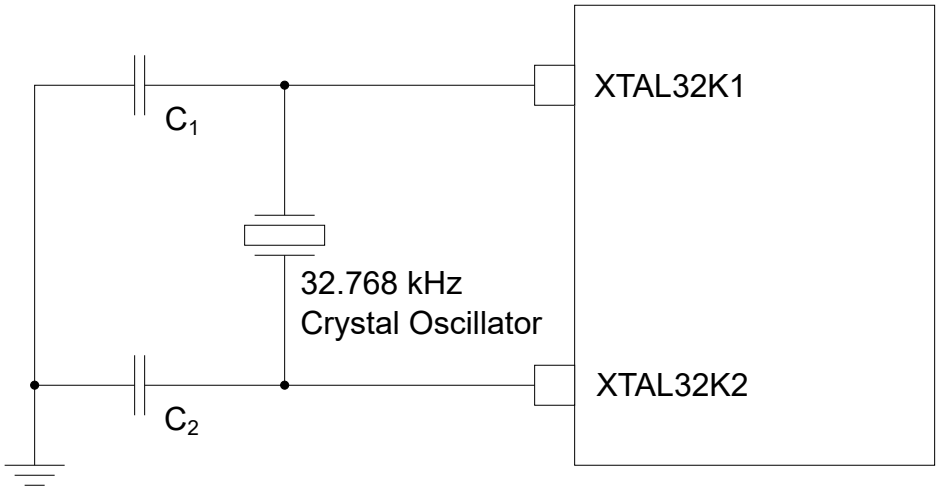
- *AN2648 - Selecting and Testing 32 KHz Crystal Oscillators for AVR® Microcontrollers*
- *AN949 - Making Your Oscillator Work*

### 4.5.1 Connection for XTAL32K (External 32.768 kHz crystal oscillator)

Ultra-low power 32.768 kHz oscillators typically dissipate significantly below 1 µW, and the current flowing in the circuit is, therefore, extremely small. The crystal frequency is highly dependent on the capacitive load.

The following figure shows how to connect an external 32.768 kHz crystal oscillator.

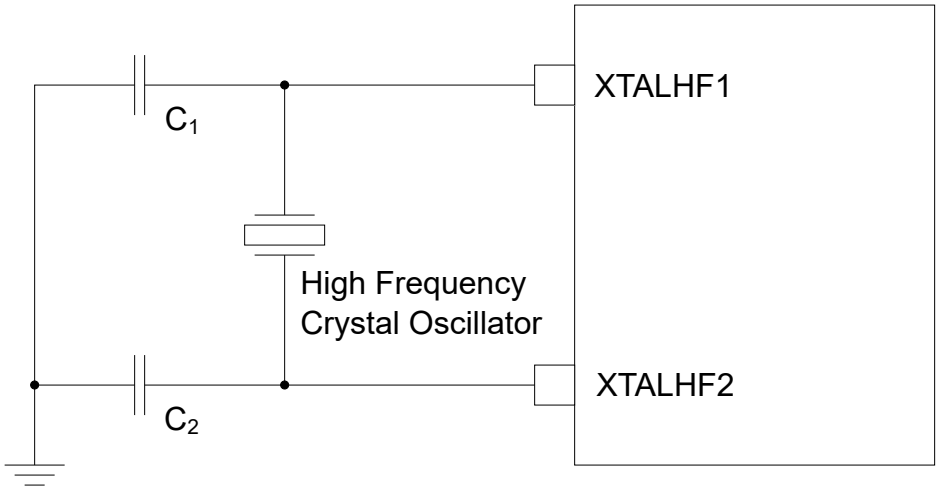
**Figure 4-6. Recommended External 32.768 kHz Oscillator Connection Circuit Schematic**




### 4.5.2 Connection for XTALHF (external HF crystal oscillator)

The following figure shows how to connect an external high-frequency crystal oscillator.

**Figure 4-7. Recommended External High-Frequency Oscillator Connection Circuit Schematic**

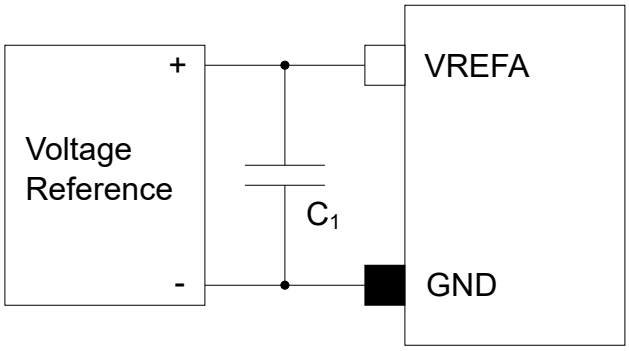



## 4.6 Connection for External Voltage Reference

If the design includes the use of an external voltage reference, the general recommendation is to use a suitable capacitor connected in parallel with the reference. The value of the capacitor depends on the nature of the reference and the type of electrical noise that needs to be filtered out.

Additional filtering components may be needed. This depends on the type of external voltage reference used.

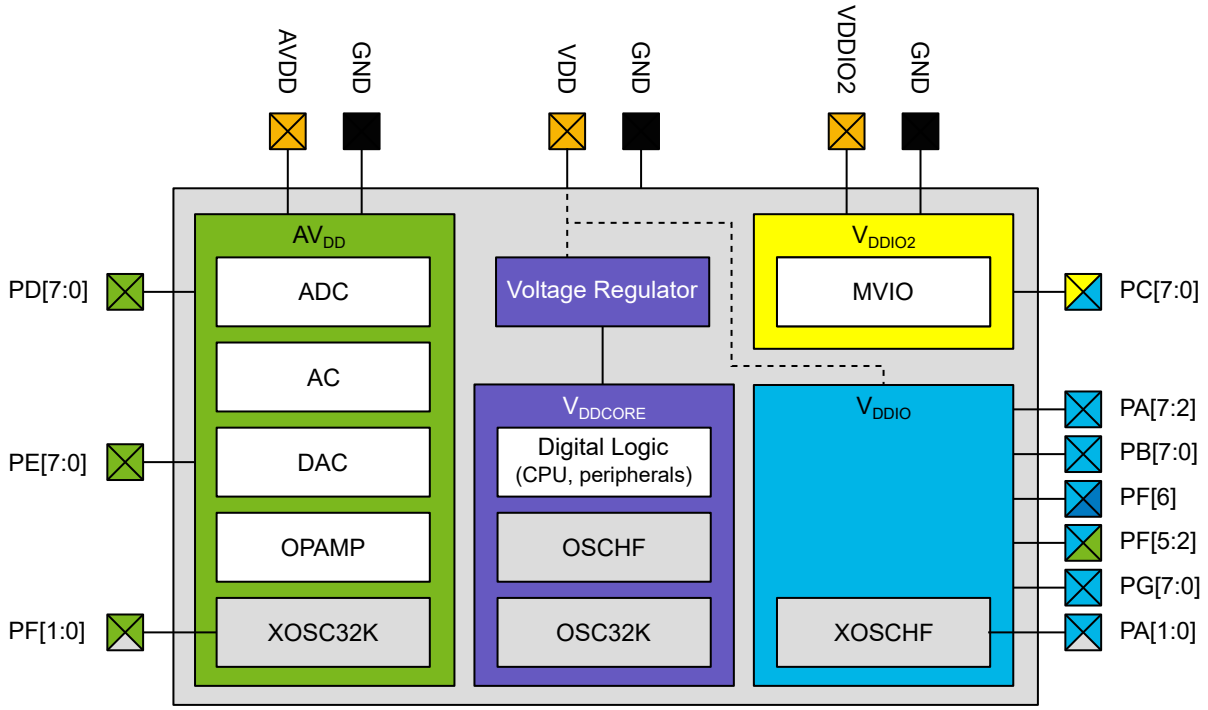
**Figure 4-8. Recommended External Voltage Reference Connection**



# 5. Power Supply

## 5.1 Power Domains

**Figure 5-1. Power Domain Overview**




**Note:** For the AVR® DBFamily of devices, AVDD is internally connected to VDD (not separate power domains).

The AVR DB family of devices has several power domains with the following power supply pins:

| | |
|---|---|
| VDD | Powers I/O lines, XOSCHF and the internal voltage regulator |
| AVDD | Powers I/O lines, XOSC32K (external 32.768 kHz oscillator) and the analog peripherals |
| VDDIO2 | Powers I/O lines, optionally at a different voltage from $V_{DD}$ |

The same voltage must be applied to all VDD and AVDD pins. This common voltage is referred to as $V_{DD}$ in the data sheet.

The ground pins, GND, are common to VDD, AVDD and VDDIO2.

A subset of the device I/O pins can be powered by $V_{DDIO2}$. This power domain is independent of $V_{DD}$. Refer to the *Multi-Voltage I/O* section for further information.

For recommendations on layout and decoupling, refer to the *Hardware Guidelines* section.

## 5.2 Voltage Regulator

The device has an internal voltage regulator that powers the $V_{DDCORE}$ domain. This domain has most of the digital logic and the internal oscillators. The voltage regulator balances power consumption when the CPU is active or in a sleep mode. Refer to the *Sleep Controller (SLPCTRL)* section for further information.

## 5.3 Power-Up

The $AV_{DD}$ voltage must ramp up closely with the $V_{DD}$ voltage during power-up to ensure proper operation.

If the device is configured in single-supply mode, the $V_{DDIO2}$ voltage must also rise closely to $V_{DD}$. In Dual-supply mode, the $V_{DDIO2}$ voltage can ramp up or down at any time without affecting the proper operation. Refer to the *Multi-Voltage I/O (MVIO)* section for further information.

The Power-on Reset (POR) and the Brown-out Detector (BOD) monitors $V_{DD}$ and will keep the system in reset if the voltage level is below the respective voltage thresholds. Refer to the *Reset Controller (RSTCTRL)* and *Brown-out Detector (BOD)* sections for further information.

Refer to the *Electrical Characteristics* section for further information on voltage thresholds.

# 6. Conventions

## 6.1 Numerical Notation

**Table 6-1. Numerical Notation**

| Symbol | Description |
|---|---|
| 165 | Decimal number |
| `0b0101` | Binary number |
| ‘`0101`’ | Binary numbers are given without prefix if unambiguous |
| 0x3B24 | Hexadecimal number |
| X | Represents an unknown or do not care value |
| Z | Represents a high-impedance (floating) state for either a signal or a bus |

## 6.2 Memory Size and Type

**Table 6-2. Memory Size and Bit Rate**

| Symbol | Description |
|---|---|
| KB | kilobyte ($2^{10}$B = 1024B) |
| MB | megabyte ($2^{20}$B = 1024 KB) |
| GB | gigabyte ($2^{30}$B = 1024 MB) |
| b | bit (binary ‘`0`’ or ‘`1`’) |
| B | byte (8 bits) |
| 1 kbit/s | 1,000 bit/s rate |
| 1 Mbit/s | 1,000,000 bit/s rate |
| 1 Gbit/s | 1,000,000,000 bit/s rate |
| word | 16-bit |

## 6.3 Frequency and Time

**Table 6-3. Frequency and Time**

| Symbol | Description |
|---|---|
| kHz | 1 kHz = $10^3$ Hz = 1,000 Hz |
| MHz | 1 MHz = $10^6$ Hz = 1,000,000 Hz |
| GHz | 1 GHz = $10^9$ Hz = 1,000,000,000 Hz |
| ms | 1 ms = $10^{-3}$s = 0.001s |
| µs | 1 µs = $10^{-6}$s = 0.000001s |
| ns | 1 ns = $10^{-9}$s = 0.000000001s |

## 6.4 Registers and Bits

**Table 6-4. Register and Bit Mnemonics**

| Symbol | Description |
|---|---|
| R/W | Read/Write accessible register bit. The user can read from and write to this bit. |
| R | Read-only accessible register bit. The user can only read this bit. Writes will be ignored. |
| W | Write-only accessible register bit. The user can only write this bit. Reading this bit will return an undefined value. |
| BITFIELD | Bitfield names are shown in uppercase. Example: INTMODE. |
| BITFIELD[n:m] | A set of bits from bit n down to m. Example: PINA[3:0] = {PINA3, PINA2, PINA1, PINA0}. |
| Reserved | Reserved bits, bit fields, and bit field values are unused and reserved for future use. For compatibility with future devices, always write reserved bits to '0' when the register is written. Reserved bits will always return zero when read. |
| PERIPHERALn | If several instances of the peripheral exist, the peripheral name is followed by a single number to identify one instance. Example: USARTn is the collection of all instances of the USART module, while USART3 is one specific instance of the USART module. |
| PERIPHERALx | If several instances of the peripheral exist, the peripheral name is followed by a single capital letter (A-Z) to identify one instance. Example: PORTx is the collection of all instances of the PORT module, while PORTB is one specific instance of the PORT module. |
| Reset | Value of a register after a Power-on Reset. This is also the value of registers in a peripheral after performing a software Reset of the peripheral, except for the Debug Control registers. |
| SET/CLR/TGL | Registers with SET/CLR/TGL suffix allow the user to clear and set bits in a register without doing a read-modify-write operation.<br>Each SET/CLR/TGL register is paired with the register it is affecting. Both registers in a register pair return the same value when read.<br>Example: In the PORT peripheral, the OUT and OUTSET registers form such a register pair. The contents of OUT will be modified by a write to OUTSET. Reading OUT and OUTSET will return the same value.<br>Writing a '1' to a bit in the CLR register will clear the corresponding bit in both registers.<br>Writing a '1' to a bit in the SET register will set the corresponding bit in both registers.<br>Writing a '1' to a bit in the TGL register will toggle the corresponding bit in both registers. |

### 6.4.1 Addressing Registers from Header Files

In order to address registers in the supplied C header files, the following rules apply:

1. A register is identified by <peripheral_instance_name>.<register_name>, e.g., CPU.SREG, USART2.CTRLA, or PORTB.DIR.
2. The peripheral name is given in the "Peripheral Address Map" in the "Peripherals and Architecture" section.
3. <peripheral_instance_name> is obtained by substituting any n or x in the peripheral name with the correct instance identifier.
4. When assigning a predefined value to a peripheral register, the value is constructed following the rule: <peripheral_name>_<bit_field_name>_<bit_field_value>_gc

   <peripheral_name> is <peripheral_instance_name>, but remove any instance identifier.

   <bit_field_value> can be found in the "Name" column in the tables in the Register Description sections describing the bit fields of the peripheral registers.

**Example 6-1. Register Assignments**

```
// EVSYS channel 0 is driven by TCB3 OVF event
EVSYS.CHANNEL0 = EVSYS_CHANNEL0_TCB3_OVF_gc;

// USART0 RXMODE uses Double Transmission Speed
USART0.CTRLB = USART_RXMODE_CLK2X_gc;
```

**Note:** For peripherals with different register sets in different modes, <peripheral_instance_name> and <peripheral_name> must be followed by a mode name, for example:

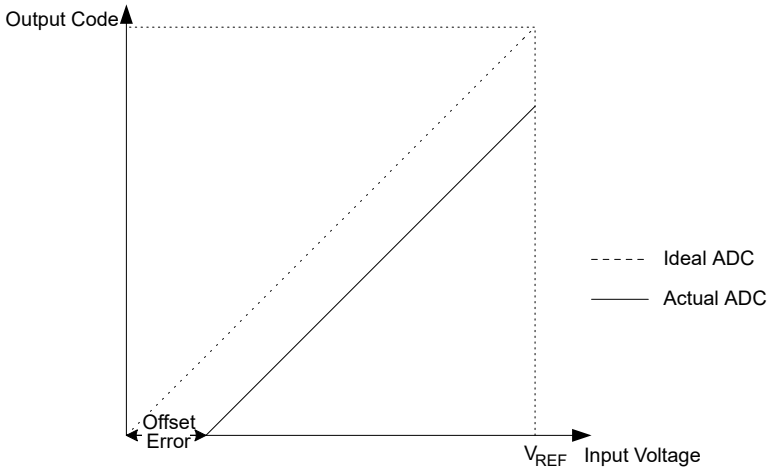
```
// TCA0 in Normal Mode (SINGLE) uses waveform generator in frequency mode
TCA0.SINGLE.CTRL=TCA_SINGLE_WGMODE_FRQ_gc;
```

## 6.5 ADC Parameter Definitions

An ideal n-bit single-ended ADC converts a voltage linearly between GND and $V_{REF}$ in $2^n$ steps (LSb). The lowest code is read as '0', and the highest code is read as '$2^n$-1'. Several parameters describe the deviation from the ideal behavior:

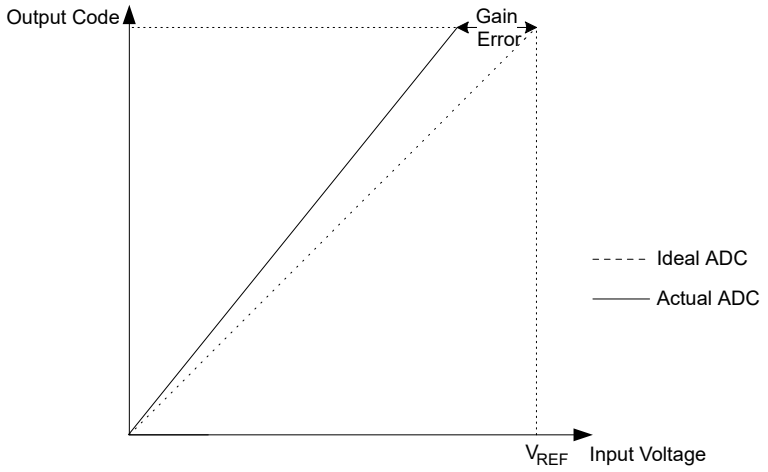
**Offset Error** The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSb). Ideal value: 0 LSb.

**Figure 6-1. Offset Error**




**Gain Error** After adjusting for offset, the gain error is found as the deviation of the last transition (e.g., 0x3FE to 0x3FF for a 10-bit ADC) compared to the ideal transition (at 1.5 LSb below maximum). Ideal value: 0 LSb.

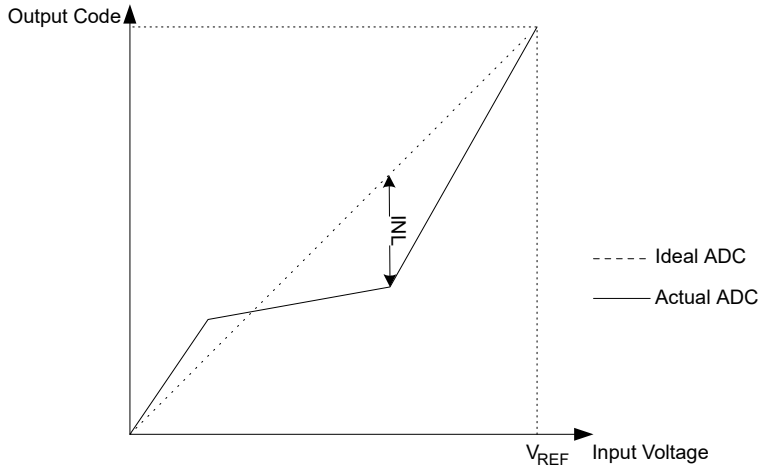
**Figure 6-2. Gain Error**




**Integral Nonlinearity (INL)**

After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSb.

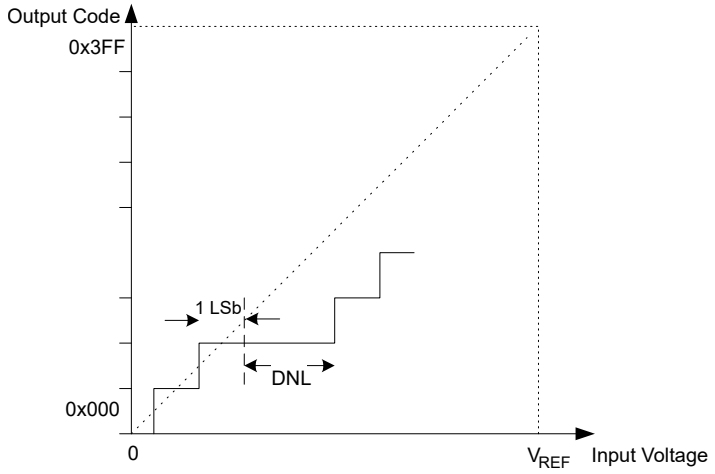
**Figure 6-3. Integral Nonlinearity**




**Differential Nonlinearity (DNL)**

The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSb). Ideal value: 0 LSb.

**Figure 6-4. Differential Nonlinearity**



**Quantization Error** Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSb wide) will code to the same value. Always ±0.5 LSb.

**Absolute Accuracy** The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of all errors mentioned before. Ideal value: ±0.5 LSb.

# 7. AVR® CPU

## 7.1 Features

- 8-bit, High-Performance AVR RISC CPU:
  - 135 instructions
  - Hardware multiplier
- 32 8-bit Registers Directly Connected to the ALU
- Stack in RAM
- Stack Pointer Accessible in I/O Memory Space
- Direct Addressing of up to 64 KB of Unified Memory
- Efficient Support for 8-, 16-, and 32-bit Arithmetic
- Configuration Change Protection for System-Critical Features
- Native On-Chip Debugging (OCD) Support:
  - Two hardware breakpoints
  - Change of flow, interrupt, and software breakpoints
  - Run-time read-out of Stack Pointer (SP) register, Program Counter (PC), and Status Register (SREG)
  - Register file read- and writable in Stopped mode

## 7.2 Overview

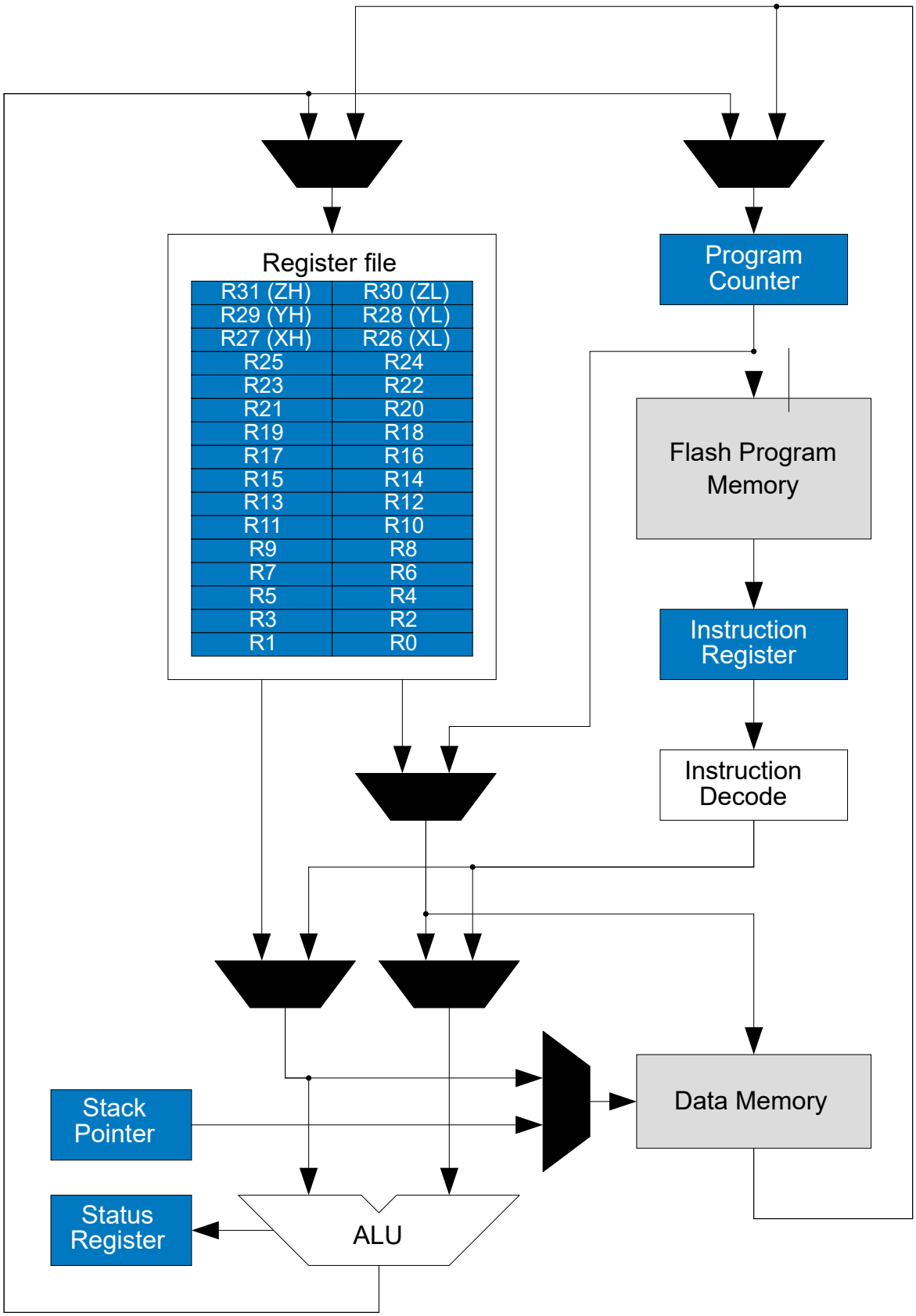
The AVR CPU can access memories, perform calculations, control peripherals, execute instructions from the program memory, and handling interrupts.

## 7.3 Architecture

To maximize performance and parallelism, the AVR CPU uses a Harvard architecture with separate buses for program and data. Instructions in the program memory are executed with a single-level pipeline. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This enables instructions to be executed on every clock cycle.

Refer to the *Instruction Set Summary* section for a summary of all AVR instructions.

**Figure 7-1. AVR® CPU Architecture**



### 7.3.1 Arithmetic Logic Unit (ALU)

The Arithmetic Logic Unit (ALU) supports arithmetic and logic operations between working registers, or between a constant and a working register. Also, single-register operations can be executed.

The ALU operates in a direct connection with all the 32 general purpose working registers in the register file. Arithmetic operations between working registers or between a working register and an immediate operand are executed in a single clock cycle, and the result is stored in the register file. After an arithmetic or logic operation, the Status Register (CPU.SREG) is updated to reflect information about the result of the operation.

ALU operations are divided into three main categories – arithmetic, logical, and bit functions. Both 8- and 16-bit arithmetic are supported, and the instruction set allows for efficient implementation of the 32-bit arithmetic. The hardware multiplier supports signed and unsigned multiplication and fractional formats.

#### 7.3.1.1 Hardware Multiplier

The multiplier is capable of multiplying two 8-bit numbers into a 16-bit result. The hardware multiplier supports different variations of signed and unsigned integer and fractional numbers:

- Multiplication of signed/unsigned integers
- Multiplication of signed/unsigned fractional numbers
- Multiplication of a signed integer with an unsigned integer
- Multiplication of a signed fractional number with an unsigned fractional number

A multiplication takes two CPU clock cycles.

## 7.4 Functional Description

### 7.4.1 Program Flow

After being reset, the CPU will execute instructions from the lowest address in the Flash program memory, 0x0000.

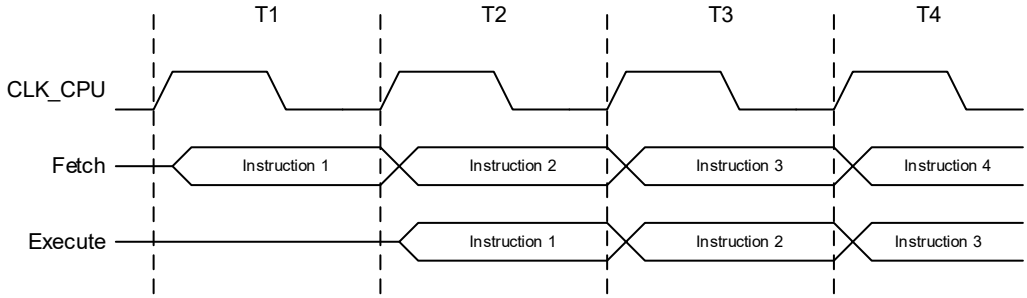
The CPU supports instructions that can change the program flow conditionally or unconditionally and are capable of addressing the whole address space directly. Most AVR instructions use a 16-bit word format, and a limited number use a 32-bit format.

During interrupts and subroutine calls, the return address PC is stored on the stack as a word pointer. The stack is allocated in the general data SRAM, and consequently, the stack size is limited only by the total SRAM size and the usage of the SRAM. After the Stack Pointer (SP) is reset, it points to the highest address in the internal SRAM. The SP is read/write accessible in the I/O memory space, enabling easy implementation of multiple stacks or stack areas. The data SRAM can easily be accessed through different `LD*`/`ST*` instructions supported by the AVR CPU. See the *Instruction Set Summary* section for details.

### 7.4.2 Instruction Execution Timing

The AVR CPU is clocked by the CPU clock, CLK_CPU. No internal clock division is applied. The figure below shows the parallel instruction fetches and executions enabled by the Harvard architecture and the fast-access register file concept. This is a two-stage pipelining concept enabling up to 1 MIPS/MHz performance with high efficiency.

**Figure 7-2. The Parallel Instruction Fetches and Executions**



### 7.4.3 Status Register

The Status Register (CPU.SREG) contains information about the result of the most recently executed arithmetic or logic instructions. This information can be used for altering the program flow to perform conditional operations.

CPU.SREG is updated after all ALU operations, as specified in the *Instruction Set Summary* section. This will, in many cases, remove the need for using the dedicated compare instructions, resulting in a faster and more compact code. CPU.SREG is not automatically stored or restored when entering or returning from an Interrupt Service Routine (ISR). Therefore, maintaining the Status Register between context switches must be handled by user-defined software. CPU.SREG is accessible in the I/O memory space.

### 7.4.4 Stack and Stack Pointer

The stack is used for storing return addresses after interrupts and subroutine calls. Also, it can be used for storing temporary data. The Stack Pointer (SP) always points to the top of the stack. The address pointed to by the SP is stored in the Stack Pointer (CPU.SP) register. CPU.SP is implemented as two 8-bit registers that are accessible in the I/O memory space.

Data are pushed and popped from the stack using the `PUSH` and `POP` instructions. The stack grows from higher to lower memory locations. This means that when pushing data onto the stack, the SP decreases, and when popping data off the stack, the SP increases. The SP is automatically set to the highest address of the internal SRAM after being reset. If the stack is changed, it must be set to point within the SRAM address space (see the SRAM Data Memory section in the Memories section for the SRAM start address), and it must be defined before any subroutine calls are executed and before interrupts are enabled. See the table below for SP details.

**Table 7-1. Stack Pointer Instructions**

| Instruction | Stack Pointer | Description |
|---|---|---|
| `PUSH` | Decremented by 1 | Data are pushed onto the stack |
| `CALL`<br>`ICALL`<br>`EICALL`<br>`RCALL` | Decremented by 2 | A return address is pushed onto the stack with a subroutine call or interrupt |
| `POP` | Incremented by 1 | Data are popped from the stack |
| `RET`<br>`RETI` | Incremented by 2 | A return address is popped from the stack with a return from subroutine or return from interrupt |

During interrupts or subroutine calls, the return address is automatically pushed on the stack as a word, and the SP is decremented by two. The return address consists of two bytes, and the Least Significant Byte (LSB) is pushed on the stack first (at the higher address). As an example, a byte pointer return address of 0x0006 is saved on the stack as 0x0003 (shifted one bit to the right), pointing to the fourth 16-bit instruction word in the program memory. The return address is popped off the stack with `RETI` (when returning from interrupts) and `RET` (when returning from subroutine calls), and the SP is incremented by two.

The SP is decremented by one when data are pushed on the stack with the `PUSH` instruction and incremented by one when data are popped off the stack using the `POP` instruction.

To prevent corruption when updating the SP from software, a write to SPL will automatically disable interrupts for up to four instructions or until the next I/O memory write, whichever comes first.

### 7.4.5 Register File

The register file consists of 32 8-bit general purpose working registers used by the CPU. The register file is located in a separate address space from the data memory.

All CPU instructions that operate on working registers have direct and single-cycle access to the register file. Some limitations apply to which working registers can be accessed by an instruction, like the constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI ORI, and LDI. These instructions apply to the second half of the working registers in the register file, R16 to R31. See the *AVR Instruction Set Manual* for further details.

**Figure 7-3. AVR® CPU General Purpose Working Registers**

| 7 … 0 | Addr. | |
|---|---|---|
| R0 | 0x00 | |
| R1 | 0x01 | |
| R2 | 0x02 | |
| … | | |
| R13 | 0x0D | |
| R14 | 0x0E | |
| R15 | 0x0F | |
| R16 | 0x10 | |
| R17 | 0x11 | |
| … | | |
| R26 | 0x1A | X-register Low Byte |
| R27 | 0x1B | X-register High Byte |
| R28 | 0x1C | Y-register Low Byte |
| R29 | 0x1D | Y-register High Byte |
| R30 | 0x1E | Z-register Low Byte |
| R31 | 0x1F | Z-register High Byte |

#### 7.4.5.1 The X-, Y-, and Z-Registers

Working registers R26...R31 have added functions besides their general purpose usage.

These registers can form 16-bit Address Pointers for indirect addressing of data memory. These three address registers are called the X-register, Y-register, and Z-register. The Z-register can also be used as Address Pointer for program memory.

**Figure 7-4. The X-, Y-, and Z-Registers**

| | | | | | |
|---|---|---|---|---|---|
| Bit (individually) | 7 | R27 | 0 | 7 | R26 | 0 |
| X-register | **XH** | | | **XL** | | |
| Bit (X-register) | 15 | | 8 | 7 | | 0 |
| Bit (individually) | 7 | R29 | 0 | 7 | R28 | 0 |
| Y-register | **YH** | | | **YL** | | |
| Bit (Y-register) | 15 | | 8 | 7 | | 0 |
| Bit (individually) | 7 | R31 | 0 | 7 | R30 | 0 |
| Z-register | **ZH** | | | **ZL** | | |
| Bit (Z-register) | 15 | | 8 | 7 | | 0 |

The lowest register address holds the Least Significant Byte (LSB), and the highest register address holds the Most Significant Byte (MSB). These address registers can function as fixed displacement, automatic increment, and automatic decrement, with different `LD*`/`ST*` instructions. See the *Instruction Set Summary* section for details.

#### 7.4.5.2 Extended Pointers

To access program memory above 64 KB, the Address Pointer must be larger than 16 bits. This is done by concatenating one of the address extension I/O registers (RAMPZ) with the internal Z-pointer. The RAMPZ register then holds the Most Significant Byte (MSB) in a 24-bit address or Address Pointer.

This address extension register is available only on devices with more than 64 KB of program memory. For the devices where extension pointers are required, only the number of bits required to address the whole program and data memory space in the device are implemented.

##### 7.4.5.2.1 Extended Program Memory Pointer

The RAMPZ register is concatenated with the Z-register to enable indirect addressing of the entire program memory.

**Figure 7-5. The Combined RAMPZ + Z Register**

| Bit (Individually) | 7 … 0 | 7 … 0 | 7 … 0 |
|---|---|---|---|
| | RAMPZ | ZH | ZL |
| Bit (Z-pointer) | 23 … 16 | 15 … 8 | 7 … 0 |

When reading (`ELPM`) above the first 64 KB of the program memory, RAMPZ is concatenated with the Z-register to form the 24-bit address. The `LPM` instruction is not affected by the RAMPZ setting.

### 7.4.6 Configuration Change Protection (CCP)

System critical I/O register settings are protected from accidental modification, and Flash self-programming is protected from accidental programming. This is handled globally by the Configuration Change Protection (CCP) register.

Changes to the protected I/O registers or bits, or execution of protected instructions, are only possible after the CPU writes a signature to the CCP register. The different signatures are listed in the description of the CCP (CPU.CCP) register.

Once the correct signature is written by the CPU, interrupts will be ignored for the duration of the configuration change enable period. Any interrupt request (including non-maskable interrupts) during the CCP period will set the corresponding interrupt flag as normal, and the request is kept pending. After the CCP period is completed, any pending interrupts are executed according to their level and priority.

There are two modes of CCP operation: One for protected I/O registers, and one for protected self-programming.

#### 7.4.6.1 Sequence for Write Operation to Configuration Change Protected I/O Registers

To write to I/O registers protected by CCP, the following steps are required:

1. The software writes the signature that enables change of protected I/O registers to the CCP bit field in the CPU.CCP register.
2. Within four instructions, the software must write the appropriate data to the protected I/O register.
   The protected change is automatically disabled after CPU executes a write instruction.

#### 7.4.6.2 Sequence for Execution of Self-Programming

To execute self-programming (the execution of writes to the NVM controller's command register), the following steps are required:

1. The software temporarily enables self-programming by writing the SPM signature to the CCP (CPU.CCP) register.
2. Within four instructions, the software must execute the appropriate instruction or change to NVM Command Register.
   The protected change is automatically disabled after the CPU executes a write instruction.

### 7.4.7 On-Chip Debug Capabilities

The AVR CPU includes native On-Chip Debug (OCD) support. It includes some powerful debug capabilities to enable profiling and detailed information about the CPU state. It is possible to alter the CPU state and resume code execution. Also, normal debug capabilities like hardware Program Counter breakpoints, breakpoints on change of flow instructions, breakpoints on interrupts, and software breakpoints (`BREAK` instruction) are present. Refer to the *Unified Program and Debug Interface* section for details about OCD.

## 7.5 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x03 | Reserved | | | | | | | | | |
| 0x04 | CCP | 7:0 | CCP[7:0] | | | | | | | |
| 0x05<br>...<br>0x0A | Reserved | | | | | | | | | |
| 0x0B | RAMPZ | 7:0 | RAMPZ[7:0] | | | | | | | |
| 0x0C | Reserved | | | | | | | | | |
| 0x0D | SP | 7:0 | SP[7:0] | | | | | | | |
| | | 15:8 | SP[15:8] | | | | | | | |
| 0x0F | SREG | 7:0 | I | T | H | S | V | N | Z | C |

## 7.6 Register Description

### 7.6.1 Configuration Change Protection

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CCP[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 – CCP[7:0]** Configuration Change Protection
Writing the correct signature to this bit field allows changing protected I/O registers or executing protected instructions within the next four CPU instructions executed.
All interrupts are ignored during these cycles. After these cycles are completed, the interrupts will automatically be handled again by the CPU, and any pending interrupts will be executed according to their level and priority.
When the protected I/O register signature is written, CCP[0] will read as '1' as long as the CCP feature is enabled.
When the protected self-programming signature is written, CCP[1] will read as '1' as long as the CCP feature is enabled.
CCP[7:2] will always read as '0'.

| Value | Name | Description |
|---|---|---|
| 0x9D | SPM | Allow Self-Programming |
| 0xD8 | IOREG | Unlock protected I/O registers |

### 7.6.2 Stack Pointer

**Name:** SP
**Offset:** 0x0D
**Reset:** Top of stack
**Property:** -

The CPU.SP register holds the Stack Pointer (SP) that points to the top of the stack. After being reset, the SP points to the highest internal SRAM address.

Only the number of bits required to address the available data memory, including external memory (up to 64 KB), is implemented for each device. Unused bits will always read as '`0`'.

The CPU.SPL and CPU.SPH register pair represents the 16-bit value, CPU.SP. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

To prevent corruption when updating the SP from software, a write to CPU.SPL will automatically disable interrupts for the next four instructions or until the next I/O memory write, whichever comes first.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | SP[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | | | | | |

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

**Bits 15:8 – SP[15:8]** Stack Pointer High Byte
These bits hold the MSB of the 16-bit register.

**Bits 7:0 – SP[7:0]** Stack Pointer Low Byte
These bits hold the LSB of the 16-bit register.

### 7.6.3 Status Register

**Name:** SREG
**Offset:** 0x0F
**Reset:** 0x00
**Property:** -

The Status Register contains information about the result of the most recently executed arithmetic or logic instructions. For details about the bits in this register and how they are influenced by different instructions, see the *Instruction Set Summary* section.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | I | T | H | S | V | N | Z | C |
| Access | R/W | 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 – I** Global Interrupt Enable Bit
Writing a '`1`' to this bit enables interrupts on the device.
Writing a '`0`' to this bit disables interrupts on the device, independent of the individual interrupt enable settings of the peripherals.
This bit is not cleared by hardware while entering an Interrupt Service Routine (ISR) or set when the RETI instruction is executed.
This bit can be set and cleared by software with the `SEI` and `CLI` instructions.
Changing the I bit through the I/O register results in a one-cycle Wait state on the access.

**Bit 6 – T** Transfer Bit
The bit copy instructions, Bit Load (`BLD`) and Bit Store (`BST`), use the T bit as source or destination for the operated bit.

**Bit 5 – H** Half Carry Flag
This flag is set when there is a half carry in arithmetic operations that support this, and is cleared otherwise. Half carry is useful in BCD arithmetic.

**Bit 4 – S** Sign Flag
This flag is always an Exclusive Or (*XOR*) between the Negative flag (N) and the Two's Complement Overflow flag (V).

**Bit 3 – V** Two's Complement Overflow Flag
This flag is set when there is an overflow in arithmetic operations that support this, and is cleared otherwise.

**Bit 2 – N** Negative Flag
This flag is set when there is a negative result in an arithmetic or logic operation, and is cleared otherwise.

**Bit 1 – Z** Zero Flag
This flag is set when there is a zero result in an arithmetic or logic operation, and is cleared otherwise.

**Bit 0 – C** Carry Flag
This flag is set when there is a carry in an arithmetic or logic operation, and is cleared otherwise.

### 7.6.4 Extended Z-Pointer Register

**Name:** RAMPZ
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

This register is concatenated with the Z-register for indirect addressing (`LD`/`LDD`/`ST`/`STD`) of the whole data memory space on devices with more than 64 KB of data memory. RAMPZ is concatenated with the Z-register when reading the (`ELPM`) program memory locations above the first 64 KB and writing the (`SPM`) program memory locations above the first 128 KB of the program memory.

This register is not available if the data memory and program memory in the device are less than 64 KB.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RAMPZ[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 – RAMPZ[7:0]** Extended Z-pointer Address Bits
These bits hold the MSB of the 24-bit address created by RAMPZ and the 16-bit Z-register. Only the number of bits required to address the available data and program memory is implemented for each device. Unused bits will always read as '`0`'.

# 8. Memories

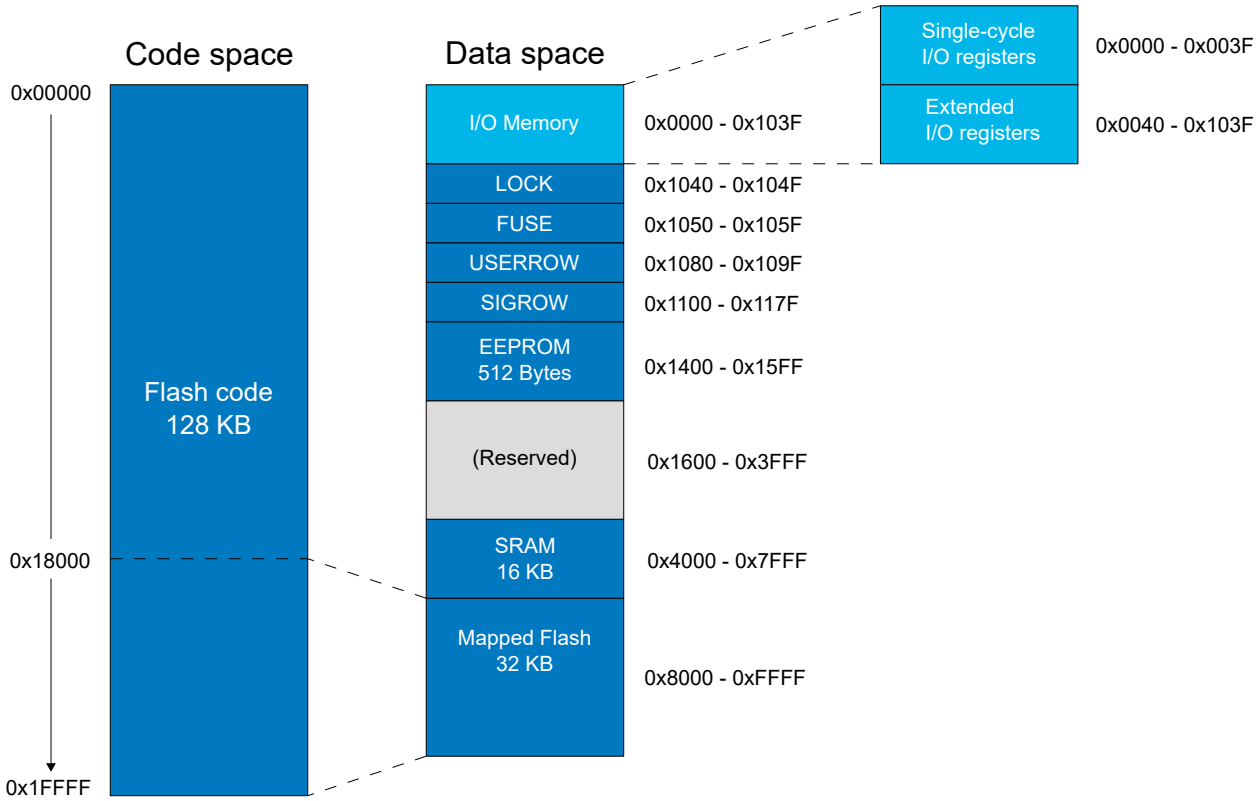
## 8.1 Overview

The main memories of the AVR128DB28/32/48/64 devices are SRAM data memory space, EEPROM data memory space, and Flash program memory space. In addition, the peripheral registers are located in the I/O memory space.

## 8.2 Memory Map

The figure below shows the memory map for the largest memory derivative in the AVR DB family. Refer to the subsequent sections and the *Peripheral Address Map* table for further details.

**Figure 8-1. Memory Map**




## 8.3 In-System Reprogrammable Flash Program Memory

The AVR128DB28/32/48/64 contains 128 KB on-chip in-system reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized with 16-bit data width. For write protection, the Flash program memory space can be divided into three sections: Boot Code section, Application Code section, and Application Data section. Code placed in one section may be restricted from writing to addresses in other sections. The Program Counter (PC) can address the whole program memory.

Refer to the Code Size (CODESIZE) and Boot Size (BOOTSIZE) descriptions and the *Nonvolatile Memory Controller* section for further details.

The Program Counter can address the whole program memory. The procedure for writing Flash memory is described in detail in the Nonvolatile Memory Controller (NVMCTRL) peripheral documentation.

Each 32 KB section from Flash memory is mapped into the data memory space and is accessible with `LD/ST` instructions. For `LD/ST` instructions, the Flash is mapped from address 0x8000 to 0xFFFF. The entire Flash memory space can be accessed with the `LPM/SPM` instruction. For the `LPM/SPM` instruction, the Flash start address is 0x0000.

**Table 8-1. Physical Properties of Flash Memory**

| Property | AVR128DB64<br>AVR128DB48<br>AVR128DB32<br>AVR128DB28 |
|---|---|
| Size | 128 KB |
| Page size | 512B |
| Number of pages | 256 |
| Start address in data space | 0x8000 |
| Start address in code space | 0x0 |

## 8.4 SRAM Data Memory

The primary task of the SRAM memory is to store application data. Also, the program stack is located at the end of SRAM. It is not possible to execute from SRAM.

**Table 8-2. Physical Properties of SRAM Memory**

| Property | AVR128DB64<br>AVR128DB48<br>AVR128DB32<br>AVR128DB28 |
|---|---|
| Size | 16 KB |
| Start address | 0x4000 |

## 8.5 EEPROM Data Memory

The task of the EEPROM memory is to store nonvolatile application data. The EEPROM memory supports single- and multi-byte read and write. The EEPROM is controlled by the Nonvolatile Memory Controller (NVMCTRL).

**Table 8-3. Physical Properties of EEPROM Memory**

| Property | AVR128DB64<br>AVR128DB48<br>AVR128DB32<br>AVR128DB28 |
|---|---|
| Size | 512B |
| Start address | 0x1400 |

## 8.6 SIGROW - Signature Row

The content of the Signature Row fuses (SIGROW) is pre-programmed and read-only. SIGROW contains information such as device ID, serial number, and calibration values.

All the AVR128DB28/32/48/64 devices have a three-byte device ID that identifies the device. The device ID can be read using the Unified Program and Debug Interface (UPDI), also when a device is locked. The device ID for the AVR128DB28/32/48/64 devices consists of three signature bytes, which is given by the following table.

**Table 8-4. Device ID**

| Device Name | Signature Byte Address and Value | | |
|---|---|---|---|
| | 0x00 | 0x01 | 0x02 |
| AVR128DB28 | 0x1E | 0x97 | 0x0E |
| AVR128DB32 | 0x1E | 0x97 | 0x0D |
| AVR128DB48 | 0x1E | 0x97 | 0x0C |
| AVR128DB64 | 0x1E | 0x97 | 0x0B |

### 8.6.1 Signature Row Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | DEVID0 | 7:0 | DEVID[7:0] | | | | | | | |
| 0x01 | DEVID1 | 7:0 | DEVID[7:0] | | | | | | | |
| 0x02 | DEVID2 | 7:0 | DEVID[7:0] | | | | | | | |
| 0x03 | Reserved | | | | | | | | | |
| 0x04 | TEMPSENSE0 | 7:0 | TEMPSENSE[7:0] | | | | | | | |
| | | 15:8 | TEMPSENSE[15:8] | | | | | | | |
| 0x06 | TEMPSENSE1 | 7:0 | TEMPSENSE[7:0] | | | | | | | |
| | | 15:8 | TEMPSENSE[15:8] | | | | | | | |
| 0x08<br>...<br>0x0F | Reserved | | | | | | | | | |
| 0x10 | SERNUM0 | 7:0 | SERNUM[7:0] | | | | | | | |
| ... | | | | | | | | | | |
| 0x1F | SERNUM15 | 7:0 | SERNUM[7:0] | | | | | | | |

### 8.6.2 Signature Row Description

#### 8.6.2.1 Device ID n

**Name:** DEVIDn
**Offset:** 0x00 + n*0x01 [n=0..2]
**Reset:** [Signature byte n of device ID]
**Property:** -

Each device has a device ID identifying the device and its properties such as memory sizes and pin count. This can be used to identify a device and hence, the available features by software. The Device ID consists of three bytes: SIGROW.DEVID[2:0].

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DEVID[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – DEVID[7:0]** Byte n of the Device ID

#### 8.6.2.2 Temperature Sensor Calibration n

**Name:** TEMPSENSEn
**Offset:** 0x04 + n*0x02 [n=0..1]
**Reset:** [Temperature sensor calibration value]
**Property:** -

The Temperature Sensor Calibration value contains correction factors for temperature measurements from the on-chip temperature sensor. The SIGROW.TEMPSENSE0 is a correction factor for the gain/slope (unsigned), and SIGROW.TEMPSENSE1 is a correction factor for the offset (signed).

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TEMPSENSE[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 |
|---|---|---|---|---|---|---|---|---|
| | TEMPSENSE[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 15:0 – TEMPSENSE[15:0]** Temperature Sensor Calibration word n
Refer to the *Analog-to-Digital Converter* section for a description of how to use the value stored in this bit field.

#### 8.6.2.3 Serial Number Byte n

**Name:** SERNUMn
**Offset:** 0x10 + n*0x01 [n=0..15]
**Reset:** [Byte n of device serial number]
**Property:** -

Each device has an individual serial number, representing a unique ID. This can be used to identify a specific device in the field. The serial number consists of 16 bytes: SIGROW.SERNUM[15:0].

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SERNUM[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – SERNUM[7:0]** Serial Number Byte n

## 8.7 USERROW - User Row

The AVR128DB28/32/48/64 devices have a special 32-byte memory section called the User Row (USERROW). USERROW can be used for end-production data and is not affected by chip erase. It can be written by the Unified Program and Debug Interface (UPDI) even if the part is locked, which enables storage of final configuration without having access to any other memory. When the part is locked, the UPDI is not allowed to read the content of the USERROW.

The CPU can write and read this memory as a normal flash. Refer to the *System Memory Address Map* for further details.

## 8.8 FUSE - Configuration and User Fuses

Fuses are part of the nonvolatile memory and hold factory calibration and device configuration. The fuses can be read by the CPU or the UPDI, but can only be programmed or cleared by the UPDI. The configuration values stored in the fuses are written to their respective target registers at the end of the start-up sequence.

The fuses for peripheral configuration (FUSE) are pre-programmed but can be altered by the user. Altered values in the configuration fuse will be effective only after a Reset.
**Note:** When writing the fuses, all reserved bits must be written to ‘0’.

### 8.8.1 Fuse Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | WDTCFG | 7:0 | WINDOW[3:0] | | | | PERIOD[3:0] | | | |
| 0x01 | BODCFG | 7:0 | LVL[2:0] | | | SAMPFREQ | ACTIVE[1:0] | | SLEEP[1:0] | |
| 0x02 | OSCCFG | 7:0 | | | | | CLKSEL[3:0] | | | |
| 0x03<br>...<br>0x04 | Reserved | | | | | | | | | |
| 0x05 | SYSCFG0 | 7:0 | CRCSRC[1:0] | | CRCSEL | | RSTPINCFG | | | EESAVE |
| 0x06 | SYSCFG1 | 7:0 | | | | MVSYSCFG[1:0] | | SUT[2:0] | | |
| 0x07 | CODESIZE | 7:0 | CODESIZE[7:0] | | | | | | | |
| 0x08 | BOOTSIZE | 7:0 | BOOTSIZE[7:0] | | | | | | | |

### 8.8.2 Fuse Description

#### 8.8.2.1 Watchdog Timer Configuration

**Name:** WDTCFG
**Offset:** 0x00
**Default:** 0x00
**Property:** -

The default value given in this fuse description is the factory-programmed value, and should not be mistaken for the Reset value.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WINDOW[3:0] | | | | PERIOD[3:0] | | | |
| Access | R | R | R | R | R | R | R | R |
| Default | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:4 – WINDOW[3:0]** Watchdog Window Time-out Period
This value is loaded into the WINDOW bit field of the Watchdog Control A (WDT.CTRLA) register at the end of the start-up sequence, after power-on or Reset.

**Bits 3:0 – PERIOD[3:0]** Watchdog Time-out Period
This value is loaded into the PERIOD bit field of the Watchdog Control A (WDT.CTRLA) register at the end of the start-up sequence after power-on or Reset.

### 8.8.2.2 Brown-Out Detector Configuration

**Name:** BODCFG
**Offset:** 0x01
**Default:** 0x00
**Property:** -

The bit values of this fuse register are written to the corresponding BOD configuration registers at the start-up.

The default value given in this fuse description is the factory-programmed value, and should not be mistaken for the Reset value.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LVL[2:0] | | | SAMPFREQ | ACTIVE[1:0] | | SLEEP[1:0] | |
| Access | R | R | R | R | R | R | R | R |
| Default | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:5 – LVL[2:0]** BOD Level
This value is loaded into the LVL bit field of the BOD Control B (BOD.CTRLB) register during Reset.

| Value | Name | Description |
|---|---|---|
| 0x0 | BODLEVEL0 | 1.9V |
| 0x1 | BODLEVEL1 | 2.45V |
| 0x2 | BODLEVEL2 | 2.70V |
| 0x3 | BODLEVEL3 | 2.85V |
| Other | - | Reserved |

**Note:** Values in the **Description** column are typical values. Refer to the *Electrical Characteristics* section for further details.

**Bit 4 – SAMPFREQ** BOD Sample Frequency
This value is loaded into the Sample Frequency (SAMPFREQ) bit of the BOD Control A (BOD.CTRLA) register during Reset. Refer to the *Brown-out Detector* section for further details.

| Value | Name | Description |
|---|---|---|
| 0 | 128HZ | The sample frequency is 128 Hz |
| 1 | 32HZ | The sample frequency is 32 Hz |

**Bits 3:2 – ACTIVE[1:0]** BOD Operation Mode in Active and Idle
This value is loaded into the ACTIVE bit field of the BOD Control A (BOD.CTRLA) register during Reset. Refer to the *Brown-out Detector* section for further details.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLE | BOD disabled |
| 0x1 | ENABLE | BOD enabled in continuous mode |
| 0x2 | SAMPLE | BOD enabled in sampled mode |
| 0x3 | ENABLEWAIT | BOD enabled in continuous mode. Execution is halted at wake-up until BOD is running |

**Bits 1:0 – SLEEP[1:0]** BOD Operation Mode in Sleep
The value is loaded into the SLEEP bit field of the BOD Control A (BOD.CTRLA) register during Reset. Refer to the *Brown-out Detector* section for further details.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLE | BOD disabled |
| 0x1 | ENABLE | BOD enabled in continuous mode |
| 0x2 | SAMPLE | BOD enabled in sampled mode |
| 0x3 | - | Reserved |

#### 8.8.2.3 Oscillator Configuration

**Name:** OSCCFG
**Offset:** 0x02
**Default:** 0x00
**Property:** -

The default value given in this fuse description is the factory-programmed value and should not be mistaken for the Reset value.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CLKSEL[3:0] | | | |
| Access | | | | | R | R | R | R |
| Default | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – CLKSEL[3:0]** Clock Select
This bit field controls the default oscillator for the device.

| Value | Name | Description |
|---|---|---|
| 0x0 | OSCHF | Device running on internal high frequency oscillator |
| 0x1 | OSC32K | Device running on internal 32.768 kHz oscillator |
| Other | - | Reserved |

#### 8.8.2.4 System Configuration 0

**Name:** SYSCFG0
**Offset:** 0x05
**Default:** 0xC0
**Property:** -

The default value given in this fuse description is the factory-programmed value and should not be mistaken for the Reset value.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CRCSRC[1:0] | | CRCSEL | | RSTPINCFG | | | EESAVE |
| Access | R | R | R | | R | | | R |
| Default | 1 | 1 | 0 | | 0 | | | 0 |

**Bits 7:6 – CRCSRC[1:0]** CRC Source
This bit field controls which section of the Flash will be checked by the CRCSCAN peripheral after Reset. Refer to the *CRCSCAN* section for more information about the functionality.

| Value | Name | Description |
|---|---|---|
| 0x0 | FLASH | CRC of full Flash (boot, application code, and application data) |
| 0x1 | BOOT | CRC of the Boot section |
| 0x2 | BOOTAPP | CRC of the Application code and Boot sections |
| 0x3 | NOCRC | No CRC |

**Bit 5 – CRCSEL** CRC Mode Selection
This bit controls the type of CRC performed by the CRCSCAN peripheral. Refer to the *CRCSCAN* section for more information about the functionality.

| Value | Name | Description |
|---|---|---|
| 0 | CRC16 | CRC-16-CCITT |
| 1 | CRC32 | CRC-32 (IEEE 802.3) |

**Bit 3 – RSTPINCFG** Reset Pin Configuration at Start-Up
This bit controls the pin configuration for the Reset pin.

| Value | Name | Description |
|---|---|---|
| 0 | INPUT | No external reset |
| 1 | RESET | External reset with pull-up enabled on PF6 |

**Bit 0 – EESAVE** EEPROM Saved During Chip Erase
This bit controls if the EEPROM will be erased or saved during a chip erase. If the device is locked, the EEPROM is always erased by a chip erase regardless of this bit.

| Value | Name | Description |
|---|---|---|
| 0 | DISABLE | EEPROM is erased during a chip erase |
| 1 | ENABLE | EEPROM is saved during a chip erase |

#### 8.8.2.5 System Configuration 1

**Name:** SYSCFG1
**Offset:** 0x06
**Default:** 0x08
**Property:** -

The default value given in this fuse description is the factory-programmed value and should not be mistaken for the Reset value.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | MVSYSCFG[1:0] | | SUT[2:0] | | |
| Access | | | | R | R | R | R | R |
| Default | | | | 0 | 1 | 0 | 0 | 0 |

**Bits 4:3 – MVSYSCFG[1:0]** MVIO System Configuration
This bit field controls the power supply mode.

| Value | Name | Description |
|---|---|---|
| 0x0 | Reserved | - |
| 0x1 | DUAL | Device used in a dual supply configuration |
| 0x2 | SINGLE | Device used in a single supply configuration |
| 0x3 | Reserved | - |

**Bits 2:0 – SUT[2:0]** Start-up Time
This bit field controls the start-up time between power-on and code execution.

| Value | Description |
|---|---|
| 0x0 | 0 ms |
| 0x1 | 1 ms |
| 0x2 | 2 ms |
| 0x3 | 4 ms |
| 0x4 | 8 ms |
| 0x5 | 16 ms |
| 0x6 | 32 ms |
| 0x7 | 64 ms |

#### 8.8.2.6 Code Size

**Name:** CODESIZE
**Offset:** 0x07
**Default:** 0x00
**Property:** -

The default value given in this fuse description is the factory-programmed value, and should not be mistaken for the Reset value.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CODESIZE[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Default | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – CODESIZE[7:0]** Code section size configuration
This bit field defines the combined size of the Boot Code (BOOT) section and Application Code (APPCODE) section in blocks of 512 bytes. For more details, refer to the *Nonvolatile Memory Controller* section.
**Note:** If FUSE.BOOTSIZE is `0x00`, the entire Flash memory is set as the Boot Code section, and the value of FUSE.CODESIZE is not used.

#### 8.8.2.7 Boot Size

**Name:** BOOTSIZE
**Offset:** 0x08
**Default:** 0x00
**Property:** -

The default value given in this fuse description is the factory-programmed value, and should not be mistaken for the Reset value.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BOOTSIZE[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Default | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – BOOTSIZE[7:0]** Boot Section Size
This bitfield controls the size of the boot section in blocks of 512 bytes. A value of `0x00` defines the entire Flash as Boot Code section.
For more details, refer to the *Nonvolatile Memory Controller* section.

## 8.9 LOCK - Memory Sections Access Protection

The device can be locked so that the memories cannot be read using the Unified Program and Debug Interface (UPDI). The locking protects both the Flash (all Boot Code, Application Code, and Application Data sections), SRAM, and the EEPROM including the FUSE data. This prevents the reading of application data or code using the debugger interface. Regular memory access from within the application is still enabled.

The device is locked by writing a non-valid key to the Lock Key (LOCK.KEY) register.

**Table 8-5. Memory Access Unlocked (LOCK.KEY Valid Key)[1]**

| Memory Section | CPU Access | | UPDI Access | |
|---|---|---|---|---|
| | Read | Write | Read | Write |
| Flash | Yes | Yes | Yes | Yes |
| SRAM | Yes | Yes | Yes | Yes |
| EEPROM | Yes | Yes | Yes | Yes |
| USERROW | Yes | Yes | Yes | Yes |
| SIGROW | Yes | No | Yes | No |
| FUSE | Yes | No | Yes | Yes |
| LOCK | Yes | No | Yes | Yes |
| Registers | Yes | Yes | Yes | Yes |

**Table 8-6. Memory Access Locked (LOCK.KEY Invalid Key)[1]**

| Memory Section | CPU Access | | UPDI Access | |
|---|---|---|---|---|
| | Read | Write | Read | Write |
| Flash | Yes | Yes | No | No |
| SRAM | Yes | Yes | No | No |
| EEPROM | Yes | Yes | No | No |
| USERROW | Yes | Yes | No | Yes[2] |

.....continued

| Memory Section | CPU Access | | UPDI Access | |
|---|---|---|---|---|
| | Read | Write | Read | Write |
| SIGROW | Yes | No | No | No |
| FUSE | Yes | No | No | No |
| LOCK | Yes | No | No | No |
| Registers | Yes | Yes | No | No |

**Notes:**

1. Read operations marked No in the tables may appear to be successful, but the data is not valid. Hence, any attempt of code validation through the UPDI will fail on these memory sections.
2. In the Locked mode, the USERROW can be written using the Fuse Write command, but the current USERROW values cannot be read.

**Important:** The only way to unlock a device is to perform a CHIPERASE. No application data is retained.

### 8.9.1 Lock Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | KEY | 7:0 | KEY[7:0] | | | | | | | |
| | | 15:8 | KEY[15:8] | | | | | | | |
| | | 23:16 | KEY[23:16] | | | | | | | |
| | | 31:24 | KEY[31:24] | | | | | | | |

### 8.9.2 Lock Description

#### 8.9.2.1 Lock Key

**Name:** KEY
**Offset:** 0x00
**Reset:** Initial factory value 0x5CC5C55C
**Property:** -

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | KEY[31:24] | | | | | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|
| | KEY[23:16] | | | | | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|
| | KEY[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 |
|---|---|---|---|---|---|---|---|---|
| | KEY[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 31:0 – KEY[31:0]** Lock Key
This bit field controls whether the device is locked or not.

| Value | Name | Description |
|---|---|---|
| `0x5CC5C55C` | UNLOCKED | Device unlocked |
| `Other` | LOCKED | Device locked |

## 8.10 I/O Memory

All AVR128DB28/32/48/64 devices I/O and peripheral registers are located in the I/O memory space. Refer to the *Peripheral Address Map* table for further details.

For compatibility with future devices, if a register containing reserved bits is written, the reserved bits should be written to '`0`'. Reserved I/O memory addresses should never be written.

### 8.10.1 Single-Cycle I/O Registers

The I/O memory ranging from 0x00 to 0x3F can be accessed by a single-cycle CPU instruction using the `IN` or `OUT` instructions.

The peripherals available in the single-cycle I/O registers are as follows:

- VPORTx
  - Refer to the *I/O Configuration* section for further details
- GPR
  - Refer to the *General Purpose Registers* section for further details
- CPU
  - Refer to the *AVR CPU* section for further details

The single-cycle I/O registers ranging from 0x00 to 0x1F (VPORTx and GPR) are also directly bit-accessible using the `SBI` or `CBI` instruction. In these single-cycle I/O registers, single bits can be checked by using the `SBIS` or `SBIC` instruction.

Refer to the *Instruction Set Summary* section for further details.

### 8.10.2 Extended I/O Registers

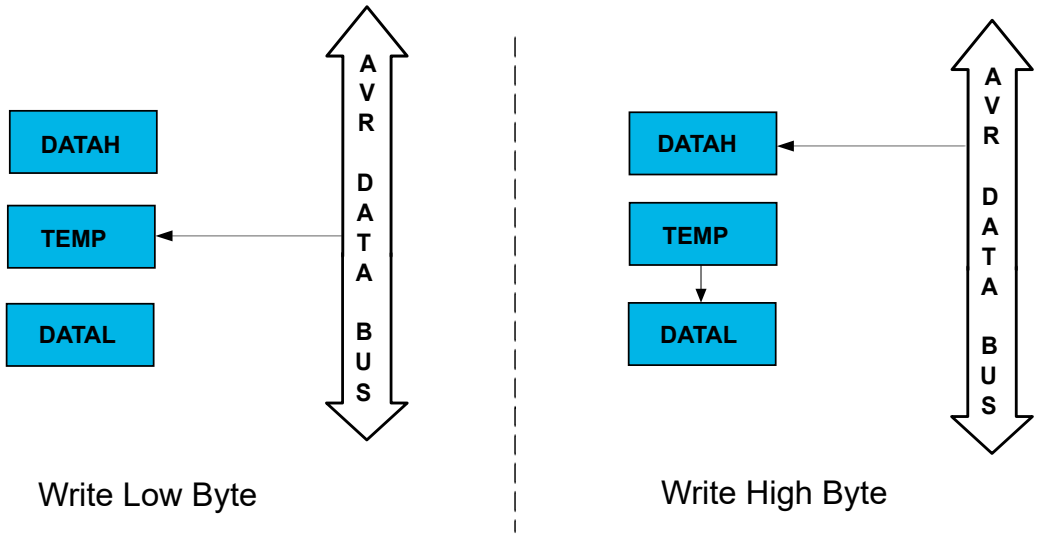
The I/O memory space ranging from 0x0040 to 0x103F can only be accessed by the `LD/LDS/LDD` or `ST/STS/STD` instructions, transferring data between the 32 general purpose working registers (R0-R31) and the I/O memory space.

Refer to the *Peripheral Address Map* table and the *Instruction Set Summary* section for further details.

### 8.10.3 Accessing 16-bit Registers

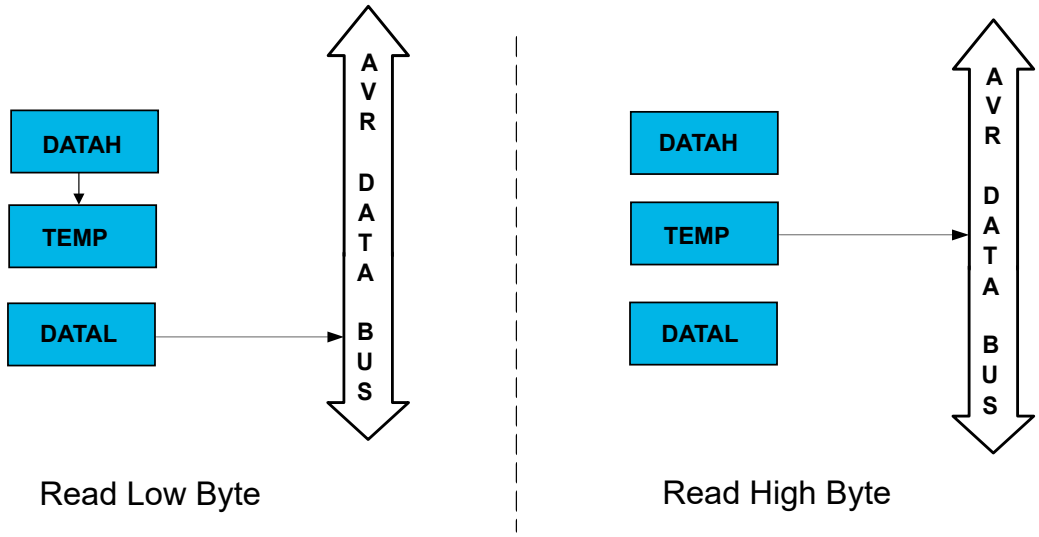
Most of the registers for the AVR128DB28/32/48/64 devices are 8-bit registers, but the devices also feature a few 16-bit registers. As the AVR data bus has a width of 8 bits, accessing the 16-bit requires two read or write operations. All the 16-bit registers of the AVR128DB28/32/48/64 devices are connected to the 8-bit bus through a temporary (TEMP) register.

**Figure 8-2. 16-Bit Register Write Operation**




For a 16-bit write operation, the low byte register (e.g., DATAL) of the 16-bit register must be written before the high byte register (e.g., DATAH). Writing the low byte register will result in a write to the temporary (TEMP) register instead of the low byte register, as shown on the left side of the 16-Bit Register Write Operation figure. When the high byte register of the 16-bit register is written, TEMP will be copied into the low byte of the 16-bit register in the same clock cycle, as shown on the right side of the 16-Bit Register Write Operation figure.

**Figure 8-3. 16-Bit Register Read Operation**




For a 16-bit read operation, the low byte register (e.g., DATAL) of the 16-bit register must be read before the high byte register (e.g., DATAH). When the low byte register is read, the high byte register of the 16-bit register is copied into the temporary (TEMP) register in the same clock cycle, as shown on the left side of the 16-Bit Register Read Operation figure. Reading the high byte register will result in a read from TEMP instead of the high byte register, as shown on right side of the 16-Bit Register Read Operation figure.

The described mechanism ensures that the low and high bytes of 16-bit registers are always accessed simultaneously when reading or writing the registers.

Interrupts can corrupt the timed sequence if an interrupt is triggered during a 16-bit read/write operation and a 16-bit register within the same peripheral is accessed in the interrupt service routine. To prevent this, interrupts should be disabled when writing or reading 16-bit registers. Alternatively, the temporary register can be read before and restored after the 16-bit access in the interrupt service routine.

### 8.10.4 Accessing 24-bit Registers

For 24-bit registers, the read and write access is done in the same way as described for 16-bit registers, except there are two temporary registers for 24-bit registers. The Most Significant Byte must be written last when writing to the register, and the Least Significant Byte must be read first when reading the register.

# 9. GPR - General Purpose Registers

The AVR128DB28/32/48/64 devices provide four General Purpose Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and interrupt flags. No implicit or explicit semantic applies to the bits in the General Purpose Registers. The interpretation of the bit values is completely determined by software.

General Purpose Registers, which reside in the address range 0x001C - 0x001F, are directly bit-accessible using the `SBI`, `CBI`, `SBIS`, and `SBIC` instructions.

## 9.1 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | GPR0 | 7:0 | GPR[7:0] | | | | | | | |
| 0x01 | GPR1 | 7:0 | GPR[7:0] | | | | | | | |
| 0x02 | GPR2 | 7:0 | GPR[7:0] | | | | | | | |
| 0x03 | GPR3 | 7:0 | GPR[7:0] | | | | | | | |

## 9.2 Register Description

### 9.2.1 General Purpose Register n

**Name:** GPRn
**Offset:** 0x00 + n*0x01 [n=0..3]
**Reset:** 0x00
**Property:** -

These are General Purpose Registers that can be used to store data, such as global variables and flags, in the bit accessible I/O memory space.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | GPR[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 – GPR[7:0]** General Purpose Register Byte

# 10. Peripherals and Architecture

## 10.1 Peripheral Address Map

The address map shows the base address for each peripheral. For complete register description and summary for each peripheral, refer to the respective peripheral sections.

**Table 10-1. Peripheral Address Map**

| Base Address | Name | Description | 28-pin | 32-pin | 48-pin | 64-pin |
|---|---|---|---|---|---|---|
| 0x0000 | VPORTA | Virtual Port A | X | X | X | X |
| 0x0004 | VPORTB | Virtual Port B | | | X | X |
| 0x0008 | VPORTC | Virtual Port C | X | X | X | X |
| 0x000C | VPORTD | Virtual Port D | X | X | X | X |
| 0x0010 | VPORTE | Virtual Port E | | | X | X |
| 0x0014 | VPORTF | Virtual Port F | X | X | X | X |
| 0x0018 | VPORTG | Virtual Port G | | | | X |
| 0x001C | GPR | General Purpose Registers | X | X | X | X |
| 0x0030 | CPU | CPU | X | X | X | X |
| 0x0040 | RSTCTRL | Reset Controller | X | X | X | X |
| 0x0050 | SLPCTRL | Sleep Controller | X | X | X | X |
| 0x0060 | CLKCTRL | Clock Controller | X | X | X | X |
| 0x00A0 | BOD | Brown-Out Reset Detector | X | X | X | X |
| 0x00B0 | VREF | Voltage Reference | X | X | X | X |
| 0x00C0 | MVIO | MVIO Controller | X | X | X | X |
| 0x0100 | WDT | Watchdog Timer | X | X | X | X |
| 0x0110 | CPUINT | Interrupt Controller | X | X | X | X |
| 0x0120 | CRCSCAN | Cyclic Redundancy Check Memory Scan | X | X | X | X |
| 0x0140 | RTC | Real Time Counter | X | X | X | X |
| 0x01C0 | CCL | Configurable Custom Logic | X | X | X | X |
| 0x0200 | EVSYS | Event System | X | X | X | X |
| 0x0400 | PORTA | Port A Configuration | X | X | X | X |
| 0x0420 | PORTB | Port B Configuration | | | X | X |
| 0x0440 | PORTC | Port C Configuration | X | X | X | X |
| 0x0460 | PORTD | Port D Configuration | X | X | X | X |
| 0x0480 | PORTE | Port E Configuration | | | X | X |
| 0x04A0 | PORTF | Port F Configuration | X | X | X | X |
| 0x04C0 | PORTG | Port G Configuration | | | | X |

.....continued

| Base Address | Name | Description | 28-pin | 32-pin | 48-pin | 64-pin |
|---|---|---|---|---|---|---|
| 0x05E0 | PORTMUX | Port Multiplexer | X | X | X | X |
| 0x0600 | ADC0 | Analog to Digital Converter 0 | X | X | X | X |
| 0x0680 | AC0 | Analog Comparator 0 | X | X | X | X |
| 0x0688 | AC1 | Analog Comparator 1 | X | X | X | X |
| 0x0690 | AC2 | Analog Comparator 2 | X | X | X | X |
| 0x06A0 | DAC0 | Digital to Analog converter 0 | X | X | X | X |
| 0x06C0 | ZCD0 | Zero Cross Detector 0 | X | X | X | X |
| 0x06C8 | ZCD1 | Zero Cross Detector 1 | | | X | X |
| 0x06D0 | ZCD2 | Zero Cross Detector 2 | | | | X |
| 0x0700 | OPAMP | Analog Signal Conditioning | X | X | X | X |
| 0x0800 | USART0 | Universal Synchronous Asynchronous Receiver and Transmitter 0 | X | X | X | X |
| 0x0820 | USART1 | Universal Synchronous Asynchronous Receiver and Transmitter 1 | X | X | X | X |
| 0x0840 | USART2 | Universal Synchronous Asynchronous Receiver and Transmitter 2 | X | X | X | X |
| 0x0860 | USART3 | Universal Synchronous Asynchronous Receiver and Transmitter 3 | | | X | X |
| 0x0880 | USART4 | Universal Synchronous Asynchronous Receiver and Transmitter 4 | | | X | X |
| 0x08A0 | USART5 | Universal Synchronous Asynchronous Receiver and Transmitter 5 | | | | X |
| 0x0900 | TWI0 | Two-Wire Interface 0 | X | X | X | X |
| 0x0920 | TWI1 | Two-Wire Interface 1 | | X | X | X |
| 0x0940 | SPI0 | Serial Peripheral Interface 0 | X | X | X | X |
| 0x0960 | SPI1 | Serial Peripheral Interface 1 | X | X | X | X |
| 0x0A00 | TCA0 | Timer/Counter Type A 0 | X | X | X | X |
| 0x0A40 | TCA1 | Timer/Counter Type A 1 | | | X | X |
| 0x0B00 | TCB0 | Timer/Counter Type B 0 | X | X | X | X |
| 0x0B10 | TCB1 | Timer/Counter Type B 1 | X | X | X | X |
| 0x0B20 | TCB2 | Timer/Counter Type B 2 | X | X | X | X |
| 0x0B30 | TCB3 | Timer/Counter Type B 3 | | | X | X |
| 0x0B40 | TCB4 | Timer/Counter Type B 4 | | | | X |
| 0x0B80 | TCD0 | Timer/Counter Type D 0 | X | X | X | X |
| 0x0F00 | SYSCFG | System Configuration | X | X | X | X |
| 0x1000 | NVMCTRL | Non Volatile Memory Controller | X | X | X | X |

**Table 10-2. System Memory Address Map**

| Base Address | Name | Description | 28-pin | 32-pin | 48-pin | 64-pin |
|---|---|---|---|---|---|---|
| 0x1040 | LOCK | Lock bits | X | X | X | X |
| 0x1050 | FUSE | User Configuration | X | X | X | X |
| 0x1080 | USERROW | User row | X | X | X | X |
| 0x1100 | SIGROW | Signature row | X | X | X | X |

## 10.2 Interrupt Vector Mapping

Each of the interrupt vectors is connected to one peripheral instance, as shown in the table below. A peripheral can have one or more interrupt sources. For more details on the available interrupt sources, see the *Interrupt* section in the *Functional Description* of the respective peripheral.

An interrupt flag is set in the Interrupt Flags register of the peripheral (peripheral.INTFLAGS) when the interrupt condition occurs, even if the interrupt is not enabled.

An interrupt is enabled or disabled by writing to the corresponding Interrupt Enable bit in the peripheral's Interrupt Control register (*peripheral*.INTCTRL).

An interrupt request is generated when the corresponding interrupt is enabled, and the interrupt flag is set. Interrupts must be enabled globally for interrupt request to be generated. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

**Table 10-3. Interrupt Vector Mapping**

| Vector number | Program Address (word) | Peripheral | Description | 28-pin | 32-pin | 48-pin | 64-pin |
|---|---|---|---|---|---|---|---|
| 0 | 0x00 | RESET | | X | X | X | X |
| 1 | 0x02 | NMI | Non-Maskable Interrupt | X | X | X | X |
| 2 | 0x04 | BOD | Voltage Level Monitor Interrupt | X | X | X | X |
| 3 | 0x06 | CLKCTRL | External crystal oscillator/clock source failure Interrupt (CFD) | X | X | X | X |
| 4 | 0x08 | MVIO | MVIO Interrupt | X | X | X | X |
| 5 | 0x0A | RTC | Overflow or Compare Match Interrupt | X | X | X | X |
| 6 | 0x0C | RTC | Periodic Interrupt (PIT) | X | X | X | X |
| 7 | 0x0E | CCL | Configurable Custom Logic Interrupt | X | X | X | X |
| 8 | 0x10 | PORTA | PORTA External interrupt | X | X | X | X |
| 9 | 0x12 | TCA0 | Normal: Overflow Interrupt<br>Split: Low Underflow Interrupt | X | X | X | X |
| 10 | 0x14 | TCA0 | Normal: Unused<br>Split: High Underflow Interrupt | X | X | X | X |
| 11 | 0x16 | TCA0 | Normal: Compare 0 Interrupt<br>Split: Low Compare 0 Interrupt | X | X | X | X |
| 12 | 0x18 | TCA0 | Normal: Compare 1 Interrupt<br>Split: Low Compare 1 Interrupt | X | X | X | X |

.....continued

| Vector number | Program Address (word) | Peripheral | Description | 28-pin | 32-pin | 48-pin | 64-pin |
|---|---|---|---|---|---|---|---|
| 13 | 0x1A | TCA0 | Normal: Compare 2 Interrupt<br>Split: Low Compare 2 Interrupt | X | X | X | X |
| 14 | 0x1C | TCB0 | Capture/Overflow Interrupt | X | X | X | X |
| 15 | 0x1E | TCB1 | Capture/Overflow Interrupt | X | X | X | X |
| 16 | 0x20 | TCD0 | Overflow (OVF) Interrupt | X | X | X | X |
| 17 | 0x22 | TCD0 | Trigger (TRIG) Interrupt | X | X | X | X |
| 18 | 0x24 | TWI0 | Slave (TWIS) Interrupt | X | X | X | X |
| 19 | 0x26 | TWI0 | Master (TWIM) Interrupt | X | X | X | X |
| 20 | 0x28 | SPI0 | SPI0 Interrupt | X | X | X | X |
| 21 | 0x2A | USART0 | Receive Complete (RXC) Interrupt | X | X | X | X |
| 22 | 0x2C | USART0 | Data Register Empty (DRE) Interrupt | X | X | X | X |
| 23 | 0x2E | USART0 | Transmit Complete (TXC) Interrupt | X | X | X | X |
| 24 | 0x30 | PORTD | External Interrupt | X | X | X | X |
| 25 | 0x32 | AC0 | Compare Interrupt | X | X | X | X |
| 26 | 0x34 | ADC0 | Result Ready (RESRDY) Interrupt | X | X | X | X |
| 27 | 0x36 | ADC0 | Window Compare (WCOMP) Interrupt | X | X | X | X |
| 28 | 0x38 | ZCD0 | Zero Cross Interrupt | X | X | X | X |
| 29 | 0x3A | AC1 | Compare Interrupt | X | X | X | X |
| 30 | 0x3C | PORTC | External Interrupt | X | X | X | X |
| 31 | 0x3E | TCB2 | Capture/Overflow Interrupt | X | X | X | X |
| 32 | 0x40 | USART1 | Receive Complete (RXC) Interrupt | X | X | X | X |
| 33 | 0x42 | USART1 | Data Register Empty (DRE) Interrupt | X | X | X | X |
| 34 | 0x44 | USART1 | Transmit Complete (TXC) Interrupt | X | X | X | X |
| 35 | 0x46 | PORTF | External Interrupt | X | X | X | X |
| 36 | 0x48 | NVMCTRL | EEPROM Ready (EEREADY) Interrupt | X | X | X | X |
| 37 | 0x4A | SPI1 | SPI1 Interrupt | X | X | X | X |
| 38 | 0x4C | USART2 | Receive Complete (RXC) Interrupt | X | X | X | X |
| 39 | 0x4E | USART2 | Data Register Empty (DRE) Interrupt | X | X | X | X |
| 40 | 0x50 | USART2 | Transmit Complete (TXC) Interrupt | X | X | X | X |
| 41 | 0x52 | AC2 | Compare Interrupt | X | X | X | X |
| 42 | 0x54 | TWI1 | Slave (TWIS) Interrupt | | X | X | X |
| 43 | 0x56 | TWI1 | Master (TWIM) Interrupt | | X | X | X |
| 44 | 0x58 | TCB3 | Capture Interrupt | | | X | X |
| 45 | 0x5A | PORTB | External Interrupt | | | X | X |

.....continued

| Vector number | Program Address (word) | Peripheral | Description | 28-pin | 32-pin | 48-pin | 64-pin |
|---|---|---|---|---|---|---|---|
| 46 | 0x5C | PORTE | External Interrupt | | | X | X |
| 47 | 0x5E | TCA1 | Normal: Overflow Interrupt<br>Split: Low Underflow Interrupt | | | X | X |
| 48 | 0x60 | TCA1 | Normal: Unused Interrupt<br>Split: High Underflow Interrupt | | | X | X |
| 49 | 0x62 | TCA1 | Normal: Compare 0 Interrupt<br>Split: Low Compare 0 Interrupt | | | X | X |
| 50 | 0x64 | TCA1 | Normal: Compare 1 Interrupt<br>Split: Low Compare 1 Interrupt | | | X | X |
| 51 | 0x66 | TCA1 | Normal: Compare 2 Interrupt<br>Split: Low Compare 2 Interrupt | | | X | X |
| 52 | 0x68 | ZCD1 | Zero Cross Interrupt | | | X | X |
| 53 | 0x6A | USART3 | Receive Complete (RXD) Interrupt | | | X | X |
| 54 | 0x6C | USART3 | Data Register Empty (DRE) Interrupt | | | X | X |
| 55 | 0x6E | USART3 | Transmit Complete (TXD) Interrupt | | | X | X |
| 56 | 0x70 | USART4 | Receive Complete (RXD) Interrupt | | | X | X |
| 57 | 0x72 | USART4 | Data Register Empty (DRE) Interrupt | | | X | X |
| 58 | 0x74 | USART4 | Transmit Complete (TXD) Interrupt | | | X | X |
| 59 | 0x76 | PORTG | External Interrupt | | | | X |
| 60 | 0x78 | ZCD2 | Zero Cross Interrupt | | | | X |
| 61 | 0x7A | TCB4 | Capture/Overflow Interrupt | | | | X |
| 62 | 0x7C | USART5 | Receive Complete (RXC) Interrupt | | | | X |
| 63 | 0x7E | USART5 | Data Register Empty (DRE) Interrupt | | | | X |
| 64 | 0x80 | USART5 | Transmit Complete (TXD) Interrupt | | | | X |

## 10.3 SYSCFG - System Configuration

The system configuration contains the revision ID of the part. The revision ID is readable from the CPU, making it useful for implementing application changes between part revisions.

### 10.3.1 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | Reserved | | | | | | | | | |
| 0x01 | REVID | 7:0 | MAJOR[3:0] | | | | MINOR[3:0] | | | |

### 10.3.2 Register Description

#### 10.3.2.1 Device Revision ID Register

**Name:** REVID
**Offset:** 0x01
**Reset:** [revision ID]
**Property:** -

This register is read only and give the device revision ID.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MAJOR[3:0] | | | | MINOR[3:0] | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:4 – MAJOR[3:0]** Major revision
This bit field contains the major revision for the device. 0x00 = A, 0x01 = B, and so on.

**Bits 3:0 – MINOR[3:0]** Minor revision
This bit field contains the minor revision for the device. 0x00 = 0, 0x01 = 1, and so on.

# 11. NVMCTRL - Nonvolatile Memory Controller

## 11.1 Features

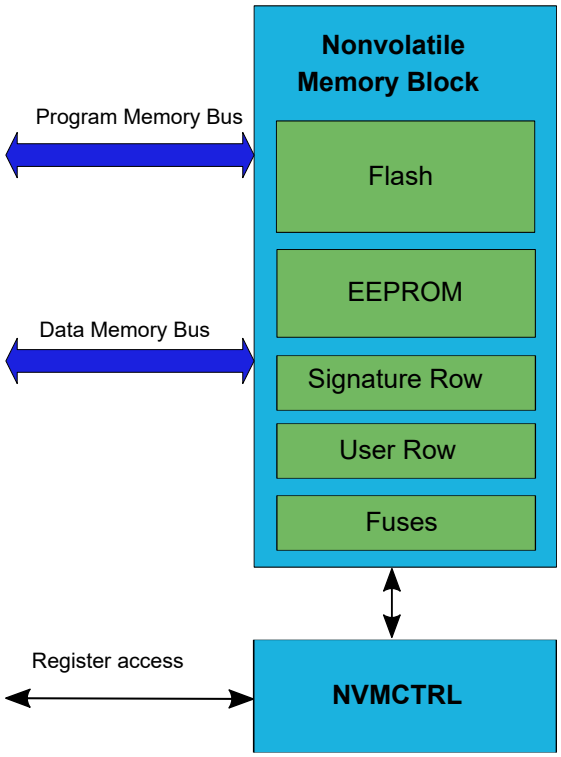
- In-System Programmable
- Self-Programming and Boot Loader Support
- Configurable Memory Sections:
  - Boot loader code section
  - Application code section
  - Application data section
- Signature Row for Factory-Programmed Data:
  - ID for each device type
  - Serial number for each device
  - Calibration bytes for factory-calibrated peripherals
- User Row for Application Data:
  - Can be read and written from software
  - Can be written from the UPDI on a locked device
  - Content is kept after chip erase

## 11.2 Overview

The NVM Controller (NVMCTRL) is the interface between the CPU and Nonvolatile Memories (Flash, EEPROM, Signature Row, User Row and fuses). These are reprogrammable memory blocks that retain their values even when they are not powered. The Flash is mainly used for program storage and can also be used for data storage. The EEPROM is used for data storage and can be programmed while the CPU is running the program from the Flash.

### 11.2.1 Block Diagram

**Figure 11-1. NVMCTRL Block Diagram**




## 11.3 Functional Description

### 11.3.1 Memory Organization

#### 11.3.1.1 Flash

The Flash is divided into a set of pages. A page is the smallest addressable unit when erasing the Flash. It is only possible to erase an entire page or multiple pages at a time. Writes can be done per byte or word. One page consists of 512 bytes.

The Flash can be divided into three sections, each consisting of a variable number of pages. These sections are:

**Boot Loader Code (BOOT) Section**
The Flash section with full write access. Boot loader software must be placed in this section if used.

**Application Code (APPCODE) Section**
The Flash section with limited write access. An executable application code is usually placed in this section.

**Application Data (APPDATA) Section**
The Flash section without write access. Parameters are usually placed in this section.

**Inter-Section Write Protection**
For security reasons, it is not possible to write to the section of Flash the code is currently executing from. Code writing to the APPCODE section needs to be executed from the BOOT section, and code writing to the APPDATA section needs to be executed from either the BOOT section or the APPCODE section.

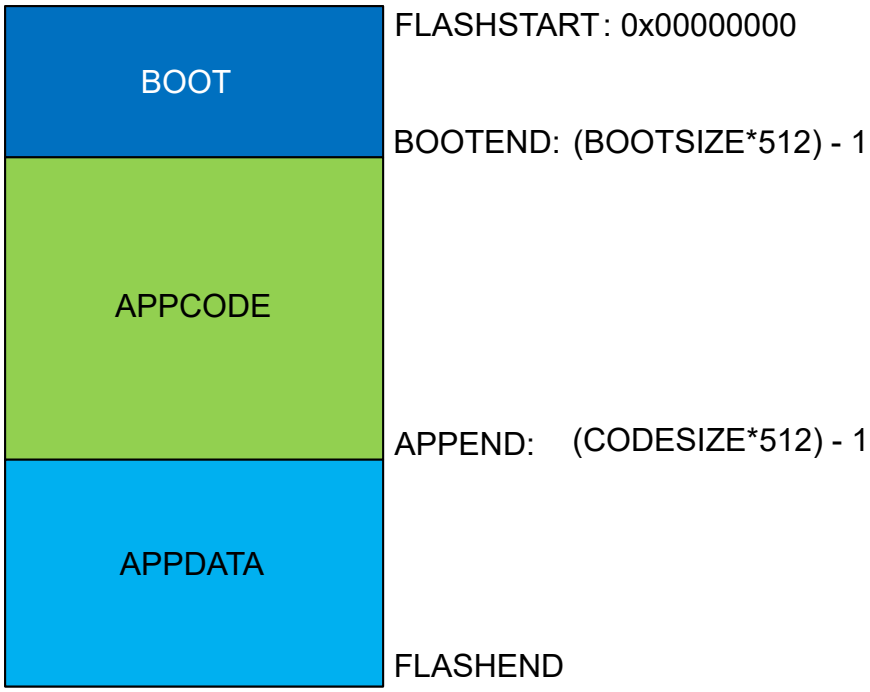
**Table 11-1. Write Protection for Self-Programming**

| Program Execution Section | Section Being Addressed | Programming Allowed? | CPU Halted? |
|---|---|---|---|
| BOOT | BOOT | No | - |
| | APPCODE | Yes | Yes |
| | APPDATA | | Yes |
| | EEPROM | | No |
| APPCODE | BOOT | No | - |
| | APPCODE | | |
| | APPDATA | Yes | Yes |
| | EEPROM | | No |
| APPDATA | BOOT | No | - |
| | APPCODE | | |
| | APPDATA | | |
| | EEPROM | | |

**Section Sizes**

The sizes of these sections are set by the Boot Size (FUSE.BOOTSIZE) fuse and the Code Size (FUSE.CODESIZE) fuse. The fuses select the section sizes in blocks of 512 bytes. The BOOT section stretches from FLASHSTART to BOOTEND. The APPCODE section spreads from BOOTEND until APPEND. The remaining area is the APPDATA section.

**Figure 11-2. Flash Sections Sizes and Locations**




If FUSE.BOOTSIZE is written to '0', the entire Flash is regarded as the BOOT section. If FUSE.CODESIZE is written to '0' and FUSE.BOOTSIZE > 0, the APPCODE section runs from BOOTEND to the end of Flash (no APPDATA section).

When FUSE.CODESIZE ≤ FUSE.BOOTSIZE, the APPCODE section is removed, and the APPDATA runs from BOOTEND to the end of Flash.

**Table 11-2. Setting Up Flash Sections**

| BOOTSIZE | CODESIZE | BOOT Section | APPCODE Section | APPDATA Section |
|---|---|---|---|---|
| 0 | - | 0 to FLASHEND | - | - |
| > 0 | 0 | 0 to BOOTEND | BOOTEND to FLASHEND | - |
| > 0 | ≤ BOOTSIZE | 0 to BOOTEND | - | BOOTEND to FLASHEND |
| > 0 | > BOOTSIZE | 0 to BOOTEND | BOOTEND to APPEND | APPEND to FLASHEND |

If there is no boot loader software, it is recommended to use the BOOT section for application code.

**Notes:**

1. After Reset, the default vector table location is at the start of the APPCODE section. The peripheral interrupts can be used in the code running in the BOOT section by relocating the interrupt vector table at the start of this section. That is done by setting the IVSEL bit in the CPUINT.CTRLA register. Refer to the *CPUINT* section for details.
2. If BOOTEND/APPEND, as resulted from BOOTSIZE/CODESIZE fuse setting, exceed the device FLASHEND, the corresponding fuse setting is ignored, and the default value is used. Refer to “Fuse” in the *Memories* section for default values.

**Example 11-1. Size of Flash Sections Example**

If FUSE.BOOTSIZE is written to `0x04` and FUSE.CODESIZE is written to `0x08`, the first 4*512 bytes will be BOOT, the next 4*512 bytes will be APPCODE, and the remaining Flash will be APPDATA.

**Flash Protection**

Additional to the inter-section write protection, the NVMCTRL provides a security mechanism to avoid unwanted access to the Flash memory sections. Even if the CPU can never write to the BOOT section, a Boot Section Read Protection (BOOTRP) bit in the Control B (NVMCTRL.CTRLB) register is provided to prevent the read and execution of code from the BOOT section. This bit can be set only from the code executed in the BOOT section and has effect only when leaving the BOOT section.

There are two other write protection (APPCODEWP and APPDATAWP) bits in the Control B (NVMCTRL.CTRLB) register that can be set to prevent further updates of the respective Application Code and Application Data sections.

#### 11.3.1.2 EEPROM

The EEPROM is a 512 bytes nonvolatile memory section that has byte granularity on erase/write. It can be erased in blocks of 1/2/4/8/16/32 bytes, but writes are done only one byte at a time. It also has an option to do a byte erase and write in one operation.

#### 11.3.1.3 Signature Row

The Signature Row contains a device ID that identifies each microcontroller device type and a serial number for each manufactured device. The serial number consists of the production lot number, wafer number, and wafer coordinates for the device. The Signature Row cannot be written or erased, but it can be read by the CPU or through the UPDI interface.

#### 11.3.1.4 User Row

The User Row is 32 bytes. This section can be used to store various data, such as calibration/configuration data and serial numbers. This section is not erased by a chip erase.

The User Row section can be read or written from the CPU. This section can be read from UPDI on a device unlocked and can be written through UPDI even on a device locked.

#### 11.3.1.5 Fuses

The fuses contain device configuration values and are copied to their respective target registers at the end of the start-up sequence.

The fuses can be read by the CPU or the UPDI, but can only be programmed or cleared by the UPDI.

### 11.3.2 Memory Access

For read/write operations, the Flash memory can be accessed either from the code space or from the CPU data space. When the code space is used, the Flash is accessible through the `LPM` and `SPM` instructions.

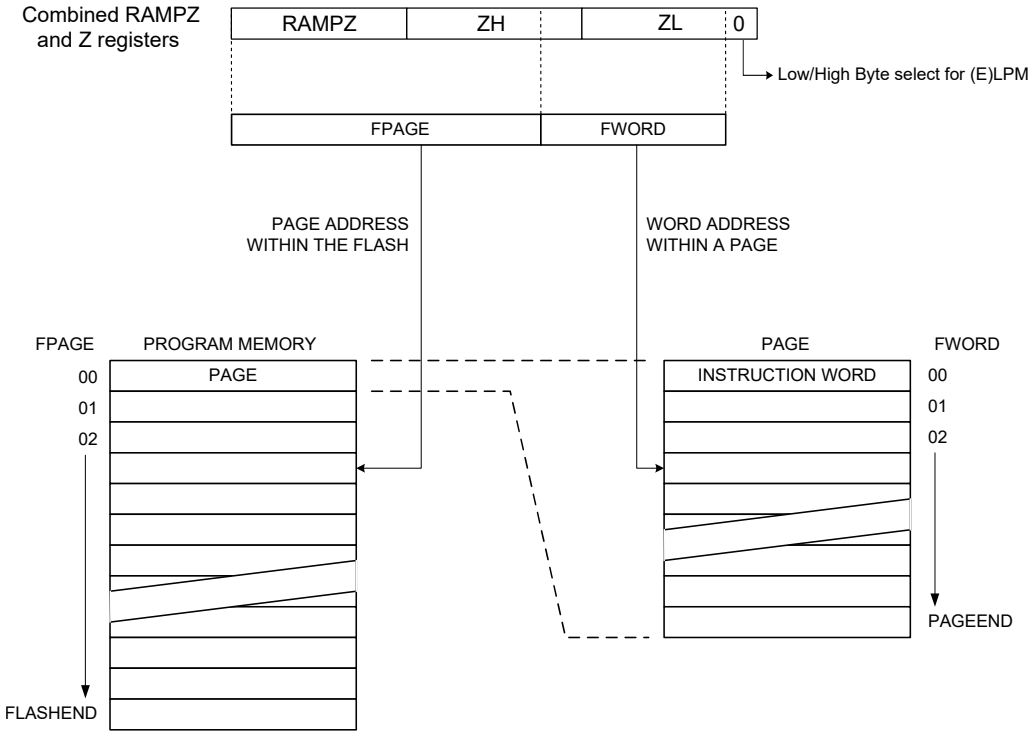
Additionally, the Flash memory is byte accessible when accessed through the CPU data space. This means that it shares the same address space and instructions as SRAM, EEPROM, and I/O registers and is accessible using `LD/ST` instructions in assembly.

For the `LPM` and `SPM` instructions, address 0x0000 is the start of the Flash, but for `LD` and `ST`, it is 0x8000, as shown in the *Memory map* section.

**Addressing Flash Memory in Code Space**

For read and write access to the Flash memory in the code space, the RAMPZ register concatenated with the Z register to create the Address Pointer that is used for `LPM`/`SPM` access.

**Figure 11-3. Flash Addressing for Self-Programming**




The Flash is word-accessed and organized in pages, so the Address Pointer can be treated as having two sections. This is shown in Figure 11-3. The word address in the page (FWORD) is held by the least significant bits in the Address Pointer, while the most significant bits in the Address Pointer hold the Flash page address (FPAGE). Together, FWORD and FPAGE hold an absolute address to a word in the Flash.

The Flash is word-accessed for code space write operations, so the least significant bit (bit 0) in the Address Pointer is ignored.

For Flash read operations, one byte is read at a time. For this, the least significant bit (bit 0) in the Address Pointer is used to select the low byte or high byte in the word address. If this bit is '`0`', the low byte is read, and if this bit is '`1`', the high byte is read.

Once a programming operation is initiated, the address is latched, and the Address Pointer can be updated and used for other operations.

**Addressing Flash in CPU Data Space**

The Flash area in data space has only 32 KB. For devices with Flash memory size greater than 32 KB, the Flash memory is divided into blocks of 32 KB. Those blocks are mapped into data space using the FLMAP bit field of the NVMCTRL.CTRLB register.

For read and write access to the Flash memory in the CPU data space, the `LD`/`ST` instructions are used to access one byte at a time.

#### 11.3.2.1 Read

Reading the Flash is done using Load Program Memory (`LPM`) instructions or load type (`LD*`) instructions with an address according to the memory map. Reading the EEPROM and Signature Row is done using load type instructions (`LD*`). Performing a read operation while a write or erase is in progress will result in a bus wait, and the instruction will be suspended until the ongoing operation is complete.

#### 11.3.2.2 Programming

The Flash programming is done by writing one byte or one word at a time. Writing from the CPU using store type instructions (`ST*`) will write one byte at a time, while a write with the Store Program Memory (`SPM`) instruction will write one word at a time.

The NVMCTRL command set supports multiple Flash erase operations. Up to 32 pages can be erased at the same time. The duration of the erase operation is independent of the number of pages being erased.

The EEPROM erasing has byte granularity with the possibility of erasing up to 32 bytes in one operation. The EEPROM is written one byte at a time, and it has an option to do the erase and write of one byte in the same operation.

The User Row is erased/written as a normal Flash. When the erasing operation is used, the entire User Row is erased at once. The User Row writing has byte granularity.

The Fuse programming is identical to the EEPROM programming, but it can be performed only via the UPDI interface.

**Table 11-3. Programming Granularity**

| Memory Section | Erase Granularity | Write Granularity |
|---|---|---|
| Flash array | Page | Word(1) |
| EEPROM array | Byte | Byte |
| User Row | Page(2) | Byte(3) |
| Fuses | Byte | Byte |

**Notes:**

1. Byte granularity when writing to the CPU data space memory mapped section.
2. One page is 32 bytes.
3. Page granularity when programming from UPDI on a locked device.

#### 11.3.2.3 Command Modes

Reading of the memory arrays is handled using the `LD*`/`LPM`(1) instructions.

The erase of the whole Flash (`CHER`) or the EEPROM (`EECHER`) is started by writing commands to the NVMCTRL.CTRLA register. The other write/erase operations are just enabled by writing commands to the NVMCTRL.CTRLA register and must be followed by writes using `ST*`/`SPM`(1) instructions to the memory arrays.

**Note:**

1. `LPM`/`SPM` cannot be used for EEPROM.

To write a command in the NVMCTRL.CTRLA register, the following sequence needs to be executed:

1. Confirm that any previous operation is completed by reading the Busy (EEBUSY and FBUSY) flags in the NVMCTRL.STATUS register.

2. Write the appropriate key to the Configuration Change Protection (CPU.CCP) register to unlock the NVM Control A (NVMCTRL.CTRLA) register.
3. Write the desired command value to the CMD bit field in the Control A (NVMCTRL.CTRLA) register within the next four instructions.

To perform a write/erase operation in the NVM, the following steps are required:

1. Confirm that any previous operation is completed by reading the Busy (EEBUSY and FBUSY) flags in the NVMCTRL.STATUS register.
2. Optional: If the Flash is accessed in the CPU data space, map the corresponding 32 KB Flash section into the data space by writing the FLMAP bit field in the NVMCTRL.CTRLB register.
3. Write the desired command value to the NVMCTRL.CTRLA register as described before.
4. Write to the correct address in the data space/code space using the `ST*`/`SPM` instructions.
5. Optional: If multiple write operations are required, go to step 4.
6. Write a `NOOP` or `NOCMD` command to the NVMCTRL.CTRLA register to clear the current command.

#### 11.3.2.3.1 Flash Write Mode

The Flash Write (FLWR) mode of the Flash controller enables writes to the Flash array to start a programming operation. Several writes can be done while the FLWR mode is enabled in the NVMCTRL.CTRLA register. When the FLWR mode is enabled, the `ST*` instructions write one byte at a time, while the `SPM` instruction writes one word at a time.

Before a write is performed to an address, its content needs to be erased.

#### 11.3.2.3.2 Flash Page Erase Mode

The Flash Page Erase (FLPER) mode will allow each write to the memory array to erase a page.

An erase operation to the Flash will halt the CPU.

#### 11.3.2.3.3 Flash Multi-Page Erase Mode

The Multi-Page Erase (FLMPERn) mode will allow each write to the memory array to erase multiple pages. When enabling FLMPERn, it is possible to select between erasing two, four, eight, 16, or 32 pages.

The LSbs of the page address are ignored when defining which Flash pages are erased. Using FLMPER4 as an example, erasing any page in the `0x08` - `0x0B` range will cause the erase of all pages in the range.

**Table 11-4. Flash Multi-Page Erase**

| CMD | Pages Erased | Description |
|---|---|---|
| FLMPER2 | 2 | Pages matching FPAGE[N:1] are erased. The value in FPAGE[0] is ignored. |
| FLMPER4 | 4 | Pages matching FPAGE[N:2] are erased. The value in FPAGE[1:0] is ignored. |
| FLMPER8 | 8 | Pages matching FPAGE[N:3] are erased. The value in FPAGE[2:0] is ignored. |
| FLMPER16 | 16 | Pages matching FPAGE[N:4] are erased. The value in FPAGE[3:0] is ignored. |
| FLMPER32 | 32 | Pages matching FPAGE[N:5] are erased. The value in FPAGE[4:0] is ignored. |

**Note:** FPAGE is the page number when doing a Flash erase. Refer to Figure 11-3 for details.

#### 11.3.2.3.4 EEPROM Write Mode

The EEPROM Write (EEWR) mode enables the EEPROM array for writing operations. Several writes can be done while the EEWR mode is enabled in the NVMCTRL.CTRLA register. When the EEWR mode is enabled, writes with the `ST*` instructions will be performed one byte at a time.

When writing the EEPROM, the CPU will continue executing code. If a new load/store operation is started before the EEPROM erase/write is completed, the CPU will be halted.

Before a write is performed to an address, its content needs to be erased.

#### 11.3.2.3.5 EEPROM Erase/Write Mode

The EEPROM Erase/Write (EEERWR) mode enables the EEPROM array for the erase operation directly followed by a write operation. Several erase/writes can be done while the EEERWR mode is enabled in the NVMCTRL.CTRLA register. When the EEERWR mode is enabled, writes with the `ST*` instructions are performed one byte at a time.

When writing/erasing the EEPROM, the CPU will continue executing code.

If a new load or store instruction is started before the erase/write is completed, the CPU will be halted.

#### 11.3.2.3.6 EEPROM Byte Erase Mode

The EEPROM Byte Erase (EEBER) mode will allow each write to the memory array to erase the selected byte. An erased byte always reads back `0xFF`, regardless of the value written to the EEPROM address.

When erasing the EEPROM, the CPU can continue to run from the Flash. If the CPU starts an erase or write operation while the EEPROM is busy, the CPU will be halted until the previous operation is finished.

#### 11.3.2.3.7 EEPROM Multi-Byte Erase Mode

The EEPROM Multi-Byte Erase (EEMBERn) mode allows erasing several bytes in one operation. When enabling the EEMBERn mode, it is possible to select between erasing two, four, eight, 16, or 32 bytes in one operation.

The LSbs of the address are ignored when defining which EEPROM locations are erased. For example, while doing an 8-byte erase, addressing any byte in the `0x18` - `0x1F` range will result in erasing the entire range of bytes.

**Table 11-5. EEPROM Multi-Byte Erase**

| CMD | Bytes Erased | Description(1) |
|---|---|---|
| EEMBER2 | 2 | Addresses matching ADDR[N:1] are erased. The value in ADDR[0] is ignored. |
| EEMBER4 | 4 | Addresses matching ADDR[N:2] are erased. The value in ADDR[1:0] is ignored. |
| EEMBER8 | 8 | Addresses matching ADDR[N:3] are erased. The value in ADDR[2:0] is ignored. |
| EEMBER16 | 16 | Addresses matching ADDR[N:4] are erased. The value in ADDR[3:0] is ignored. |
| EEMBER32 | 32 | Addresses matching ADDR[N:5] are erased. The value in ADDR[4:0] is ignored. |

**Note:** ADDR is the address written when doing an EEPROM erase.

When erasing the EEPROM, the CPU can continue to execute instructions from the Flash. If the CPU starts an erase or write operation while the EEPROM is busy, the NVMCTRL module will give a wait on the bus, and the CPU will be halted until the current operation is finished.

#### 11.3.2.3.8 Chip Erase Command

The Chip Erase (`CHER`) command erases the Flash and the EEPROM. The EEPROM is unaltered if the EEPROM Save During Chip Erase (EESAVE) fuse in FUSE.SYSCFG0 is set.

If the device is locked, the EEPROM is always erased by a chip erase regardless of the EESAVE bit. The read/write protection (BOOTRP, APPCODEWP, APPDATAWP) bits in NVMCTRL.CTRLB do not prevent the operation. All Flash and EEPROM bytes will read back `0xFF` after this command.

This command can only be started from the UPDI.

#### 11.3.2.3.9 EEPROM Erase Command

The EEPROM Erase (`EECHER`) command erases the EEPROM. All EEPROM bytes will read back `0xFF` after the operation. The CPU will be halted while the EEPROM is being erased.

### 11.3.3 Preventing Flash/EEPROM Corruption

A Flash/EEPROM write or erase can cause memory corruption if the supply voltage is too low for the CPU and the Flash/EEPROM to operate properly. These issues are the same on-board level systems using Flash/EEPROM, and it is recommended to use the internal or an external Brown-Out Detector (BOD) to ensure that the device is not operating at too low voltage.

When the voltage is too low, a Flash/EEPROM corruption may be caused by two circumstances:

1. A regular write sequence to the Flash, which requires a minimum voltage to operate correctly.

2. The CPU itself can execute instructions incorrectly when the supply voltage is too low.

The chip erase does not clear fuses. If the BOD is enabled before starting the Chip Erase command, it is automatically enabled at its previous configured level during the chip erase.

Refer to the *Electrical Characteristics* section for Maximum Frequency vs. $V_{DD}$.

**Attention:** Flash/EEPROM corruption can be avoided by taking the following measures:

1. Keep the device in Reset during periods of insufficient power supply voltage. This can be done by enabling the internal BOD.
2. The Voltage Level Monitor (VLM) in the BOD can be used to prevent starting a write to the EEPROM close to the BOD level.
3. If the detection levels of the internal BOD do not match the required detection level, an external low $V_{DD}$ Reset protection circuit can be used. If a Reset occurs while a write operation is ongoing, the write operation will be aborted.

### 11.3.4 Interrupts

**Table 11-6. Available Interrupt Vectors and Sources**

| Offset | Name | Vector Description | Conditions |
|---|---|---|---|
| 0x00 | EEREADY | NVM | The EEPROM is ready for new write/erase operations. |

When an interrupt condition occurs, the corresponding interrupt flag is set in the Interrupt Flags (NVMCTRL.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding bit in the Interrupt Control (NVMCTRL.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the NVMCTRL.INTFLAGS register for details on how to clear interrupt flags.

### 11.3.5 Sleep Mode Operation

If there is no ongoing EEPROM write/erase operation, the NVMCTRL will enter sleep mode, when the system enters sleep mode.

If an EEPROM write/erase operation is ongoing when the system enters a sleep mode, the NVM block, the NVMCTRL and the peripheral clock will remain ON until the operation is finished and will be automatically turned off once the operation is completed. This is valid for all sleep modes, including Power-Down.

The EEPROM Ready interrupt will wake up the device only from Idle sleep mode.

### 11.3.6 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a certain key must first be written to the CPU.CCP register, followed by a write access to the protected bits within four CPU instructions.

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged.

The following registers are under CCP:

**Table 11-7. NVMCTRL - Registers under Configuration Change Protection**

| Register | Key |
|---|---|
| NVMCTRL.CTRLA | SPM |
| NVMCTRL.CTRLB | IOREG |

## 11.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | CMD[6:0] | | | | | | |
| 0x01 | CTRLB | 7:0 | FLMAPLOCK | | FLMAP[1:0] | | | APPDATAWP | BOOTRP | APPCODEWP |
| 0x02 | STATUS | 7:0 | | ERROR[2:0] | | | | | EEBUSY | FBUSY |
| 0x03 | INTCTRL | 7:0 | | | | | | | | EEREADY |
| 0x04 | INTFLAGS | 7:0 | | | | | | | | EEREADY |
| 0x05 | Reserved | | | | | | | | | |
| 0x06 | DATA | 7:0 | DATA[7:0] | | | | | | | |
| | | 15:8 | DATA[15:8] | | | | | | | |
| 0x08 | ADDR | 7:0 | ADDR[7:0] | | | | | | | |
| | | 15:8 | ADDR[15:8] | | | | | | | |
| | | 23:16 | ADDR[23:16] | | | | | | | |

## 11.5 Register Description

### 11.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** Configuration Change Protection

| 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 6:0 – CMD[6:0]** Command
Write this bit field to enable or issue a command. The Chip Erase and EEPROM Erase commands are started when the command is written. The others enable an erase or write operation. The operation is started by doing a store instruction to an address location.
A change from one command to another must always go through No command (`NOCMD`) or No operation (`NOOP`) command to avoid the Command Collision error being set in the ERROR bit field from the NVMCTRL.STATUS register.

| Value | Name | Description |
|---|---|---|
| 0x00 | NOCMD | No command |
| 0x01 | NOOP | No operation |
| 0x02 | FLWR | Flash Write Enable |
| 0x08 | FLPER | Flash Page Erase Enable |
| 0x09 | FLMPER2 | Flash 2-page Erase Enable |
| 0x0A | FLMPER4 | Flash 4-page Erase Enable |
| 0x0B | FLMPER8 | Flash 8-page Erase Enable |
| 0x0C | FLMPER16 | Flash 16-page Erase Enable |
| 0x0D | FLMPER32 | Flash 32-page Erase Enable |
| 0x12 | EEWR | EEPROM Write Enable |
| 0x13 | EEERWR | EEPROM Erase and Write Enable |
| 0x18 | EEBER | EEPROM Byte Erase Enable |
| 0x19 | EEMBER2 | EEPROM 2-byte Erase Enable |
| 0x1A | EEMBER4 | EEPROM 4-byte Erase Enable |
| 0x1B | EEMBER8 | EEPROM 8-byte Erase Enable |
| 0x1C | EEMBER16 | EEPROM 16-byte Erase Enable |
| 0x1D | EEMBER32 | EEPROM 32-byte Erase Enable |
| 0x20 | CHER | Erase Flash and EEPROM. EEPROM is skipped if EESAVE fuse is set. (UPDI access only.) |
| 0x30 | EECHER | Erase EEPROM |
| Other | - | Reserved |

### 11.5.2 Control B

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x30
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FLMAPLOCK | | FLMAP[1:0] | | | APPDATAWP | BOOTRP | APPCODEWP |
| Access | R/W | | R/W | R/W | | R/W | R/W | R/W |
| Reset | 0 | | 1 | 1 | | 0 | 0 | 0 |

**Bit 7 – FLMAPLOCK** Flash Mapping Lock
Setting this bit to '1' prevents further updates of FLMAP[1:0]. This bit can only be cleared by a Reset.

**Bits 5:4 – FLMAP[1:0]** Flash Section Mapped into Data Space
Select what part (in blocks of 32 KB) of the Flash will be mapped as part of the CPU data space and will be accessible through `LD/ST` instructions.
This bit field is not under Configuration Change Protection.
Description: Flash mapping

| Value | Name | Mapped Flash Section (128 KB) |
|---|---|---|
| 0x0 | SECTION0 | 0-32 |
| 0x1 | SECTION1 | 32-64 |
| 0x2 | SECTION2 | 64-96 |
| 0x3 | SECTION3 | 96-128 |

**Bit 2 – APPDATAWP** Application Data Section Write Protection
Writing this bit to '1' prevents further updates to the Application Data section. This bit can only be cleared by a Reset.

**Bit 1 – BOOTRP** Boot Section Read Protection
Writing this bit to '1' will protect the BOOT section from reading and instruction fetching. If a read is issued from the other Flash sections, it will return '0'. An instruction fetch from the BOOT section will return a `NOP` instruction. This bit can only be written from the BOOT section, and it can only be cleared by a Reset. The read protection will only take effect when leaving the BOOT section after the bit is written.

**Bit 0 – APPCODEWP** Application Code Section Write Protection
Writing this bit to '1' prevents further updates to the Application Code section. This bit can only be cleared by a Reset.

### 11.5.3 Status

**Name:** STATUS
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | ERROR[2:0] | | | | | EEBUSY | FBUSY |
| Access | | R/W | R/W | R/W | | | R | R |
| Reset | | 0 | 0 | 0 | | | 0 | 0 |

**Bits 6:4 – ERROR[2:0]** Error Code
The Error Code bit field will show the last error occurring. This bit field can be cleared by writing it to '0'.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No error |
| 0x1 | INVALIDCMD | The selected command is not supported |
| 0x2 | WRITEPROTECT | Attempt to write a section that is protected |
| 0x3 | CMDCOLLISION | A new write/erase command was selected while a write/erase command is already ongoing |
| Other | — | Reserved |

**Bit 1 – EEBUSY** EEPROM Busy
This bit will read '1' when an EEPROM programming operation is ongoing.

**Bit 0 – FBUSY** Flash Busy
This bit will read '1' when a Flash programming operation is ongoing.

### 11.5.4 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

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

**Bit 0 – EEREADY** EEPROM Ready Interrupt
Writing a '1' to this bit enables the interrupt which indicates that the EEPROM is ready for new write/erase operations.
This is a level interrupt that will be triggered only when the EEREADY bit in the INTFLAGS register is set to '1'. The interrupt must not be enabled before triggering an EEPROM write/erase operation, as the EEREADY bit will not be cleared before this command is issued. The interrupt must be disabled in the interrupt handler.

### 11.5.5 Interrupt Flags

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

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

**Bit 0 – EEREADY** EEREADY Interrupt Flag
This flag is set continuously as long as the EEPROM is not busy. This flag is cleared by writing a '`1`' to it.

### 11.5.6 Data

**Name:** DATA
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

The NVMCTRL.DATAL and NVMCTRL.DATAH register pair represents the 16-bit value, NVMCTRL.DATA.

The low byte [7:0] (suffix L) is accessible at the original offset.

The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

| 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 Register
The Data register will contain the last read value from Flash, EEPROM, or NVMCTRL. For EEPROM access, only DATA[7:0] is used.

### 11.5.7 Address

**Name:** ADDR
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

NVMCTRL.ADDR0, NVMCTRL.ADDR1 and NVMCTRL.ADDR2 represent the 24-bit value NVMCTRL.ADDR.

The low byte [7:0] (suffix 0) is accessible at the original offset.

The high byte [15:8] (suffix 1) can be accessed at offset +`0x01`.

The extended byte [23:16] (suffix 2) can be accessed at offset +`0x02`.

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

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

**Bits 23:0 – ADDR[23:0]** Address
The Address register contains the address of the last memory location that has been accessed. Only the number of bits required to access the memory is used.

# 12. CLKCTRL - Clock Controller

## 12.1 Features

- All Clocks and Clock Sources are Automatically Enabled when Requested by Peripherals
- Internal Oscillators:
  - Up to 24 MHz Internal High-Frequency Oscillator (OSCHF)
  - 32.768 kHz Ultra Low-Power Oscillator (OSC32K)
  - Up to 48 MHz Phase-Locked Loop (PLL), with 2x or 3x clock multiplier
- Auto-Tuning for Improved Internal Oscillator Accuracy
- External Clock Options:
  - 32.768 kHz Crystal Oscillator (XOSC32K)
  - High-Frequency Crystal Oscillator (XOSCHF)
  - External clock
- Main Clock Features:
  - Safe run-time switching
  - Prescaler with a division factor ranging from 1 to 64
  - Clock Failure Detection with automatic clock switching to internal source

## 12.2 Overview

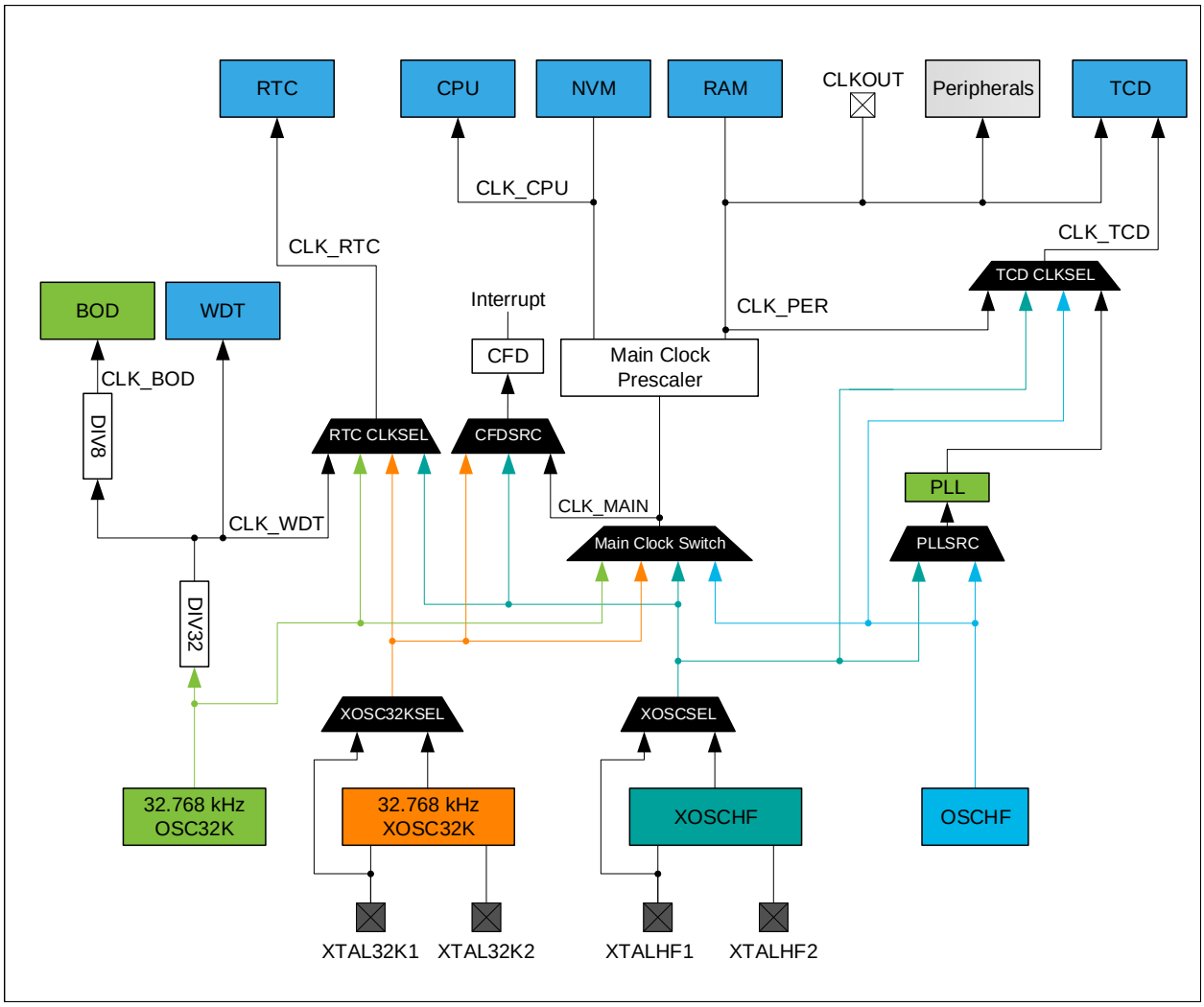
The Clock Controller (CLKCTRL) controls, distributes and prescales the clock signals from the available oscillators. The CLKCTRL supports internal and external clock sources.

The CLKCTRL is based on an automatic clock request system, implemented in all peripherals on the device. The peripherals will automatically request the clocks needed. The request is routed to the correct clock source if multiple clock sources are available.

The Main Clock (CLK_MAIN) is used by the CPU, RAM and all peripherals connected to the I/O bus. The main clock source can be selected and prescaled. Some peripherals can share the same clock source as the main clock, or run asynchronously to the main clock domain.

### 12.2.1 Block Diagram - CLKCTRL

**Figure 12-1. CLKCTRL Block Diagram**




The clock system consists of the main clock and clocks derived from the main clock, as well as several asynchronous clocks:

- Main Clock (CLK_MAIN) is always running in Active and Idle sleep modes. If requested, it will also run in Standby sleep mode.
- CLK_MAIN is prescaled and distributed by the clock controller:
  - CLK_CPU is used by the CPU and the Nonvolatile Memory Controller (NVMCTRL)
  - CLK_PER is used by SRAM and all peripherals that are not listed under asynchronous clocks and can also be routed to the CLKOUT pin
  - All the clock sources can be used as main clock
- Clocks running asynchronously to the main clock domain:
  - CLK_RTC is used by the Real-Time Counter (RTC) and the Periodic Interrupt Timer (PIT). It will be requested when the RTC/PIT is enabled. The clock source for CLK_RTC may be changed only if the peripheral is disabled.
  - CLK_WDT is used by the Watchdog Timer (WDT). It will be requested when the WDT is enabled.
  - CLK_BOD is used by the Brown-out Detector (BOD). It will be requested when the BOD is enabled in the Sampled mode. The alternative clock source is controlled by a fuse.
  - CLK_TCD is used by the Timer Counter type D (TCD). It will be requested when the TCD is enabled. The clock source may be changed only if the peripheral is disabled.

– Clock Failure Detector (CFD) is an asynchronous mechanism to detect a failure on an external crystal or clock source

The clock source for the main clock domain is configured by writing to the Clock Select (CLKSEL) bit field in the Main Clock Control A (CLKCTRL.MCLKCTRLA) register. This register has Configuration Change Protection (CCP), and the appropriate key must be written to the CCP register before writing to the CLKSEL bit field. The asynchronous clock sources are configured by the registers in the respective peripheral.

### 12.2.2 Signal Description

| Signal | Type | Description |
|---|---|---|
| CLKOUT | Digital output | CLK_PER output |
| XTALHF1 | Analog input | Input for external clock source (EXTCLK) or one pin of a high-frequency crystal |
| XTALHF2 | Analog input | Input for one pin of a high-frequency crystal |
| XTAL32K1 | Analog input | Input for external 32.768 kHz clock source or one pin of a 32.768 kHz crystal |
| XTAL32K2 | Analog input | Input for one pin of a 32.768 kHz crystal |

For more details, refer to the *I/O Multiplexing* section.

## 12.3 Functional Description

### 12.3.1 Initialization

To initialize a clock source as the main clock, these steps should be followed:

1. Optional: Force the clock to always run by writing the Run Standby (RUNSTDBY) bit in the respective clock source CTRLA register to '`1`'.
2. Configure the clock source as needed in the corresponding clock source CTRLA register and, if applicable, enable the clock source by writing a '`1`' to the Enable bit.
3. Optional: If RUNSTDBY is '`1`', wait for the clock source to stabilize by polling the respective status bit in CLKCTRL.MCLKSTATUS.
4. The following sub-steps need to be performed in an order such that the main clock frequency never exceeds the allowed maximum clock frequency. Refer to the *Electrical Characteristics* section for further information.
   4.1. If required, divide the clock source frequency by writing to the Prescaler Division (PDIV) bit field and enable the main clock prescaler by writing a '`1`' to the Prescaler Enable (PEN) bit in CLKCTRL.MCLKCTRLB.
   4.2. Select the configured clock source as the main clock in the Clock Select (CLKSEL) bit field in CLKCTRL.MCLKCTRLA.
5. Wait for Main Clock to change by polling the Main Clock Oscillator Changing (SOSC) bit in the Main Clock Status (CLKCTRL.MCLKSTATUS) register.
6. Optional: Clear the RUNSTDBY bit in the clock source CTRLA register.

### 12.3.2 Main Clock Selection and Prescaler

All available oscillators and the external clock (EXTCLK) can be used as the main clock source for the Main Clock (CLK_MAIN). The main clock source is selectable from software and can be safely changed during normal operation.

The Configuration Change Protection mechanism prevents unsafe clock switching. For more details, refer to the *Configuration Change Protection* section.

The Clock Failure Detection mechanism ensures safe switching to internal clock source upon clock failure when enabled.

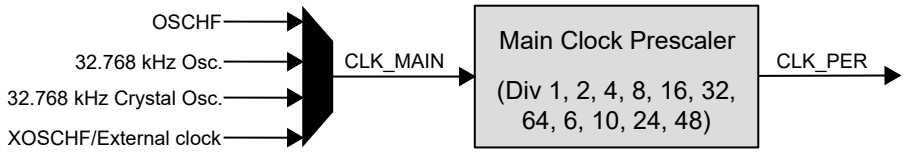
Upon the selection of an external clock source, a switch to the chosen clock source will occur only if edges are detected. Until a sufficient number of clock edges are detected, the switch will not occur, and it will not be possible to change to another clock source again without executing a Reset.

An ongoing clock source switch is indicated by the Main Clock Oscillator Changing (SOSC) bit in the Main Clock Status (CLKCTRL.MCLKSTATUS) register. The stability of the external clock sources is indicated by the respective Status (EXTS and XOSC32KS) bits in CLKCTRL.MCLKSTATUS.

**⚠ CAUTION** If an external clock source fails while used as the CLK_MAIN source, only the Watchdog Timer (WDT) can provide a System Reset.

The CLK_MAIN is fed into the prescaler before being used by the peripherals (CLK_PER) in the device. The prescaler divides CLK_MAIN by a factor from 1 to 64.

**Figure 12-2. Main Clock and Prescaler**




### 12.3.3 Main Clock After Reset

After any Reset, the Main Clock (CLK_MAIN) is provided either by the OSCHF, running at the default frequency of 4 MHz, or the OSC32K, depending on the Clock Select (CLKSEL) bit field configuration of the Oscillator Configuration (FUSE.OSCCFG) fuse. Refer to the description of the FUSE.OSCCFG fuse for details of the possible frequencies after Reset.

### 12.3.4 Clock Sources

All the internal clock sources are automatically enabled when they are requested by a peripheral. The crystal oscillators, based on an external crystal, must be enabled before it can serve as a clock source.

- The XOSC32K oscillator is enabled by writing a '`1`' to the ENABLE bit in the 32.768 kHz Crystal Oscillator Control A (CLKCTRL.XOSC32KCTRLA) register
- The XOSCHF oscillator is enabled by writing a '`1`' to the ENABLE bit in the High Frequency Crystal Oscillator Control A (CLKCTRL.XOSCHFCTRLA) register

After reset, the device starts up running from the internal high-frequency oscillator or the internal 32.768 kHz oscillator.

The respective oscillator status bits in the Main Clock Status (CLKCTRL.MCLKSTATUS) register indicate if the clock source is running and stable.

#### 12.3.4.1 Internal Oscillators

The internal oscillators do not require any external components to run. Refer to the *Electrical Characteristics* section for accuracy and electrical specifications.

##### 12.3.4.1.1 Internal High-Frequency Oscillator (OSCHF)

The OSCHF supports output frequencies of 1, 2, 3, 4 MHz, and multiples of 4, up to 24 MHz, which can be used as main clock, peripheral clock, or as input to the Phase-Locked Loop (PLL).

##### 12.3.4.1.2 32.768 kHz Oscillator (OSC32K)

The 32.768 kHz oscillator is optimized for Ultra Low-Power (ULP) operation. Power consumption is decreased at the cost of decreased accuracy compared to an external crystal oscillator.

This oscillator provides a 1.024 kHz or 32.768 kHz clock for the Real-Time Counter (RTC), the Watchdog Timer (WDT) and the Brown-out Detector (BOD). Additionally, this oscillator can also provide a 32.768 kHz clock to the Main Clock (CLK_MAIN).

For the start-up time of this oscillator, refer to the *Electrical Characteristics* section.

#### 12.3.4.2 External Clock Sources

These external clock sources are available:

- The XTALHF1 and XTALHF2 pins are dedicated to driving a high-frequency crystal oscillator (XOSCHF)
- Instead of a crystal oscillator, XTALHF1 can be configured to accept an external clock source
- The XTAL32K1 and XTAL32K2 pins are dedicated to driving a 32.768 kHz crystal oscillator (XOSC32K)
- Instead of a crystal oscillator, XTAL32K1 can be configured to accept an external clock source

#### 12.3.4.2.1 High Frequency Crystal Oscillator (XOSCHF)

This oscillator supports two input options:

- A crystal is connected to the XTALHF1 and XTALHF2 pins
- An external clock running at up to 32 MHz connected to XTALHF1

The input option must be configured by writing to the Source Select (SELHF) bit in the XOSCHF Control A (CLKCTRL.XOSCHFCTRLA) register.

The maximum frequency of the crystal must be configured by writing to the Frequency Range (FRQRANGE) bit field in XOSCHFCTRLA. This is to ensure sufficient power is delivered to the oscillator to drive the crystal.

The XOSCHF is enabled by writing a '`1`' to the ENABLE bit in XOSCHFCTRLA. When enabled, the configuration of the general purpose input/output (GPIO) pins used by the XOSCHF is overridden as XTALHF1 and XTALHF2 pins. The oscillator needs to be enabled to start running when requested.

The start-up time of a given crystal oscillator can be accommodated by writing to the Crystal Start-up Time (CSUTHF) bit field in XOSCHFCTRLA.

When XOSCHF is configured to use an external clock on XTALHF1, the start-up time is fixed to two cycles.

#### 12.3.4.2.2 32.768 kHz Crystal Oscillator (XOSC32K)

This oscillator supports two input options:

- A crystal is connected to the XTAL32K1 and XTAL32K2 pins
- An external clock running at 32.768 kHz, connected to XTAL32K1

The input option must be configured by writing the Source Select (SEL) bit in the XOSC32K Control A (CLKCTRL.XOSC32KCTRLA) register.

The XOSC32K is enabled by writing a '`1`' to the ENABLE bit in CLKCTRL.XOSC32KCTRLA. When enabled, the configuration of the general purpose input/output (GPIO) pins used by the XOSC32K is overridden as XTAL32K1 and XTAL32K2 pins. The oscillator needs to be enabled to start running when requested.

The start-up time of a given crystal oscillator can be accommodated by writing to the Crystal Start-up Time (CSUT) bit field in XOSC32KCTRLA.

When XOSC32K is configured to use an external clock on XTAL32K1, the start-up time is fixed to two cycles.

### 12.3.5 Phase-Locked Loop (PLL)

The PLL can be used to increase the frequency of the clock source defined by the SOURCE bit in the PLL Control A (CLKCTRL.PLLCTRLA) register. The minimum input frequency of the PLL is 16 MHz, and the maximum output frequency is 48 MHz.

Initialization:

1. Enable the clock source to be used as input.
2. Configure SOURCE in CLKCTRL.PLLCTRLA to the desired clock source.
3. Enable the PLL by writing the desired multiplication factor to the Frequency Select (MULFAC) bit field in PLLCTRLA.
4. Wait for the PLL Status (PLLS) bit in the Main Clock Status (CLKCTRL.MCLKSTATUS) register to become '`1`', indicating that the PLL has locked in on the desired frequency.

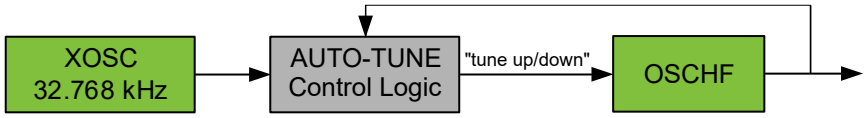
For available connections, refer to the *Block Diagram - CLKCTRL* figure in the Clock Controller (CLKCTRL) section.

### 12.3.6 Auto-Tune

The auto-tune feature can be used to improve the accuracy of the internal oscillator. The internal 1 MHz clock is compared to a 1.024 kHz reference from the crystal. If an error is detected, the frequency calibration will be adjusted according to the error.

The check against the reference clock runs for as long as an error is detected. When the oscillator is tuned in, and no error is detected, the auto-tune feature will disable itself for 64 ms before it starts a new check.

**Figure 12-3. OSCHF Auto-Tune Block Diagram**



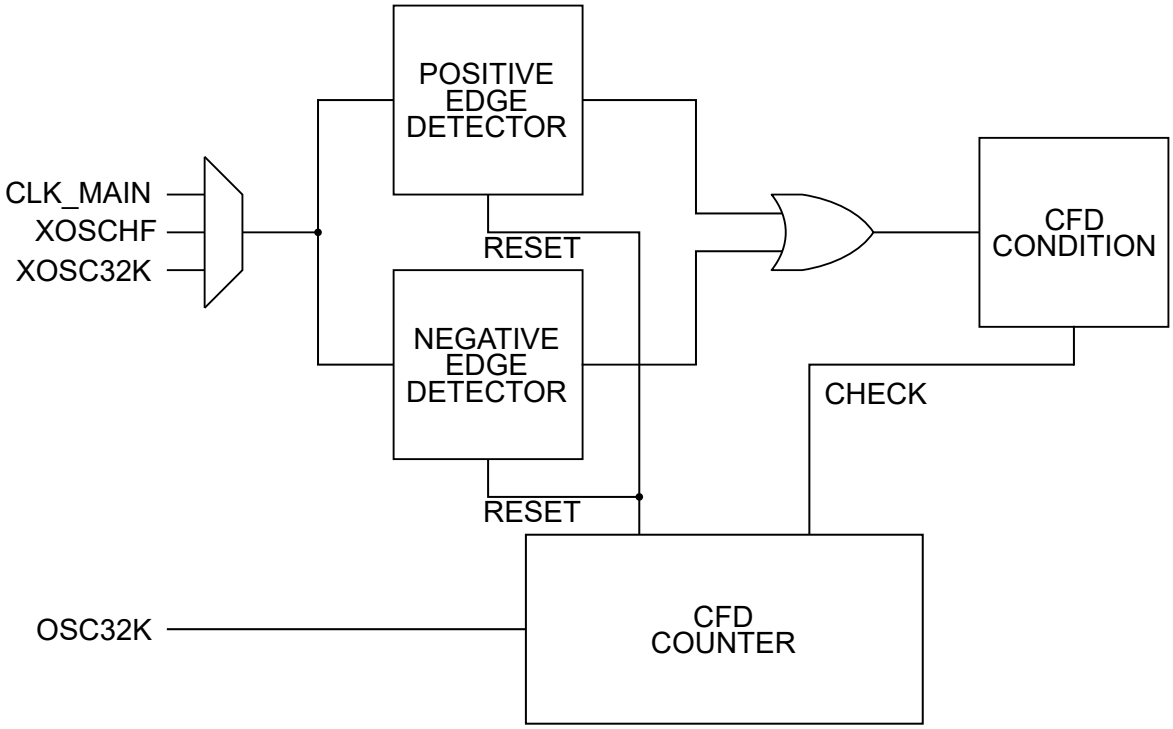

### 12.3.7 Clock Failure Detection

The Clock Failure Detection (CFD) allows the device to continue operating if an external crystal oscillator or clock source should fail. The CFD is enabled by writing a '1' to the Clock Failure Detection Enable (CFDEN) bit in the Main Clock Control C (CLKCTRL.MCLKCTRLC) register. See the Clock Failure Detection Block Diagram for monitorable oscillators and clocks sources.

#### 12.3.7.1 CFD Operation

The Clock Failure Detection (CFD) module detects a failed oscillator or clock source by checking for edges on the selected oscillator/clock during a CFD period. A CFD period is ten cycles on the OSC32K internal oscillator. In the first eight cycles of the CFD period, edge detectors will detect edges on the monitored oscillator/clock. The two remaining cycles are used to check and issue a CFD condition if no edges are present, and to reset the edge detectors at the end of the CFD period.

**Figure 12-4. Clock Failure Detection Block Diagram**




When the CFD module is enabled, it will monitor the selected source from the Clock Failure Detection Source (CFDSRC) bit field in the Main Clock Control C (CLKCTRL.MCLKCTRLC). In sleep, the CFD will only be enabled if the selected source is active.

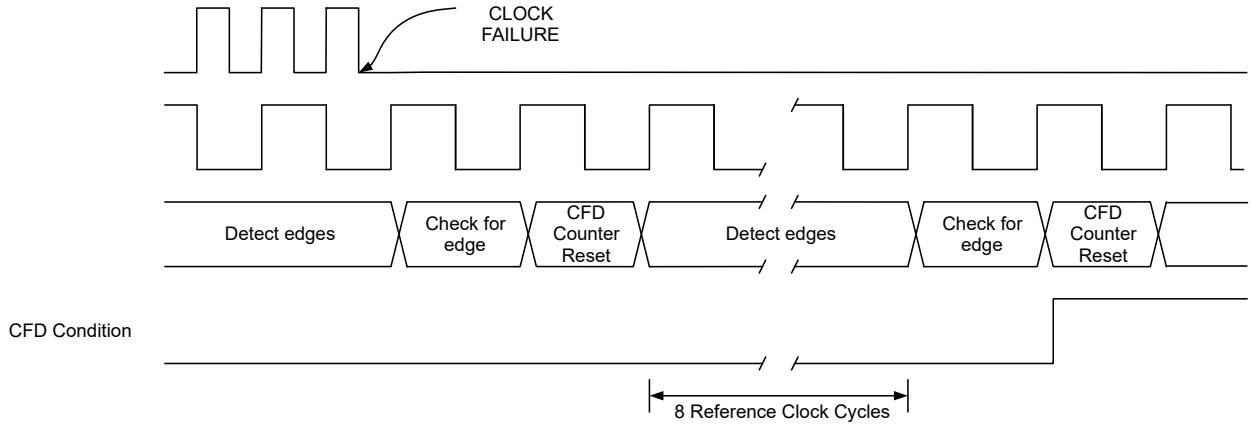
If a CFD condition occurs, the CFD interrupt flag in the Main Clock Interrupt Flags (CLKCTRL.MCLKINTFLAGS) register is set, and if the interrupt is enabled, an interrupt request is issued. The Interrupt Type (INTTYP) bit in the Main clock Interrupt control (CLKCTRL.MCLKINTCTRL) register determines if a normal interrupt or a non-maskable interrupt (NMI) will be issued. If the NMI is selected, the interrupt will change from vector number 3 (CLKCTRL) to vector number 1 (NMI). If more than one interrupt source is set to NMI, it is necessary to check the vector when an interrupt triggers to see which source experienced an interrupt.

If the selected source is the main clock and it fails, everything running on it will stop. Therefore, the CFD condition will overwrite the Clock Selection (CLKSEL) bit field in the Main Clock Control A(CLKCTRL.MCLKCTRLA) register to select the start-up clock source, and the Main Clock Control B (CLKCTRL.MCLKCTRLB) register will be written to the reset value. The start-up clock source is changed back to its reset frequency.

When the CLKSEL is overridden by a CFD event, the Main Clock Out (CLKOUT) bit in CLKCTRL.MCLKCTRLA will be set to '0', disabling the CLKOUT signal.

The start-up clock source is defined as the clock the system runs on after a power-on reset. This start-up clock source is selectable by fuse(s).

**Figure 12-5. Clock Failure Detection Timing Diagram**




### 12.3.7.2 Condition Clearing

The Clock Failure Detection (CFD) condition is cleared after a Reset, the monitored source starts toggling again or the CFD flag in the Main Clock Interrupt Flags (CLKCTRL.MCLKINTFLAGS) register is set. Note that as long as the failure condition is met, the interrupt will trigger every ten OSC32K cycles. If these repeated interrupts are not desired, write a '0' to the Clock Failure Detection Interrupt Enable (CFD) bit in the Main Clock Interrupt Control (CLKCTRL.MCLKCTRL) register. If it is the main clock that is being monitored, changing back to the default start-up clock will make the main clock start toggling again, clearing the condition.

### 12.3.7.3 CFD Test

The Clock Failure Detection Test (CFDTST) bit in the Main Clock Control C (CLKCTL.CTRLC) register can be used to trigger a clock failure in the clock failure detector. Depending on the use-case, there are two different modes of testing the clock failure detector.

#### 12.3.7.3.1 Testing the Clock Failure Without Influencing the Main Clock

This mode is intended to use run-time. To not influence the main clock when writing to the Clock Failure Detection Test (CFDTST) in the Main Clock Control C (CLKCTRL.MCLKCTRLC) register, the Clock Failure Detection Source (CFDSRC) in CLKCTRL.MCLKCTRLC must be configured to a clock source different than the main clock. CFDSRC must be different from '0'. The CFD Interrupt Flag in the Main Clock Interrupt Flags (CLKCTRL.MCLKINTFLAGS) register will be set, but the main clock will not change to the start-up clock source.

If the clock failure detector is monitoring the main clock and a run-time check of the clock failure detector is needed, it is necessary to do the following steps:

1. Disable the clock failure detector by writing a '0' to the Clock Failure Detection Enable (CFDEN) bit in CLKCTRL.MCLKCTRLC, and change the source to the oscillator directly by writing a number other than a '0' to the CFDSRC bit.
2. Write a '1' to the CFD Interrupt Flag in CLKCTRL.MCLKINTFLAGS to clear the flag.
3. Write a '1' to the CFDTST bit and enable the clock failure detector again by writing a '1' to the CFDEN bit.
4. Wait for the CFD bit in CLKCTRL.MCLKINTFLAGS to be set to check that the clock failure works.
5. Disable the clock failure detector by writing a '0' to the CFDEN bit, and change the source to the main clock again by writing a '0' to the CFDSRC.
6. Enable the clock failure detector again by writing a '1' to the CFDEN bit, and write a '0' to the CFDTST bit.

#### 12.3.7.3.2 Testing the Clock Failure and Changing the Main Clock to the Start-up Clock Source

If the Clock Failure Detection Source (CFDSRC) bit field in the Main Clock Control C (CLKCTRL.MCLKCTRLC) register has the value `0x0` and the main clock is monitored, writing a '`1`' to the Clock Failure Detection Test (CDFTST) bit in MCLKCTRLC will trigger a fault that will change the main clock to the start-up clock source.

### 12.3.8 Sleep Mode Operation

When a clock source is not used or requested, it will stop. It is possible to request a clock source directly by writing a '`1`' to the Run Standby (RUNSTDBY) bit in the respective oscillator's Control A (CLKCTRL.*oscillator*CTRLA) register. This will cause the oscillator to run constantly, except for Power-Down sleep mode. Additionally, when this bit is written to a '`1`', the oscillator start-up time is eliminated when the clock source is requested by a peripheral.

The main clock will always run in Active and Idle sleep modes. In Standby sleep mode, the main clock will run only if any peripheral is requesting it, or RUNSTDBY in the respective oscillator's CLKCTRL.*oscillator*CTRLA register is written to a '`1`'.

In Power-Down sleep mode, the main clock will stop after all nonvolatile memory (NVM) operations are completed. Refer to the *Sleep Controller* section for more details on sleep mode operation.

In sleep, the Clock Failure Detection (CFD) will only be enabled if the selected source is active. After a Reset, the CFD will not start looking for failure until a time equivalent to the monitored Oscillator Start-up Timer (SUT) has expired.

### 12.3.9 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a certain key must first be written to the CPU.CCP register, followed by a write access to the protected bits within four CPU instructions.

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged.

The following registers are under CCP:

**Table 12-1. CLKCTRL - Registers Under Configuration Change Protection**

| Register | Key |
|---|---|
| CLKCTRL.MCLKCTRLA | IOREG |
| CLKCTRL.MCLKCTRLB | IOREG |
| CLKCTRL.MCLKCTRLC | IOREG |
| CLKCTRL.MCLKINTCTRL | IOREG |
| CLKCTRL.OSCHFCTRLA | IOREG |
| CLKCTRL.PLLCTRLA | IOREG |
| CLKCTRL.OSC32KCTRLA | IOREG |
| CLKCTRL.XOSC32KCTRLA | IOREG |
| CLKCTRL.XOSCHFCTRLA | IOREG |

## 12.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | MCLKCTRLA | 7:0 | CLKOUT | | | | CLKSEL[3:0] | | | |
| 0x01 | MCLKCTRLB | 7:0 | | | | PDIV[3:0] | | | | PEN |
| 0x02 | MCLKCTRLC | 7:0 | | | | | CFDSRC[1:0] | | CFDTST | CFDEN |
| 0x03 | MCLKINTCTRL | 7:0 | INTTYPE | | | | | | | CFD |
| 0x04 | MCLKINTFLAGS | 7:0 | | | | | | | | CFD |
| 0x05 | MCLKSTATUS | 7:0 | | | PLLS | EXTS | XOSC32KS | OSC32KS | OSCHFS | SOSC |
| 0x06 ... 0x07 | Reserved | | | | | | | | | |
| 0x08 | OSCHFCTRLA | 7:0 | RUNSTDBY | | FRQSEL[3:0] | | | | | AUTOTUNE |
| 0x09 | OSCHFTUNE | 7:0 | TUNE[7:0] | | | | | | | |
| 0x0A ... 0x0F | Reserved | | | | | | | | | |
| 0x10 | PLLCTRLA | 7:0 | RUNSTDBY | SOURCE | | | | | MULFAC[1:0] | |
| 0x11 ... 0x17 | Reserved | | | | | | | | | |
| 0x18 | OSC32KCTRLA | 7:0 | RUNSTDBY | | | | | | | |
| 0x19 ... 0x1B | Reserved | | | | | | | | | |
| 0x1C | XOSC32KCTRLA | 7:0 | RUNSTDBY | | CSUT[1:0] | | | SEL | LPMODE | ENABLE |
| 0x1D ... 0x1F | Reserved | | | | | | | | | |
| 0x20 | XOSCHFCTRLA | 7:0 | RUNSTDBY | | CSUTHF[1:0] | | FRQRANGE[1:0] | | SELHF | ENABLE |

## 12.5 Register Description

### 12.5.1 Main Clock Control A

**Name:** MCLKCTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CLKOUT | | | | CLKSEL[3:0] | | | |
| Access | R/W | | | | R/W | R/W | R/W | R/W |
| Reset | 0 | | | | 0 | 0 | 0 | 0 |

**Bit 7 – CLKOUT** Main Clock Out
This bit controls whether the main clock is available on the Main Clock Out (CLKOUT) pin or not, when the main clock is running.
This bit is cleared when a '0' is written to it or when a Clock Failure Detection (CFD) condition with the main clock as source occurs.
This bit is set when a '1' is written to it.

| Value | Description |
|---|---|
| 0 | The main clock is not available on the CLKOUT pin |
| 1 | The main clock is available on the CLKOUT pin |

**Bits 3:0 – CLKSEL[3:0]** Clock Select
This bit field controls the source for the Main Clock (CLK_MAIN).

| Value | Name | Description |
|---|---|---|
| 0x0 | OSCHF | Internal high-frequency oscillator |
| 0x1 | OSC32K | 32.768 kHz internal oscillator |
| 0x2 | XOSC32K | 32.768 kHz external crystal oscillator |
| 0x3 | EXTCLK | External clock or external crystal, depending on the SELHF bit in XOSCHFCTRLA |
| Other | Reserved | Reserved |

### 12.5.2 Main Clock Control B

**Name:** MCLKCTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | PDIV[3:0] | | | | PEN |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:1 – PDIV[3:0]** Prescaler Division
This bit field controls the division ratio of the Main Clock (CLK_MAIN) prescaler, when the Prescaler (PEN) bit is '1'.

| Value | Name | Description |
|---|---|---|
| 0x0 | 2X | Divide by 2 |
| 0x1 | 4X | Divide by 4 |
| 0x2 | 8X | Divide by 8 |
| 0x3 | 16X | Divide by 16 |
| 0x4 | 32X | Divide by 32 |
| 0x5 | 64X | Divide by 64 |
| 0x8 | 6X | Divide by 6 |
| 0x9 | 10X | Divide by 10 |
| 0xA | 12X | Divide by 12 |
| 0xB | 24X | Divide by 24 |
| 0xC | 48X | Divide by 48 |
| Other | - | Reserved |

**Note:** Configuration of the input frequency (CLK_MAIN) and prescaler settings must not exceed the allowed maximum frequency of the peripheral clock (CLK_PER) or CPU clock (CLK_CPU). Refer to the *Electrical Characteristics* section for further information.

**Bit 0 – PEN** Prescaler Enable
This bit controls whether the Main Clock (CLK_MAIN) prescaler is enabled or not.

| Value | Description |
|---|---|
| 0 | The CLK_MAIN prescaler is disabled |
| 1 | The CLK_MAIN prescaler is enabled, and the division ratio is controlled by the Prescaler Division (PDIV) bit field |

### 12.5.3 Main Clock Control C

**Name:** MCLKCTRLC
**Offset:** 0x02
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CFDSRC[1:0] | | CFDTST | CFDEN |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:2 – CFDSRC[1:0]** Clock Failure Detection Source
This bit field controls which clock source to monitor, when the Clock Failure Detection Enable (CFDEN) bit is '1'.

| Value | Name | Description |
|---|---|---|
| 0x0 | CLKMAIN | Main Clock |
| 0x1 | XOSCHF | External High Frequency Oscillator |
| 0x2 | XOSC32K | External 32.768 kHz Oscillator |
| Other | Reserved | Reserved |

**Note:** This bit field is read-only when the CFDEN bit is '1', and both the Clock Failure Detection (CFD) interrupt enable bit and Interrupt Type (INTTYPE) bit in the Main Clock Interrupt Control (CLKCTRL.MCLKINTCTRL) are '1'. This bit will remain read-only until a System Reset occurs.

**Bit 1 – CFDTST** Clock Failure Detection Test
This bit controls testing of the Clock Failure Detection (CFD) functionality.
Writing a '0' to this bit will clear the bit, and the ongoing CFD test fail condition.
Writing a '1' to this bit will set the bit and force a CFD fail condition.

| Value | Description |
|---|---|
| 0 | No ongoing test of the CFD functionality |
| 1 | A CFD fail condition has been forced |

**Bit 0 – CFDEN** Clock Failure Detection Enable
This bit controls whether Clock Failure Detection (CFD) is enabled or not.

| Value | Description |
|---|---|
| 0 | CFD is disabled |
| 1 | CFD is enabled |

**Note:** This bit is read-only when this bit is '1', and both the Clock Failure Detection (CFD) interrupt enable bit and Interrupt Type (INTTYPE) bit in the Main Clock Interrupt Control (CLKCTRL.MCLKINTCTRL) are '1'. This bit will remain read-only until a System Reset occurs.

### 12.5.4 Main Clock Interrupt Control

**Name:** MCLKINTCTRL
**Offset:** 0x03
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INTTYPE | | | | | | | CFD |
| Access | R/W | | | | | | | R/W |
| Reset | 0 | | | | | | | 0 |

**Bit 7 – INTTYPE** Interrupt Type
This bit controls the type of the Clock Failure Detection (CFD) interrupt.

| Value | Name | Description |
|---|---|---|
| 0 | INT | Regular Interrupt |
| 1 | NMI | Non-Maskable Interrupt |

**Note:** This bit is read-only when the Clock Failure Detection Enable (CFDEN) bit in the Main Clock Control C (CLKCTRL.MCLKCTRLC) register is '1', and both the Clock Failure Detection (CFD) interrupt enable bit and this bit are '1.' This bit will remain read-only until a System Reset occurs.

**Bit 0 – CFD** Clock Failure Detection Interrupt Enable
This bit controls whether the Clock Failure Detection (CFD) interrupt is enabled or not.

| Value | Description |
|---|---|
| 0 | The CFD interrupt is disabled |
| 1 | The CFD interrupt is enabled |

**Note:** This bit is read-only when the Clock Failure Detection Enable (CFDEN) bit in the Main Clock Control C (CLKCTRL.MCLKCTRLC) register is '1', and both the Interrupt Type (INTTYPE) bit and this bit are '1.' This bit will remain read-only until a System Reset occurs.

### 12.5.5 Main Clock Interrupt Flags

**Name:** MCLKINTFLAGS
**Offset:** 0x04
**Reset:** 0x0
**Property:** -

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

**Bit 0 – CFD** Clock Failure Detection Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when a clock failure is detected.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Clock Failure Detection (CFD) interrupt flag.

### 12.5.6 Main Clock Status

**Name:** MCLKSTATUS
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | PLLS | EXTS | XOSC32KS | OSC32KS | OSCHFS | SOSC |
| Access | | | R | R | R | R | R | R |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 5 – PLLS** PLL Status

| Value | Description |
|---|---|
| 0 | PLL is not running |
| 1 | PLL is running |

**Bit 4 – EXTS** External Crystal/Clock Status

| Value | Description |
|---|---|
| 0 | The external high-frequency crystal is not stable, when the Source Select (SELHF) bit in the External High-Frequency Oscillator Control A (CLKCTRL.XOSCHFCTRLA) register is '0'.<br>The external high-frequency clock is not running, when the SELHF bit is '1'. |
| 1 | The external high-frequency crystal is stable, when the SELHF bit is '0'.<br>The external high-frequency clock is running, when the SELHF bit is '1'. |

**Bit 3 – XOSC32KS** XOSC32K Status

| Value | Description |
|---|---|
| 0 | The external 32.768 kHz crystal is not stable, when the Source Select (SEL) bit in the 32.768 Crystal Oscillator Control A (CLKCTRL.XOSC32K) register is '0'.<br>The external 32.768 kHz clock is not running, when the SEL bit is '1'. |
| 1 | The external 32.768 kHz crystal is stable, when the SEL bit is '0'.<br>The external 32.768 kHz clock is running, when the SEL bit is '1'. |

**Bit 2 – OSC32KS** OSC32K Status

| Value | Description |
|---|---|
| 0 | OSC32K is not stable |
| 1 | OSC32K is stable |

**Bit 1 – OSCHFS** Internal High-Frequency Oscillator Status

| Value | Description |
|---|---|
| 0 | OSCHF is not stable |
| 1 | OSCHF is stable |

**Bit 0 – SOSC** Main Clock Oscillator Changing

| Value | Description |
|---|---|
| 0 | The clock source for CLK_MAIN is not undergoing a switch |
| 1 | The clock source for CLK_MAIN is undergoing a switch and will change as soon as the new source is stable |

### 12.5.7 Internal High-Frequency Oscillator Control A

**Name:** OSCHFCTRLA
**Offset:** 0x08
**Reset:** 0x0C
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | | FRQSEL[3:0] | | | | | AUTOTUNE |
| Access | R/W | | R/W | R/W | R/W | R/W | | R/W |
| Reset | 0 | | 0 | 0 | 1 | 1 | | 0 |

**Bit 7 – RUNSTDBY** Run Standby
This bit controls whether the Internal High-Frequency Oscillator (OSCHF) is always running or not.

| Value | Description |
|---|---|
| 0 | The OSCHF oscillator will only run when requested by a peripheral or by the main clock (1) |
| 1 | The OSCHF oscillator will always run in Active, Idle and Standby sleep modes (2) |

**Notes:**
1. The requesting peripheral, or the main clock, must take the oscillator start-up time into account.
2. The oscillator signal is only available if requested, and will be available after two OSCHF cycles.

**Bits 5:2 – FRQSEL[3:0]** Frequency Select
This bit field controls the output frequency of the Internal High-Frequency (OSCHF) oscillator.

| Value | Name | Description |
|---|---|---|
| 0x0 | 1M | 1 MHz output |
| 0x1 | 2M | 2 MHz output |
| 0x2 | 3M | 3 MHz output |
| 0x3 | 4M | 4 MHz output (default) |
| 0x4 | - | Reserved |
| 0x5 | 8M | 8 MHz output |
| 0x6 | 12M | 12 MHz output |
| 0x7 | 16M | 16 MHz output |
| 0x8 | 20M | 20 MHz output |
| 0x9 | 24M | 24 MHz output |
| Other | - | Reserved |

**Bit 0 – AUTOTUNE** Auto-Tune Enable
This bit controls whether the 32.768 kHz crystal auto-tune functionality of the Internal High-Frequency Oscillator (OSCHF) is enabled or not.

| Value | Description |
|---|---|
| 0 | The auto-tune functionality of the OSCHF oscillator is disabled |
| 1 | The auto-tune functionality of the OSCHF oscillator is enabled |

### 12.5.8 Internal High-Frequency Oscillator Frequency Tune

**Name:** OSCHFTUNE
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TUNE[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 – TUNE[7:0]** User Frequency Tuning
This bit field controls the manual tuning of the output frequency of the Internal High-Frequency Oscillator (OSCHF). The frequency can be tuned 32 steps down or 31 steps up from the oscillator's target frequency. Thus, the register's acceptable input value range is -32 to +31.
Writing to bit 6 and 7 has no effect, as bit 5 will be mirrored to bit 6 and 7, due to the 6-bit value in this bit field being represented in a signed (two's complement) form.

**Note:** If the Auto-Tune Enable (AUTOTUNE) bit in the Internal High-Frequency Oscillator Control A (CLKCTRL.OSCHFCTRLA) register is enabled, the TUNE value is locked. This bit field is updated with the latest tune value from the auto-tune operation when AUTOTUNE is disabled.

### 12.5.9 PLL Control A

**Name:** PLLCTRLA
**Offset:** 0x10
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | SOURCE | | | | | MULFAC[1:0] | |
| Access | R/W | R/W | | | | | R/W | R/W |
| Reset | 0 | 0 | | | | | 0 | 0 |

**Bit 7 – RUNSTDBY** Run Standby
This bit controls whether the Phase-Locked Loop (PLL) is always running or not.

| Value | Description |
|---|---|
| 0 | The PLL will only run if requested by a peripheral (1) |
| 1 | The PLL will always run in Active, Idle and Standby sleep modes (2) |

**Notes:**
1. The requesting peripheral must take the PLL start-up time and PLL source start-up time into account.
2. The oscillator signal will only be available if requested and will be available after two PLL cycles.

**Bit 6 – SOURCE** Select Source for PLL
This bit controls the Phase-Locked Loop (PLL) clock source.

| Value | Name | Description |
|---|---|---|
| 0 | OSCHF | High-frequency internal oscillator as PLL source |
| 1 | XOSCHF | High-frequency external clock or external high-frequency oscillator as PLL source |

**Bits 1:0 – MULFAC[1:0]** Multiplication Factor
This bit field controls the multiplication factor for the Phased-Locked Loop (PLL).

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLE | PLL is disabled |
| 0x1 | 2x | 2 x multiplication factor |
| 0x2 | 3x | 3 x multiplication factor |
| 0x3 | - | Reserved |

### 12.5.10 32.768 kHz Oscillator Control A

**Name:** OSC32KCTRLA
**Offset:** 0x18
**Reset:** 0x00
**Property:** Configuration Change Protection

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

**Bit 7 – RUNSTDBY** Run Standby
This bit controls whether the 32.768 kHz Oscillator (OSC32K) is always running or not.

| Value | Description |
|---|---|
| 0 | The OSC32K oscillator will only run when requested by a peripheral or by the main clock (1) |
| 1 | The OSC32K oscillator will always run in Active, Idle, Standby and Power-Down sleep modes (2) |

**Notes:**
1. The requesting peripheral, or the main clock, must take the oscillator start-up time into account.
2. The oscillator signal is only available if requested and will be available after four OSC32K cycles.

### 12.5.11 32.768 kHz Crystal Oscillator Control A

**Name:** XOSC32KCTRLA
**Offset:** 0x1C
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | | CSUT[1:0] | | | SEL | LPMODE | ENABLE |
| Access | R/W | | R/W | R/W | | R/W | R/W | R/W |
| Reset | 0 | | 0 | 0 | | 0 | 0 | 0 |

**Bit 7 – RUNSTDBY** Run Standby
This bit controls whether the 32.768 kHz Crystal Oscillator (XOSC32K) is always running or not, and in which modes, when the ENABLE bit is '1'.

| Value | Description |
|---|---|
| 0 | The XOSC32K oscillator will only run when requested by a peripheral or by the main clock, in Active and Idle sleep mode(1) |
| 1 | The XOSC32K oscillator will always run in Active, Idle, Standby and Power-Down sleep modes(2) |

**Notes:**
1. The requesting peripheral, or the main clock, must take the oscillator start-up time into account.
2. The oscillator signal is only available if requested, and will be available after a maximum of three XOSC32K cycles, if the initial crystal start-up time has already ended.

**Bits 5:4 – CSUT[1:0]** Crystal Start-Up Time
This bit field controls the start-up time of the 32.768 kHz Crystal Oscillator (XOSC32K), when the Source Select (SELHF) is '0'.

| Value | Name | Description |
|---|---|---|
| 0x0 | 1K | 1k cycles |
| 0x1 | 16K | 16k cycles |
| 0x2 | 32K | 32k cycles |
| 0x3 | 64K | 64k cycles |

**Note:** This bit field is read-only when the ENABLE bit or the XOSC32K Status (XOSCS) bit in the Main Clock Status (CLKCTRL.MCLKSTATUS) register is '1'.

**Bit 2 – SEL** Source Select
This bit controls the source of the 32.768 kHz Crystal Oscillator (XOSC32K).

| Value | Description |
|---|---|
| 0 | External crystal the XTAL32K1 and XTAL32K2 pins |
| 1 | External clock on the XTAL32K1 pin |

**Note:** This bit field is read-only when the ENABLE bit or the XOSC32K Status (XOSCS) bit in the Main Clock Status (CLKCTRL.MCLKSTATUS) register is '1'.

**Bit 1 – LPMODE** Low-Power Mode
This bit controls whether the 32.768 kHz Crystal Oscillator (XOSC32K) is in Low-Power mode or not.

| Value | Description |
|---|---|
| 0 | The Low-Power mode is disabled |
| 1 | The Low-Power mode is enabled |

**Bit 0 – ENABLE** Enable
This bit controls whether the 32.768 kHz Crystal Oscillator (XOSC32K) is enabled or not.

| Value | Description |
|---|---|
| 0 | The XOSC32K oscillator is disabled |

| Value | Description |
|---|---|
| 1 | The XOSC32K oscillator is enabled, and overrides normal port operation for the respective oscillator pins |

### 12.5.12 External High-Frequency Oscillator Control A

**Name:** XOSCHFCTRLA
**Offset:** 0x20
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | | CSUTHF[1:0] | | FRQRANGE[1:0] | | SELHF | ENABLE |
| 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 – RUNSTDBY** Run Standby
This bit controls whether the External High-Frequency Oscillator (XOSCHF) is always running or not, when the ENABLE bit is '1'.

| Value | Description |
|---|---|
| 0 | The XOSCHF oscillator will only run when requested by a peripheral or by the main clock (1) |
| 1 | The XOSCHF oscillator will always run in Active, Idle and Standby sleep modes (2) |

**Notes:**
1. The requesting peripheral, or the main clock, must take the oscillator start-up time into account.
2. The oscillator signal is only available if requested, and will be available after two XOSCHF cycles, if the initial crystal start-up time has already ended.

**Bits 5:4 – CSUTHF[1:0]** Crystal Start-up Time
This bit field controls the start-up time for the External High-Frequency Oscillator (XOSCHF), when the Source Select (SELHF) bit is '0'.

| Value | Name | Description |
|---|---|---|
| 0x0 | 256 | 256 XOSCHF cycles |
| 0x1 | 1K | 1K XOSCHF cycles |
| 0x2 | 4K | 4K XOSCHF cycles |
| 0x3 | - | Reserved |

**Note:** This bit field is read-only when the ENABLE bit or the External Crystal/Clock Status (XOSCHFS) bit in the Main Clock Status (MCLKSTATUS) register is '1'.

**Bits 3:2 – FRQRANGE[1:0]** Frequency Range
This bit field controls the maximum frequency supported for the external crystal. The larger the range selected, the higher the current consumption by the oscillator.

| Value | Name | Description |
|---|---|---|
| 0x0 | 8M | Max. 8 MHz XTAL frequency |
| 0x1 | 16M | Max. 16 MHz XTAL frequency |
| 0x2 | 24M | Max. 24 MHz XTAL frequency |
| 0x3 | 32M | Max. 32 MHz XTAL frequency |

**Note:** If a crystal with a frequency larger than the maximum supported CLK_CPU frequency is used and used as the main clock, it is necessary to divide it down by writing the appropriate configuration to the PDIV bit field in the Main Clock Control B register.

**Bit 1 – SELHF** Source Select
This bit controls the source of the External High-Frequency Oscillator (XOSCHF).

| Value | Name | Description |
|---|---|---|
| 0 | CRYSTAL | External Crystal on the XTALHF1 and XTALHF2 pins |
| 1 | EXTCLOCK | External Clock on the XTALHF1 pin |

**Note:** This bit field is read-only when the ENABLE bit or the External Crystal/Clock Status (XOSCHFS) bit in the Main Clock Status (MCLKSTATUS) register is '1'.

**Bit 0 – ENABLE** Enable
This bit controls whether the External High-Frequency Oscillator (XOSCHF) is enabled or not.

| Value | Description |
|---|---|
| 0 | The XOSCHF oscillator is disabled |
| 1 | The XOSCHF oscillator is enabled, and overrides normal port operation for the respective oscillator pins |

# 13. SLPCTRL - Sleep Controller

## 13.1 Features

- Power Management for Adjusting Power Consumption and Functions
- Three Sleep Modes:
  - Idle
  - Standby
  - Power-Down
- Configurable Standby Mode where Peripherals Can Be Configured as ON or OFF

## 13.2 Overview

Sleep modes are used to shut down peripherals and clock domains in the device in order to save power. The Sleep Controller (SLPCTRL) controls and handles the transitions between Active and sleep modes.

There are four modes available: One Active mode in which software is executed, and three sleep modes. The available sleep modes are Idle, Standby and Power-Down.

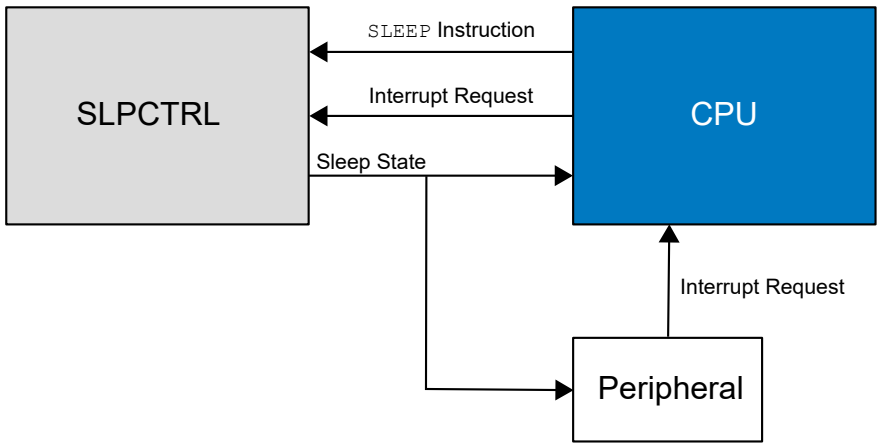
All sleep modes are available and can be entered from the Active mode. In Active mode, the CPU is executing application code. When the device enters sleep mode, the program execution is stopped. The application code decides which sleep mode to enter and when.

Interrupts are used to wake the device from sleep. The available interrupt wake-up sources depend on the configured sleep mode. When an interrupt occurs, the device will wake up and execute the Interrupt Service Routine before continuing normal program execution from the first instruction after the `SLEEP` instruction. Any Reset will take the device out of sleep mode.

The content of the register file, SRAM and registers, is kept during sleep. If a Reset occurs during sleep, the device will reset, start and execute from the Reset vector.

### 13.2.1 Block Diagram

**Figure 13-1. Sleep Controller in the System**




## 13.3 Functional Description

### 13.3.1 Initialization

To put the device into a sleep mode, follow these steps:

1. Configure and enable the interrupts that are able to wake the device from sleep.
   Also, enable global interrupts.

   **WARNING** If there are no interrupts enabled when going to sleep, the device cannot wake up again. Only a Reset will allow the device to continue operation.

2. Select which sleep mode to enter and enable the Sleep Controller by writing to the Sleep Mode (SMODE) bit field and the Enable (SEN) bit in the Control A (SLPCTRL.CTRLA) register.
   The `SLEEP` instruction must be executed to make the device go to sleep.

### 13.3.2 Voltage Regulator Configuration

A voltage regulator is used to regulate the core voltage. The regulator can be configured to balance power consumption, wake-up time from Sleep, and maximum clock speed.

The Voltage Regulator Control (SLPCTRL.VREGCTRL) register is used to configure the regulator start-up time and power consumption. The Power Mode Select (PMODE) bit field in SLPCTRL.VREGCTRL can be set to make the regulator switch to Normal mode when OSC32K is the only oscillator enabled and if the device is in sleep mode. In Normal mode, the regulator consumes less power, but can supply only a limited amount of current, permitting only a low clock frequency.

The user may select one of the following Voltage Regulator Power modes:

**Table 13-1. Voltage Regulator Power Modes Description**

| Voltage Regulator Power Mode | Description |
|---|---|
| Normal (AUTO) | Maximum performance in Active mode and Idle mode |
| Performance (FULL) | Maximum performance in all modes (Active and Sleep) and fast start-up from all sleep modes |

### 13.3.3 Operation

#### 13.3.3.1 Sleep Modes

There are three sleep modes that can be enabled to reduce power consumption.

**Idle** The CPU stops executing code. All peripherals are running and all interrupt sources can wake the device.

**Standby** The user can configure peripherals to be enabled or not, using the respective RUNSTDBY bit. This means that the power consumption is highly dependent on what functionality is enabled, and thus may vary between the Idle and Power-Down modes.
Operation in Standby mode is enabled for a peripheral by writing a '`1`' to the RUNSTDBY bit within the control register of each peripheral.

**Power-Down** The Watchdog Timer (WDT) and the Periodic Interrupt Timer (PIT) are active.
The only wake-up sources are the pin change interrupt, TWI address match, and CCL (if filter and edge-detect are disabled).
For temperature ranges above 70°C, the HTLLEN bit in the SLPCTRL.VREGCTRL register can be used to reduce power consumption (High-Temperature Low Leakage Enable).

**Important:** The TWI address match and CCL wake-up sources are not functional when High-Temperature Low Leakage Enable is activated and must be disabled by the user to avoid unpredictable behavior.

**Table 13-2. Sleep Mode Activity Overview for Peripherals**

| Peripheral | Active in Sleep Mode | | | |
|---|---|---|---|---|
| | Idle | Standby | Power-Down | |
| | | | HTLLEN=0 | HTLLEN=1 |
| CPU | | | | |
| RTC | X | X(1,2) | X(2) | X(2) |
| WDT | X | X | X | X |
| BOD | X | X | X | X |
| EVSYS | X | X | X | X |
| CCL | X | X(1) | | |
| ACn | | | | |
| ADCn | | | | |
| DACn | | | | |
| OPAMP | | | | |
| ZCDn | | | | |
| TCAn | | | | |
| TCBn | | | | |
| All other peripherals | X | | | |

**Notes:**

1. For the peripheral to run in Standby sleep mode, the RUNSTDBY bit of the corresponding peripheral must be set.
2. In Standby sleep mode, only the RTC functionality requires the RUNSTDBY bit to be set.
   In Power-Down sleep mode, only the PIT functionality is available.

**Table 13-3. Sleep Mode Activity Overview for Clock Sources**

| Clock Source | Active in Sleep Mode | | | |
|---|---|---|---|---|
| | Idle | Standby | Power-Down | |
| | | | HTLLEN=0 | HTLLEN=1 |
| Main clock source | X | X(1) | | |
| RTC clock source | X | X(1,2) | X(2) | X(2) |
| WDT oscillator | X | X | X | X |
| BOD oscillator(3) | X | X | X | X |
| CCL clock source | X | X(1) | | |
| TCD clock source | X | | | |

**Notes:**

1. For the clock source to run in Standby sleep mode, the RUNSTDBY bit of the corresponding peripheral must be set.
2. In Standby sleep mode, only the RTC functionality requires the RUNSTDBY bit to be set.
   In Power-Down sleep mode, only the PIT functionality is available.
3. The Sampled mode only.

**Table 13-4. Sleep Mode Wake-up Sources**

| Wake-Up Sources | Active in Sleep Mode | | | |
|---|---|---|---|---|
| | Idle | Standby | Power-Down | |
| | | | HTLLEN=0 | HTLLEN=1 |
| PORT Pin interrupt | X | X | X(1) | X(1) |
| BOD VLM interrupt | X | X | X | X |
| MVIO interrupts | X | X | X | X |
| RTC interrupts | X | X(2,3) | X(3) | X(3) |
| TWI Address Match interrupt | X | X | X | |
| CCL interrupts | X | X | X(4) | |
| USART Start-Of-Frame interrupt | X | X | | |
| TCAn interrupts | | | | |
| TCBn interrupts | | | | |
| ADCn interrupts | | | | |
| ACn interrupts | | | | |
| ZCD interrupts | | | | |
| All other interrupts | X | | | |

**Notes:**

1. The I/O pin must be configured according to *Asynchronous Sensing Pin Properties* in the PORT section.
2. For the peripheral to run in Standby sleep mode, the RUNSTDBY bit of the corresponding peripheral must be set.
3. In Standby sleep mode, only the RTC functionality requires the RUNSTDBY bit to be set.
   In Power-Down sleep mode, only the PIT functionality is available.
4. CCL will only wake up the device if the path through LUTn is asynchronous (FILTSEL=`0x0` and EDGEDET=`0x0` in the CCL.LUTnCTRLA register).

#### 13.3.3.2 Wake-up Time

The normal wake-up time for the device is six main clock cycles (CLK_PER), plus the time it takes to start the main clock source and the time it takes to start the regulator, if it has been switched off:

- In Idle mode, the main clock source is kept running to eliminate additional wake-up time
- In Standby mode, the main clock might be running depending on the peripheral configuration
- In Power-Down mode, only the OSC32K oscillator and the Real-Time Clock (RTC) may be running if it is used by the Brown-out Detector (BOD), Watchdog Timer (WDT) or Periodic Interrupt Timer (PIT). All the other clock sources will be OFF.

**Table 13-5. Sleep Modes and Start-up Time**

| Sleep Mode | Start-up Time |
|---|---|
| Idle | Six clock cycles |
| Standby | Six clock cycles + one (OSC start-up + Regulator start-up) |
| Power-Down | Six clock cycles + one (OSC start-up + Regulator start-up) |

The start-up time for the different clock sources is described in the *CLKCTRL - Clock Controller* section.

In addition to the normal wake-up time, it is possible to make the device wait until the BOD is ready before executing code. This is done by writing `0x3` to the BOD operation mode in Active and Idle (ACTIVE) bit field in the BOD

Configuration (FUSE.BODCFG) fuse. If the BOD is ready before the normal wake-up time, the total wake-up time will be the same. If the BOD takes longer than the normal wake-up time, the wake-up time will be extended until the BOD is ready. This ensures correct supply voltage whenever code is executed.

### 13.3.4 Debug Operation

During run-time debugging, this peripheral will continue normal operation. The SLPCTRL is only affected by a break in the debug operation: If the SLPCTRL is in a sleep mode when a break occurs, the device will wake up, and the SLPCTRL will go to Active mode, even if there are no pending interrupt requests.

If the peripheral is configured to require periodic service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

## 13.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | | | | SMODE[2:0] | | | SEN |
| 0x01 | VREGCTRL | 7:0 | | | | HTLLEN | | PMODE[2:0] | | |

## 13.5 Register Description

### 13.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SMODE[2:0] | | | SEN |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:1 – SMODE[2:0]** Sleep Mode
Writing these bits selects the desired sleep mode when the Sleep Enable (SEN) bit is written to '`1`' and the `SLEEP` instruction is executed.

| Value | Name | Description |
|---|---|---|
| `0x0` | IDLE | Idle mode enabled |
| `0x1` | STANDBY | Standby mode enabled |
| `0x2` | PDOWN | Power-Down mode enabled |
| `Other` | - | Reserved |

**Bit 0 – SEN** Sleep Enable
This bit must be written to '`1`' before the `SLEEP` instruction is executed to make the microcontroller enter the selected sleep mode.

### 13.5.2 Voltage Regulator Control Register

**Name:** VREGCTRL
**Offset:** 0x01
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | HTLLEN | | PMODE[2:0] | | |
| Access | | | | R/W | | R/W | R/W | R/W |
| Reset | | | | 0 | | 0 | 0 | 0 |

**Bit 4 – HTLLEN** High-Temperature Low Leakage Enable
This bit is write-protected.
When writing this bit to '`1`', the leakage is reduced when operating at a temperature above 70°C. This bit has an effect only in Power-Down mode when PMODE is set to AUTO. To have effect, it must be written before executing the `SLEEP` instruction.

**WARNING** This feature allows an additional power-saving feature, but the TWI address match and CCL wake-up sources must be disabled in software before setting this bit.

| Value | Name | Description |
|---|---|---|
| `0` | OFF | High-temperature low leakage disabled |
| `1` | ON | High-temperature low leakage enabled |

**Bits 2:0 – PMODE[2:0]** Power Mode Select
This bit field is write-protected.

| Value | Name | Description |
|---|---|---|
| `0x0` | AUTO | In Auto mode, the regulator will run at the full performance unless the 32.768 kHz oscillator is selected, then it runs in a lower power mode. |
| `0x1` | FULL | Full performance voltage regulator drive strength in all modes. |
| `Other` | - | Reserved |

# 14. RSTCTRL - Reset Controller

## 14.1 Features

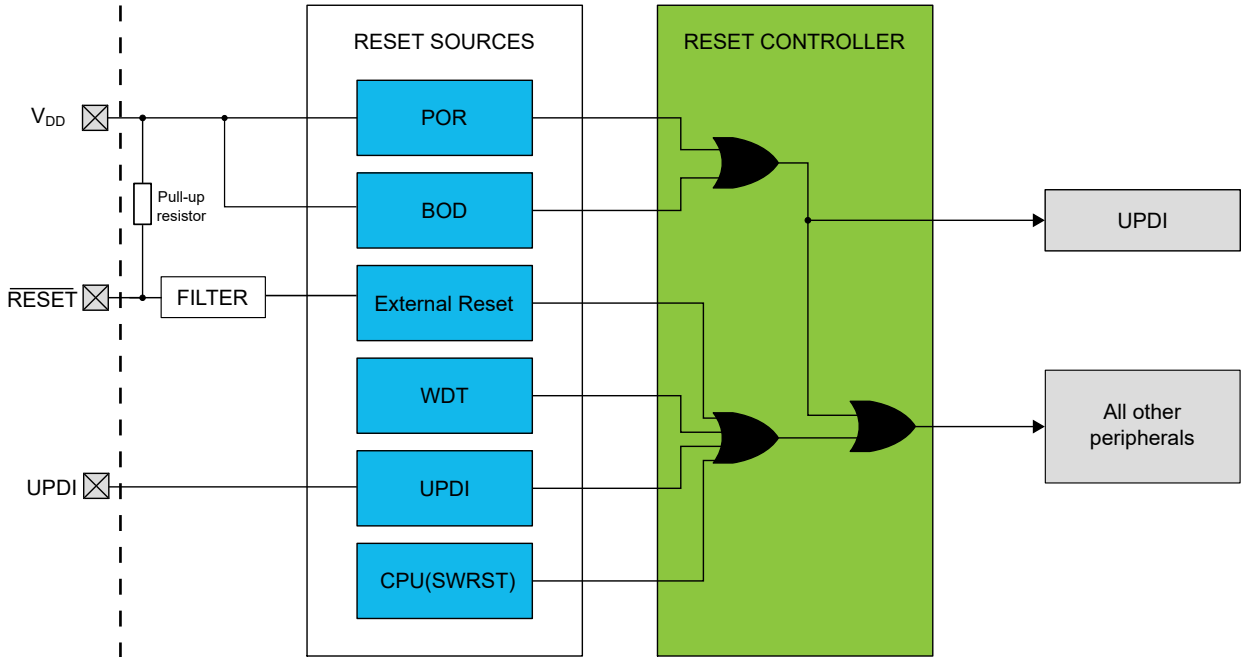
- Returns the Device to an Initial State after a Reset
- Identifies the Previous Reset Source
- Power Supply Reset Sources:
  - Power-on Reset (POR)
  - Brown-out Detector (BOD) Reset
- User Reset Sources:
  - External Reset ($\overline{\text{RESET}}$)
  - Watchdog Timer (WDT) Reset
  - Software Reset (SWRST)
  - Unified Program and Debug Interface (UPDI) Reset

## 14.2 Overview

The Reset Controller (RSTCTRL) manages the Reset of the device. When receiving a Reset request, it sets the device to an initial state and allows the Reset source to be identified by the software. The Reset controller can also be used to issue a Software Reset (SWRST).

### 14.2.1 Block Diagram

**Figure 14-1. Reset System Overview**




### 14.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| $\overline{\text{RESET}}$ | External Reset (active-low) | Digital input |

| .....continued | | |
|---|---|---|
| **Signal** | **Description** | **Type** |
| UPDI | Unified Program and Debug Interface | Digital input |

## 14.3 Functional Description

### 14.3.1 Initialization

The RSTCTRL is always enabled, but some of the Reset sources must be enabled individually (either by Fuses or by software) before they can request a Reset.

After a Reset from any source, the registers in the device with automatic loading from the Fuses or from the Signature Row are updated.

### 14.3.2 Operation

#### 14.3.2.1 Reset Sources

After any Reset, the source that caused the Reset is found in the Reset Flag (RSTCTRL.RSTFR) register. The user can identify the previous Reset source by reading this register in the software application.

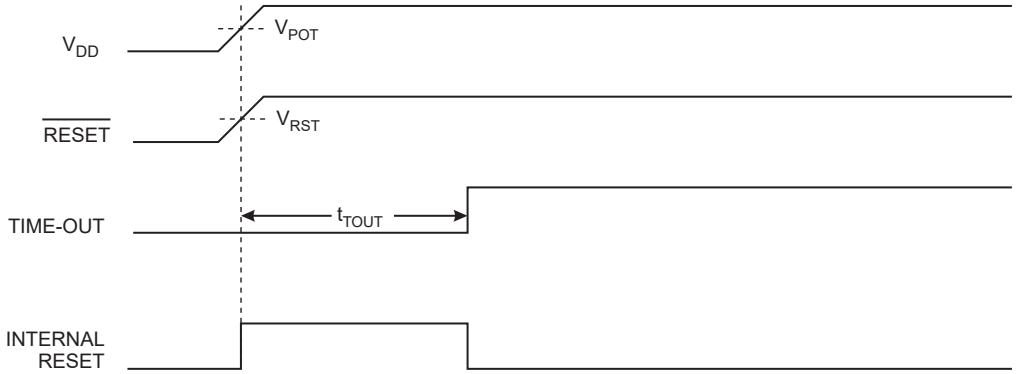
There are two types of Resets based on the source:

- Power Supply Reset Sources:
  - Power-on Reset (POR)
  - Brown-out Detector (BOD) Reset
- User Reset Sources:
  - External Reset ($\overline{\text{RESET}}$)
  - Watchdog Timer (WDT) Reset
  - Software Reset (SWRST)
  - Unified Program and Debug Interface (UPDI) Reset

##### 14.3.2.1.1 Power-on Reset (POR)

The purpose of the Power-on Reset (POR) is to ensure a safe start-up of logic and memories. It is generated by an on-chip detection circuit and is always enabled. The POR is activated when the $V_{DD}$ rises and reaches the POR threshold voltage ($V_{POT}$), starting the reset sequence. Before $V_{POT}$ has been reached, the external $\overline{\text{RESET}}$ pin keeps the device in reset, either through the internal pull-up to $V_{DD}$ or by external control.

**Figure 14-2. MCU Start-Up, $\overline{\text{RESET}}$ Tied to $V_{DD}$**

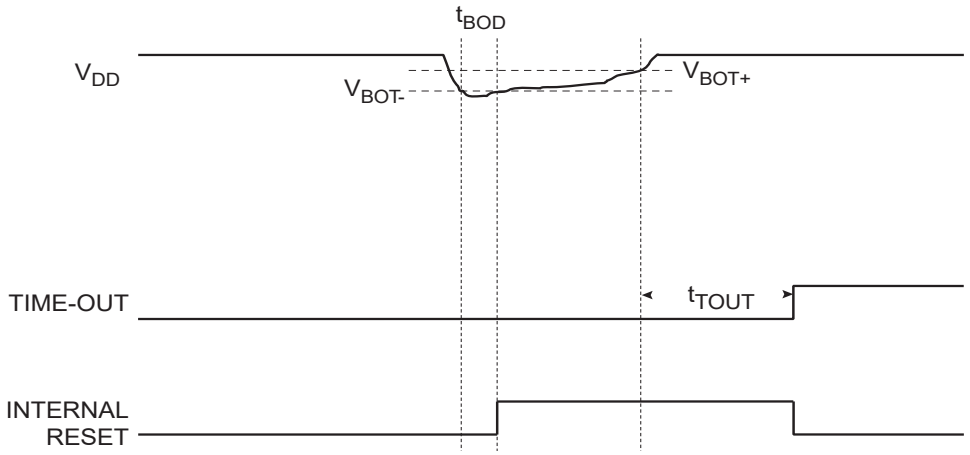



##### 14.3.2.1.2 Brown-out Detector (BOD) Reset

The Brown-out Detector (BOD) needs to be enabled by the user. The BOD is preventing code execution when the voltage drops below a set threshold. This will ensure the voltage level needed for the oscillator to run at the speed required by the application and will avoid code corruption due to low-voltage level.

The BOD issues a System Reset and is not released until the voltage level increases above the set threshold. The on-chip BOD circuit will monitor the $V_{DD}$ level during operation by comparing it to a fixed trigger level. The trigger level for the BOD must be selected by the BOD Configuration (FUSE.BODCFG) fuse.

**Figure 14-3. Brown-out Detector Reset**



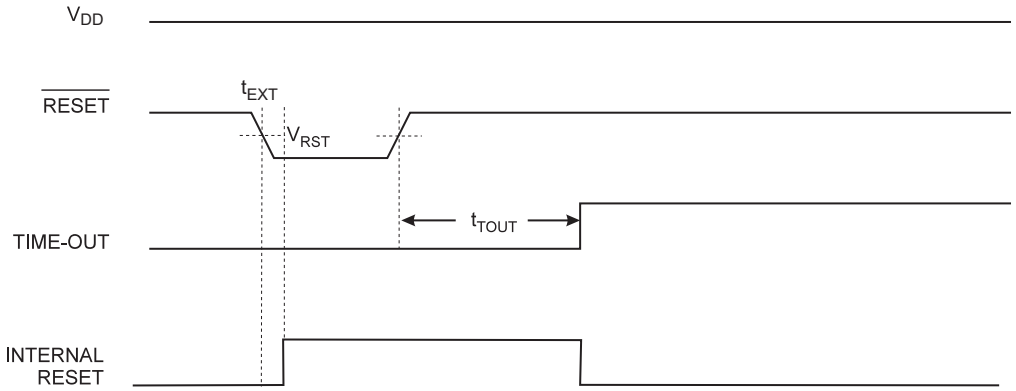

#### 14.3.2.1.3 External Reset ($\overline{\text{RESET}}$)

The $\overline{\text{RESET}}$ pin requires a noise filter that eliminates short, low-going pulses. Filtering the input assures that an external Reset event is only issued when the $\overline{\text{RESET}}$ has been low for a minimum amount of time. See the *Electrical Characteristics* section for the minimum pulse width of the $\overline{\text{RESET}}$ signal.

The external Reset is enabled by configuring the Reset Pin Configuration (RSTPINCFG) bits in the System Configuration 0 (FUSE.SYSCFG0) fuse.

When enabled, the external Reset requests a Reset as long as the $\overline{\text{RESET}}$ pin is low. The device will stay in Reset until the $\overline{\text{RESET}}$ pin is high again.

**Figure 14-4. External Reset Characteristics**




#### 14.3.2.1.4 Watchdog Timer (WDT) Reset

The Watchdog Timer (WDT) is a system function which monitors the correct operation of the program. If the WDT is not handled by software according to the programmed time-out period, a Watchdog Reset will be issued. More details can be found in the WDT section.

#### 14.3.2.1.5 Software Reset (SWRST)

The software Reset makes it possible to issue a System Reset from the software. The Reset is generated by writing a '`1`' to the Software Reset (SWRST) bit in the Software Reset (RSTCTRL.SWRR) register.

The Reset sequence will start immediately after the bit is written.

#### 14.3.2.1.6 Unified Program and Debug Interface (UPDI) Reset

The Unified Program and Debug Interface (UPDI) contains a separate Reset source used to reset the device during external programming and debugging. The Reset source is accessible only from external debuggers and programmers. More details can be found in the UPDI section.

#### 14.3.2.1.7 Domains Affected By Reset

The following logic domains are affected by the different Reset sources:

**Table 14-1. Logic Domains Affected by Various Resets**

| Reset Source | Fuses are Reloaded | Reset of BOD Configuration | Reset of UPDI | Reset of Other Volatile Logic |
|---|---|---|---|---|
| POR | X | X | X | X |
| BOD | X | | X | X |
| Software Reset | X | | | X |
| External Reset | X | | | X |
| Watchdog Reset | X | | | X |
| UPDI Reset | X | | | X |

### 14.3.2.2 Reset Time

The Reset time can be split into two parts.

The first part is when any of the Reset sources are active. This part depends on the input to the Reset sources. The external Reset is active as long as the $\overline{\text{RESET}}$ pin is low. The Power-on Reset (POR) and the Brown-out Detector (BOD) are active as long as the supply voltage is below the Reset source threshold.

The second part is when all the Reset sources are released, and an internal Reset initialization of the device is done. This time will be increased with the start-up time given by the Start-Up Time Setting (SUT) bit field in the System Configuration 1 (FUSE.SYSCFG1) fuse. The internal Reset initialization time will also increase if the Cyclic Redundancy Check Memory Scan (CRCSCAN) is configured to run at start-up. This configuration can be changed in the CRC Source (CRCSRC) bit field in the System Configuration 0 (FUSE.SYSCFG0) fuse.

## 14.3.3 Sleep Mode Operation

The RSTCTRL operates in Active mode and in all sleep modes.

## 14.3.4 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a certain key must first be written to the CPU.CCP register, followed by a write access to the protected bits within four CPU instructions.

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged.

The following registers are under CCP:

**Table 14-2. RSTCTRL - Registers Under Configuration Change Protection**

| Register | Key |
|---|---|
| RSTCTRL.SWRR | IOREG |

## 14.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | RSTFR | 7:0 | | | UPDIRF | SWRF | WDRF | EXTRF | BORF | PORF |
| 0x01 | SWRR | 7:0 | | | | | | | | SWRST |

## 14.5 Register Description

### 14.5.1 Reset Flag Register

**Name:** RSTFR
**Offset:** 0x00
**Reset:** 0xXX
**Property:** -

The Reset flags can be cleared by writing a '1' to the respective flag. All flags will be cleared by a Power-on Reset (POR), with the exception of the Power-on Reset (PORF) flag. All flags will be cleared by a Brown-out Reset (BOR), with the exception of the Power-on Reset (PORF) and Brown-out Reset (BORF) flags.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | UPDIRF | SWRF | WDRF | EXTRF | BORF | PORF |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | x | x | x | x | x | x |

**Bit 5 – UPDIRF** UPDI Reset Flag
This bit is set to '1' if a UPDI Reset has occurred.

**Bit 4 – SWRF** Software Reset Flag
This bit is set to '1' if a Software Reset has occurred.

**Bit 3 – WDRF** Watchdog Reset Flag
This bit is set to '1' if a Watchdog Reset has occurred.

**Bit 2 – EXTRF** External Reset Flag
This bit is set to '1' if an External Reset has occurred.

**Bit 1 – BORF** Brown-out Reset Flag
This bit is set to '1' if a Brown-out Reset has occurred.

**Bit 0 – PORF** Power-on Reset Flag
This bit is set to '1' if a Power-on Reset has occurred.

### 14.5.2 Software Reset Register

**Name:** SWRR
**Offset:** 0x01
**Reset:** 0x00
**Property:** Configuration Change Protection

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

**Bit 0 – SWRST** Software Reset
When this bit is written to '1', a Software Reset will occur.
This bit will always read as '0'.

# 15. CPUINT - CPU Interrupt Controller

## 15.1 Features

- Short and Predictable Interrupt Response Time
- Separate Interrupt Configuration and Vector Address for Each Interrupt
- Interrupt Prioritizing by Level and Vector Address
- Non-Maskable Interrupts (NMI) for Critical Functions
- Two Interrupt Priority Levels: 0 (Normal) and 1 (High):
  - One of the interrupt requests can optionally be assigned as a priority level 1 interrupt
  - Optional round robin priority scheme for priority level 0 interrupts
- Interrupt Vectors Optionally Placed in the Application Section or the Boot Loader Section
- Selectable Compact Vector Table (CVT)

## 15.2 Overview

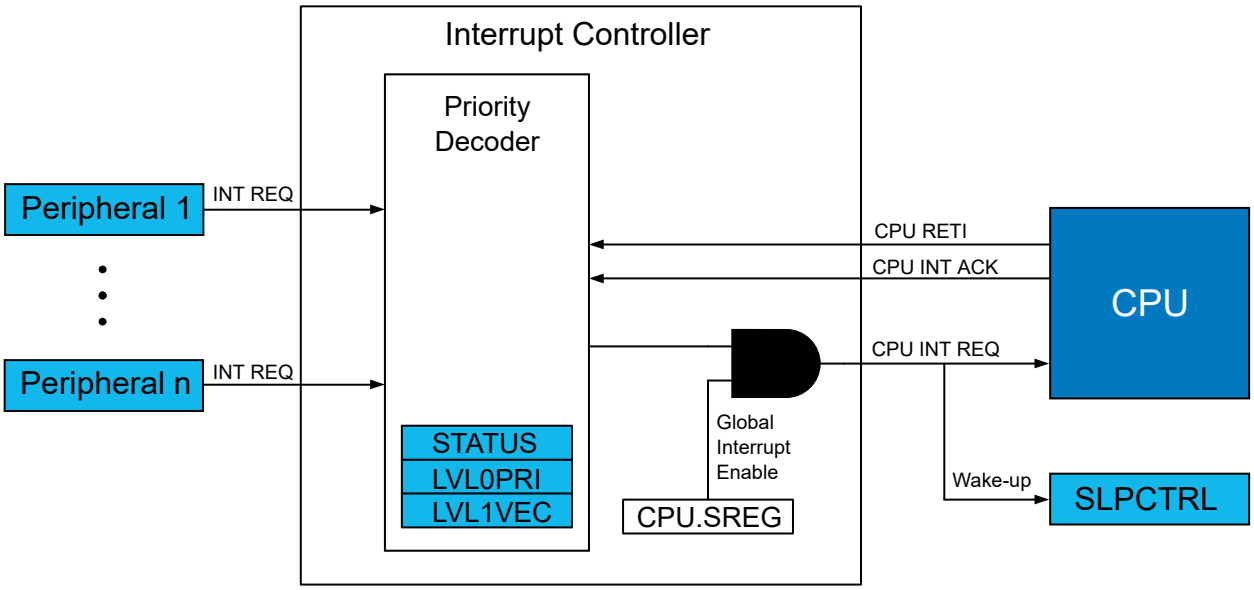
An interrupt request signals a change of state inside a peripheral and can be used to alter the program execution. The peripherals can have one or more interrupts. All interrupts are individually enabled and configured. When an interrupt is enabled and configured, it will generate an interrupt request when the interrupt condition occurs.

The CPU Interrupt Controller (CPUINT) handles and prioritizes the interrupt requests. When an interrupt is enabled and the interrupt condition occurs, the CPUINT will receive the interrupt request. Based on the interrupt's priority level and the priority level of any ongoing interrupt, the interrupt request is either acknowledged or kept pending until it has priority. After returning from the interrupt handler, the program execution continues from where it was before the interrupt occurred, and any pending interrupts are served after one instruction is executed.

The CPUINT offers NMI for critical functions, one selectable high-priority interrupt and an optional round robin scheduling scheme for normal-priority interrupts. The round robin scheduling ensures that all interrupts are serviced within a certain amount of time.

### 15.2.1 Block Diagram

**Figure 15-1. CPUINT Block Diagram**



## 15.3 Functional Description

### 15.3.1 Initialization

An interrupt must be initialized in the following order:

1. Configure the CPUINT if the default configuration is not adequate (optional):
   - Vector handling is configured by writing to the respective bits (IVSEL and CVT) in the Control A (CPUINT.CTRLA) register.
   - Vector prioritizing by round robin is enabled by writing a '1' to the Round Robin Priority Enable (LVL0RR) bit in CPUINT.CTRLA.
   - Select the Priority Level 1 vector by writing the interrupt vector number to the Interrupt Vector with Priority Level 1 (CPUINT.LVL1VEC) register.
2. Configure the interrupt conditions within the peripheral and enable the peripheral's interrupt.
3. Enable interrupts globally by writing a '1' to the Global Interrupt Enable (I) bit in the CPU Status (CPU.SREG) register.

### 15.3.2 Operation

#### 15.3.2.1 Enabling, Disabling and Resetting

The global enabling of interrupts is done by writing a '1' to the Global Interrupt Enable (I) bit in the CPU Status (CPU.SREG) register. To disable interrupts globally, write a '0' to the I bit in CPU.SREG.

The desired interrupt lines must also be enabled in the respective peripheral by writing to the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

The interrupt flags are not automatically cleared after the interrupt is executed. The respective INTFLAGS register descriptions provide information on how to clear specific flags.

#### 15.3.2.2 Interrupt Vector Locations

The interrupt vector placement is dependent on the value of the Interrupt Vector Select (IVSEL) bit in the Control A (CPUINT.CTRLA) register. Refer to the IVSEL description in CPUINT.CTRLA for the possible locations.

If the program never enables an interrupt source, the interrupt vectors are not used, and the regular program code can be placed at these locations.

#### 15.3.2.3 Interrupt Response Time

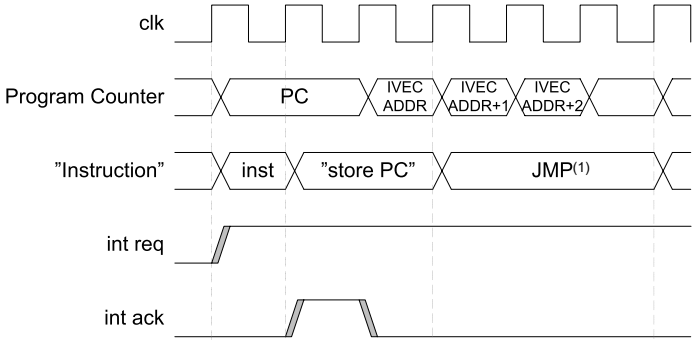
The minimum interrupt response time is represented in the following table.

**Table 15-1. Minimum Interrupt Response Time**

| | Flash Size > 8 KB | Flash Size ≤ 8 KB |
|---|---|---|
| Finish ongoing instruction | One cycle | One cycle |
| Store PC to stack | Two cycles | Two cycles |
| Jump to interrupt handler | Three cycles (`jmp`) | Two cycles (`rjmp`) |

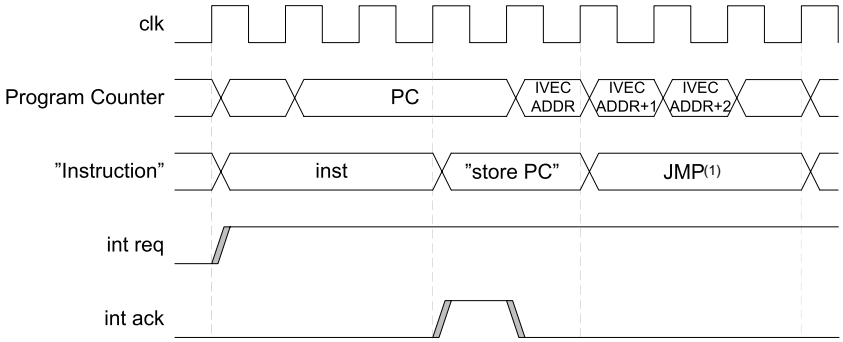
After the Program Counter is pushed on the stack, the program vector for the interrupt is executed. See the following figure.

**Figure 15-2. Interrupt Execution of Single-Cycle Instruction**



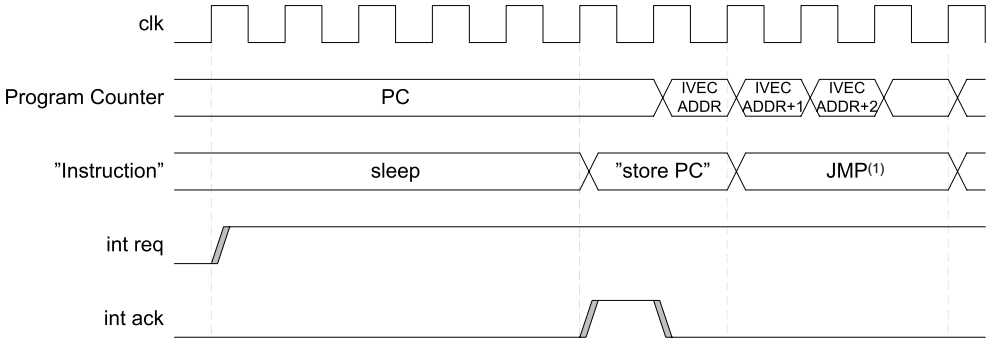

If an interrupt occurs during the execution of a multi-cycle instruction, the instruction is completed before the interrupt is served, as shown in the following figure.

**Figure 15-3. Interrupt Execution of Multi-Cycle Instruction**




If an interrupt occurs when the device is in a sleep mode, the interrupt execution response time is increased by five clock cycles, as shown in the figure below. Also, the response time is increased by the start-up time from the selected sleep mode.

**Figure 15-4. Interrupt Execution From Sleep**




A return from an interrupt handling routine takes four to five clock cycles, depending on the size of the Program Counter. During these clock cycles, the Program Counter is popped from the stack, and the Stack Pointer is incremented.

**Note:**

1. Devices with 8 KB of Flash or less use `RJMP` instead of `JMP`, which takes only two clock cycles.

### 15.3.2.4 Interrupt Priority

All interrupt vectors are assigned to one of three possible priority levels, as shown in the table below. An interrupt request from a high-priority source will interrupt any ongoing interrupt handler from a normal-priority source. When returning from the high-priority interrupt handler, the execution of the normal-priority interrupt handler will resume.

**Table 15-2. Interrupt Priority Levels**

| Priority | Level | Source |
|---|---|---|
| Highest | Non-Maskable Interrupt | Device-dependent and statically assigned |
| ... | Level 1 (high priority) | One vector is optionally user selectable as level 1 |
| Lowest | Level 0 (normal priority) | The remaining interrupt vectors |

#### 15.3.2.4.1 Non-Maskable Interrupts

A Non-Maskable Interrupt (NMI) will be executed regardless of the setting of the I bit in CPU.SREG. An NMI will never change the I bit. No other interrupt can interrupt an NMI handler. If more than one NMI is requested at the same time, the priority is static according to the interrupt vector address, where the lowest address has the highest priority.

Which interrupts are non-maskable is device-dependent and not subject to configuration. Non-maskable interrupts must be enabled before they can be used. Refer to the interrupt vector mapping of the device for available NMI lines.

#### 15.3.2.4.2 High-Priority Interrupt

It is possible to assign one interrupt request to level 1 (high priority) by writing its interrupt vector number to the CPUINT.LVL1VEC register. This interrupt request will have a higher priority than the other (normal priority) interrupt requests. The priority level 1 interrupts will interrupt the level 0 interrupt handlers.

#### 15.3.2.4.3 Normal-Priority Interrupts

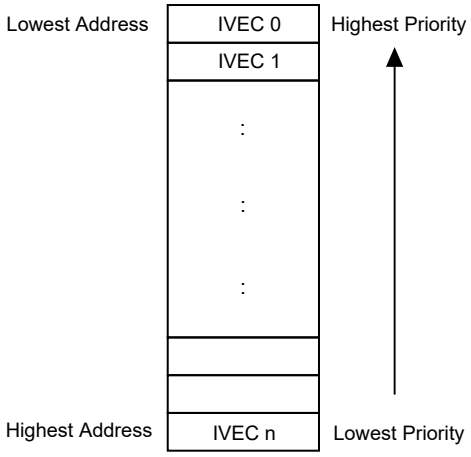
All interrupt vectors other than NMI are assigned to priority level 0 (normal) by default. The user may override this by assigning one of these vectors as a high-priority vector. The device will have many normal-priority vectors, and some of these may be pending at the same time. Two different scheduling schemes are available to choose which of the pending normal-priority interrupts to service first: Static or round robin.

IVEC is the interrupt vector mapping, as listed in the *Peripherals and Architecture* chapter. The following sections use IVEC to explain the scheduling schemes. IVEC0 is the Reset vector, IVEC1 is the NMI vector, and so on. In a vector table with n+1 elements, the vector with the highest vector number is denoted IVECn. Reset, non-maskable interrupts and high-level interrupts are included in the IVEC map, but will always be prioritized over the normal-priority interrupts.

***Static Scheduling***

If several level 0 interrupt requests are pending at the same time, the one with the highest priority is scheduled for execution first. The following figure illustrates the default configuration, where the interrupt vector with the lowest address has the highest priority.

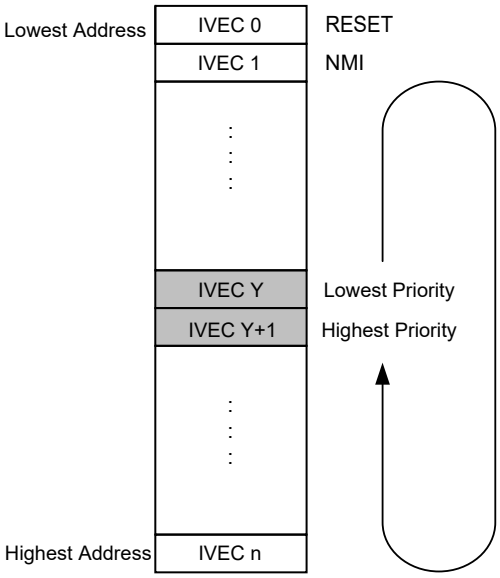
**Figure 15-5. Default Static Scheduling**




***Modified Static Scheduling***

The default priority can be changed by writing a vector number to the CPUINT.LVL0PRI register. This vector number will be assigned the lowest priority. The next interrupt vector in the IVEC will have the highest priority among the LVL0 interrupts, as shown in the following figure.

**Figure 15-6. Static Scheduling when CPUINT.LVL0PRI is Different From Zero**



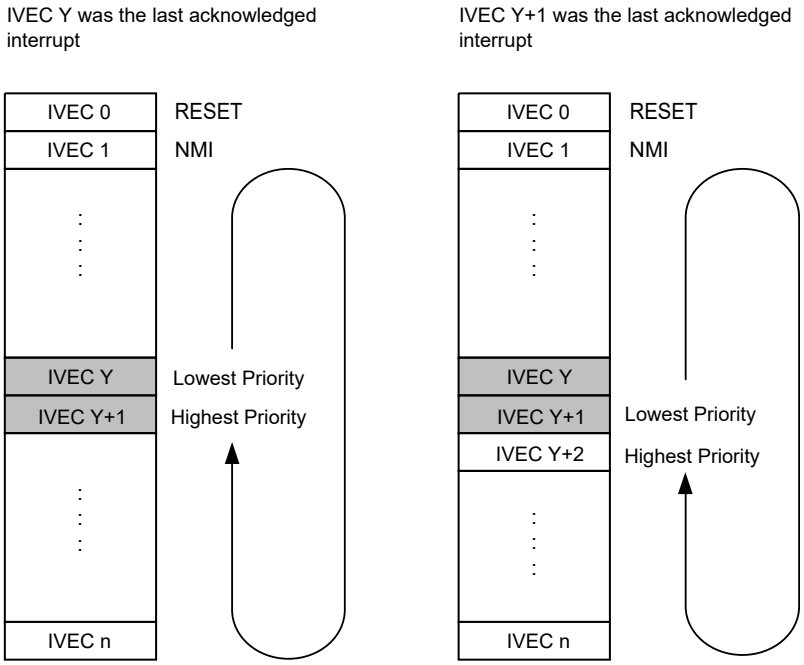

Here, value Y has been written to CPUINT.LVL0PRI, so that interrupt vector Y+1 has the highest priority. Note that, in this case, the priorities will wrap so that the lowest address no longer has the highest priority. This does not include RESET and NMI, which will always have the highest priority.

Refer to the interrupt vector mapping of the device for available interrupt requests and their interrupt vector number.

***Round Robin Scheduling***

The static scheduling may prevent some interrupt requests from being serviced. To avoid this, the CPUINT offers round robin scheduling for normal-priority (LVL0) interrupts. In the round robin scheduling, the CPUINT.LVL0PRI register stores the last acknowledged interrupt vector number. This register ensures that the last acknowledged interrupt vector gets the lowest priority and is automatically updated by the hardware. The following figure illustrates the priority order after acknowledging IVEC Y and after acknowledging IVEC Y+1.

**Figure 15-7. Round Robin Scheduling**




The round robin scheduling for LVL0 interrupt requests is enabled by writing a '1' to the Round Robin Priority Enable (LVL0RR) bit in the Control A (CPUINT.CTRLA) register.

#### 15.3.2.5 Compact Vector Table

The Compact Vector Table (CVT) is a feature to allow writing of compact code by having all level 0 interrupts share the same interrupt vector number. Thus, the interrupts share the same Interrupt Service Routine (ISR). This reduces the number of interrupt handlers and thereby frees up memory that can be used for the application code.

When CVT is enabled by writing a '`1`' to the CVT bit in the Control A (CPUINT.CTRLA) register, the vector table contains these three interrupt vectors:

1. The non-maskable interrupts (NMI) at vector address 1.
2. The Priority Level 1 (LVL1) interrupt at vector address 2.
3. All priority level 0 (LVL0) interrupts at vector address 3.

This feature is most suitable for devices with limited memory and applications using a small number of interrupt generators.

### 15.3.3 Debug Operation

When using a level 1 priority interrupt, it is important to make sure the Interrupt Service Routine is configured correctly as it may cause the application to be stuck in an interrupt loop with level 1 priority.

By reading the CPUINT STATUS (CPUINT.STATUS) register, it is possible to see if the application has executed the correct `RETI` (interrupt return) instruction. The CPUINT.STATUS register contains state information, which ensures that the CPUINT returns to the correct interrupt level when the `RETI` instruction is executed at the end of an interrupt handler. Returning from an interrupt will return the CPUINT to the state it had before entering the interrupt.

### 15.3.4 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a certain key must first be written to the CPU.CCP register, followed by a write access to the protected bits within four CPU instructions.

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged.

The following registers are under CCP:

**Table 15-3. CPUINT - Registers under Configuration Change Protection**

| Register | Key |
|---|---|
| IVSEL in CPUINT.CTRLA | IOREG |
| CVT in CPUINT.CTRLA | IOREG |

## 15.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | IVSEL | CVT | | | | | LVL0RR |
| 0x01 | STATUS | 7:0 | NMIEX | | | | | | LVL1EX | LVL0EX |
| 0x02 | LVL0PRI | 7:0 | LVL0PRI[7:0] | | | | | | | |
| 0x03 | LVL1VEC | 7:0 | LVL1VEC[7:0] | | | | | | | |

## 15.5 Register Description

### 15.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | IVSEL | CVT | | | | | LVL0RR |
| Access | | R/W | R/W | | | | | R/W |
| Reset | | 0 | 0 | | | | | 0 |

**Bit 6 – IVSEL** Interrupt Vector Select
This bit is protected by the Configuration Change Protection mechanism.

| Value | Description |
|---|---|
| 0 | Interrupt vectors are placed at the start of the application section of the Flash |
| 1 | Interrupt vectors are placed at the start of the boot section of the Flash |

**Bit 5 – CVT** Compact Vector Table
This bit is protected by the Configuration Change Protection mechanism.

| Value | Description |
|---|---|
| 0 | Compact Vector Table function is disabled |
| 1 | Compact Vector Table function is enabled |

**Bit 0 – LVL0RR** Round Robin Priority Enable
This bit is not protected by the Configuration Change Protection mechanism.

| Value | Description |
|---|---|
| 0 | Priority is fixed for priority level 0 interrupt requests: The lowest interrupt vector address has the highest priority |
| 1 | The round robin priority scheme is enabled for priority level 0 interrupt requests |

### 15.5.2 Status

**Name:** STATUS
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NMIEX | | | | | | LVL1EX | LVL0EX |
| Access | R | | | | | | R | R |
| Reset | 0 | | | | | | 0 | 0 |

**Bit 7 – NMIEX** Non-Maskable Interrupt Executing
This flag is set if a non-maskable interrupt is executing. The flag is cleared when returning (RETI) from the interrupt handler.

**Bit 1 – LVL1EX** Level 1 Interrupt Executing
This flag is set when a priority level 1 interrupt is executing, or when the interrupt handler has been interrupted by an NMI. The flag is cleared when returning (RETI) from the interrupt handler.

**Bit 0 – LVL0EX** Level 0 Interrupt Executing
This flag is set when a priority level 0 interrupt is executing, or when the interrupt handler has been interrupted by a priority level 1 interrupt or an NMI. The flag is cleared when returning (RETI) from the interrupt handler.

### 15.5.3 Interrupt Priority Level 0

**Name:** LVL0PRI
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LVL0PRI[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 – LVL0PRI[7:0]** Interrupt Priority Level 0
This register is used to modify the priority of the LVL0 interrupts. See the section Normal-Priority Interrupts for more information.

### 15.5.4 Interrupt Vector with Priority Level 1

**Name:** LVL1VEC
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LVL1VEC[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 – LVL1VEC[7:0]** Interrupt Vector with Priority Level 1
This bit field contains the number of the single vector with increased priority level 1 (LVL1). If this bit field has the value `0x00`, no vector has LVL1. Consequently, the LVL1 interrupt is disabled.

# 16. EVSYS - Event System

## 16.1 Features

- System for Direct Peripheral-to-Peripheral Signaling
- Peripherals Can Directly Produce, Use, and React to Peripheral Events
- Short and Predictable Response Time
- Up to 10 Parallel Event Channels Available
- Each Channel is Driven by One Event Generator and Can Have Multiple Event Users
- Events Can be Sent and/or Received by Most Peripherals and by Software
- The Event System Works in Active, Idle, and Standby Sleep Modes

## 16.2 Overview

The Event System (EVSYS) enables direct peripheral-to-peripheral signaling. It allows a change in one peripheral (the event generator) to trigger actions in other peripherals (the event users) through event channels, without using the CPU. It is designed to provide a short and predictable response time between peripherals, allowing for autonomous peripheral control and interaction, and for synchronized timing of actions in several peripheral modules. Thus, it is a powerful tool for reducing the complexity, size, and execution time of the software.

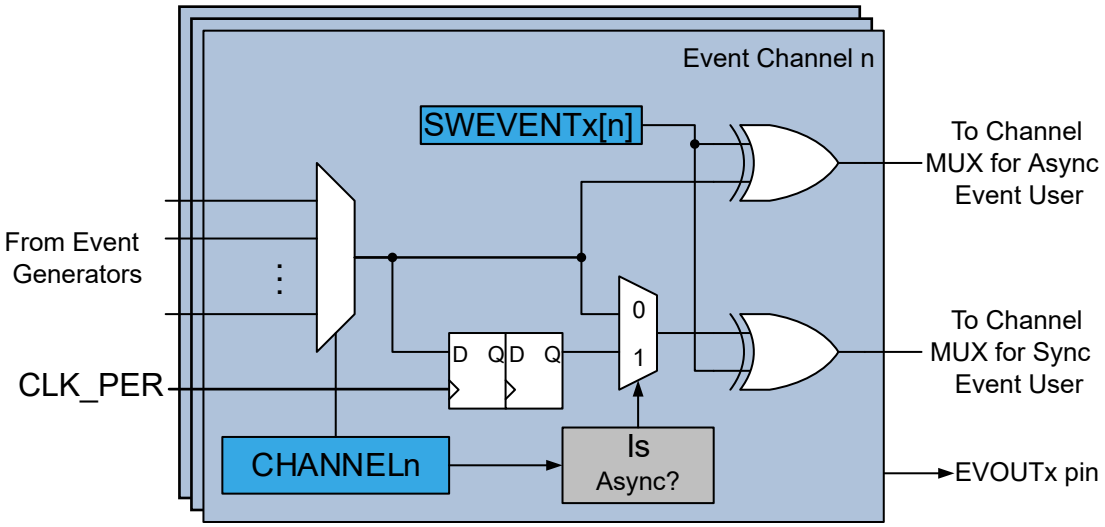
A change of the event generator's state is referred to as an event and usually corresponds to one of the peripheral's interrupt conditions. Events can be forwarded directly to other peripherals using the dedicated event routing network. The routing of each channel is configured in software, including event generation and use.

Only one event signal can be routed on each channel. Multiple peripherals can use events from the same channel.

The EVSYS can connect peripherals such as ADCs, analog comparators, I/O PORT pins, the real-time counter, timer/counters, and the configurable custom logic peripheral. Events can also be generated from software.

### 16.2.1 Block Diagram

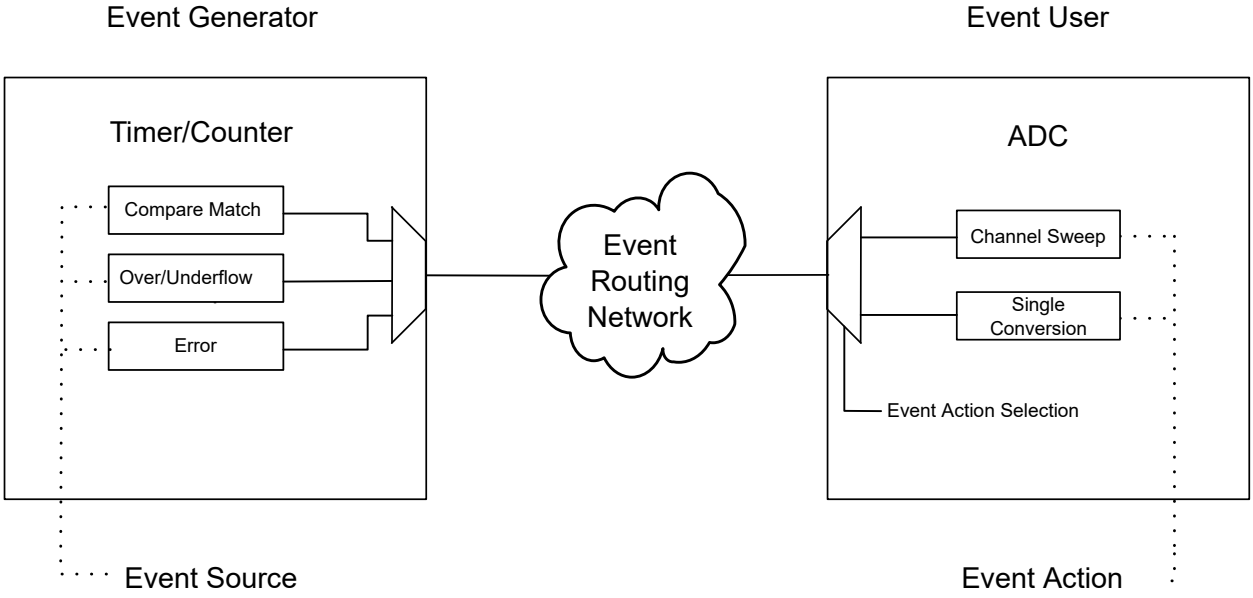
**Figure 16-1. Block Diagram**




The block diagram shows the operation of an event channel. A multiplexer controlled by Channel n Generator Selection (EVSYS.CHANNELn) register at the input selects which of the event sources to route onto the event channel. Each event channel has two subchannels: one asynchronous and one synchronous. A synchronous user will listen to the synchronous subchannel, and an asynchronous user will listen to the asynchronous subchannel.

An event signal from an asynchronous source will be synchronized by the Event System before being routed to the synchronous subchannel. An asynchronous event signal to be used by a synchronous consumer must last for at least one peripheral clock cycle to ensure that it will propagate through the synchronizer. The synchronizer will delay such an event between two and three clock cycles, depending on when the event occurs.

**Figure 16-2. Example of Event Source, Generator, User, and Action**




### 16.2.2 Signal Description

| Signal | Type | Description |
|---|---|---|
| EVOUTx | Digital output | Event output, one output per I/O Port |

## 16.3 Functional Description

### 16.3.1 Initialization

To utilize events, the Event System, the generating peripheral, and the peripheral(s) using the event must be set up accordingly:

1. Configure the generating peripheral appropriately. For example, if the generating peripheral is a timer, set the prescaling, the Compare register, etc., so that the desired event is generated.
2. Configure the event user peripheral(s) appropriately. For example, if the ADC is the event user, set the ADC prescaler, resolution, conversion time, etc., as desired, and configure the ADC conversion to start at the reception of an event.
3. Configure the Event System to route the desired source. In this case, the Timer/Compare match to the desired event channel. This may, for example, be Channel 0, which is accomplished by writing to the Channel 0 Generator Selection (EVSYS.CHANNEL0) register.
4. Configure the ADC to listen to this channel by writing to the corresponding User x Channel MUX (EVSYS.USERx) register.

### 16.3.2 Operation

#### 16.3.2.1 Event User Multiplexer Setup

Each event user has one dedicated event user multiplexer selecting which event channel to listen to. The application configures these multiplexers by writing to the corresponding EVSYS.USERx register.

#### 16.3.2.2 Event System Channel

An event channel can be connected to one of the event generators.

The source for each event channel is configured by writing to the respective Channel n Generator Selection (EVSYS.CHANNELn) register.

#### 16.3.2.3 Event Generators

Each event channel has several possible event generators, but only one can be selected at a time. The event generator for a channel is selected by writing to the respective Channel n Generator Selection (EVSYS.CHANNELn) register. By default, the channels are not connected to any event generator. For details on event generation, refer to the documentation of the corresponding peripheral.

A generated event is either synchronous or asynchronous to the device peripheral clock (CLK_PER). Asynchronous events can be generated outside the normal edges of the peripheral clock, making the system respond faster than the selected clock frequency would suggest. Asynchronous events can also be generated while the device is in a Sleep mode when the peripheral clock is not running.

Any generated event is classified as either a pulse event or a level event. In both cases, the event can be either synchronous or asynchronous, with properties according to the table below.

**Table 16-1. Properties of Generated Events**

| Event Type | Sync/Async | Description |
|---|---|---|
| Pulse | Sync | An event generated from CLK_PER that lasts one clock cycle |
| | Async | An event generated from a clock other than CLK_PER lasting one clock cycle |
| Level | Sync | An event generated from CLK_PER that lasts multiple clock cycles |
| | Async | An event generated without a clock (for example, a pin or a comparator), or an event generated from a clock other than CLK_PER that lasts multiple clock cycles |

The properties of both the generated event and the intended event user must be considered in order to ensure reliable and predictable operation.

The table below shows the available event generators for this device family.

**Table 16-2. Event Generators**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| UPDI | SYNCH | SYNCH character | Level | CLK_PDI | SYNCH character on PDI RX input synchronized to CLK_PDI |
| MVIO | VDDIO2OK | VDDIO2 is OK | Level | Asynchronous | High as long as VDDIO2 is OK |

.....continued

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| RTC | OVF | Overflow | Pulse | CLK_RTC | One CLK_RTC period |
| | CMP | Compare Match | | | |
| | PIT_DIV8192 | Prescaled RTC clock divided by 8192 | Level | | Given by prescaled RTC clock divided by 8192 |
| | PIT_DIV4096 | Prescaled RTC clock divided by 4096 | | | Given by prescaled RTC clock divided by 4096 |
| | PIT_DIV2048 | Prescaled RTC clock divided by 2048 | | | Given by prescaled RTC clock divided by 2048 |
| | PIT_DIV1024 | Prescaled RTC clock divided by 1024 | | | Given by prescaled RTC clock divided by 1024 |
| | PIT_DIV512 | Prescaled RTC clock divided by 512 | | | Given by prescaled RTC clock divided by 512 |
| | PIT_DIV256 | Prescaled RTC clock divided by 256 | | | Given by prescaled RTC clock divided by 256 |
| | PIT_DIV128 | Prescaled RTC clock divided by 128 | | | Given by prescaled RTC clock divided by 128 |
| | PIT_DIV64 | Prescaled RTC clock divided by 64 | | | Given by prescaled RTC clock divided by 64 |
| CCL | LUTn | LUT output level | Level | Asynchronous | Depends on CCL configuration |
| ACn | OUT | Comparator output level | Level | Asynchronous | Given by AC output level |
| ADCn | RESRDY | Result ready | Pulse | CLK_PER | One CLK_PER period |
| ZCDn | OUT | ZCD output level | Level | Asynchronous | Given by ZCD output level |
| OPAMPn | READY | Op amp ready | Pulse | CLK_PER | One CLK_PER period |
| PORTx | PINn | Pin level | Level | Asynchronous | Given by pin level |
| USARTn | XCK | USART Baud clock | Level | CLK_PER | Minimum two CLK_PER periods |
| SPIn | SCK | SPI Master clock | Level | CLK_PER | Minimum two CLK_PER periods |
| TCAn | OVF_LUNF | Overflow/Low byte timer underflow | Pulse | CLK_PER | One CLK_PER period |
| | HUNF | High byte timer underflow | | | |
| | CMP0_LCMP0 | Compare channel 0 match/Low byte timer compare channel 0 match | | | |
| | CMP1_LCMP1 | Compare channel 1 match/Low byte timer compare channel 1 match | | | |
| | CMP2_LCMP2 | Compare channel 2 match/Low byte timer compare channel 2 match | | | |

.....continued

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| TCBn | CAPT | CAPT flag set | Pulse | CLK_PER | One CLK_PER period |
| | OVF | Overflow | | | |
| TCDn | CMPBCLR | Counter matches CMPBCLR | Pulse | CLK_TCD | One CLK_TCD period |
| | CMPASET | Counter matches CMPASET | | | |
| | CMPBSET | Counter matches CMPBSET | | | |
| | PROGEV | Programmable event output | | | |

#### 16.3.2.4 Event Users

The event channel to listen to is selected by configuring the event user. An event user may require the event signal to be either synchronous or asynchronous to the peripheral clock. An asynchronous event user can respond to events in Sleep modes when clocks are not running. Such events can be responded to outside the normal edges of the peripheral clock, making the event user respond faster than the clock frequency would suggest. For details on the requirements of each peripheral, refer to the documentation of the corresponding peripheral.

Most event users implement edge or level detection to trigger actions in the corresponding peripheral based on the incoming event signal. In both cases, a user can either be synchronous, which requires that the incoming event is generated from the peripheral clock (CLK_PER), or asynchronous, if not. Some asynchronous event users do not apply event input detection but use the event signal directly. The different event user properties are described in general in the table below.

**Table 16-3. Properties of Event Users**

| Input Detection | Async/Sync | Description |
|---|---|---|
| Edge | Sync | An event user is triggered by an event edge and requires that the incoming event is generated from CLK_PER. |
| | Async | An event user is triggered by an event edge and has asynchronous detection or an internal synchronizer. |
| Level | Sync | An event user is triggered by an event level and requires that the incoming event is generated from CLK_PER. |
| | Async | An event user is triggered by an event level and has asynchronous detection or an internal synchronizer. |
| No detection | Async | An event user will use the event signal directly. |

The table below shows the available event users for this device family.

**Table 16-4. Event Users**

| USER Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| CCL | LUTnx | LUTn input x or clock signal | No detection | Async |
| ADCn | START | ADC start on event | Edge | Async |
| EVSYS | EVOUTx | Forward event signal to pin | No detection | Async |

.....continued

| USER Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| USARTn | IRDA | IrDA mode input | Level | Sync |
| TCAn | CNTA | Count on positive event edge | Edge | Sync |
| | | Count on any event edge | Edge | |
| | | Count while event signal is high | Level | |
| | | Event level controls count direction | Level | |
| | CNTB | Event level controls count direction | Level | Sync |
| | | Restart counter on positive event edge | Edge | |
| | | Restart counter on any event edge | Edge | |
| | | Restart counter while event signal is high | Level | |
| TCBn | CAPT | Time-out check | Edge | Sync |
| | | Input capture on event | Edge | |
| | | Input capture frequency measurement | Edge | |
| | | Input capture pulse-width measurement | Edge | |
| | | Input capture frequency and pulse-width measurement | Edge | |
| | | Single-shot | Edge | Both |
| | COUNT | Count on event | Edge | Sync |
| TCDn | INPUTA | Fault or capture | Level or edge | Async |
| | INPUTB | | | |
| OPAMPn | ENABLE | Enable OPAMP | Edge | Async |
| | DISABLE | Disable OPAMP | Edge | Sync |
| | DUMP | OPAMP $V_{OUT}$ Dump for Integrator Mode | Level | Sync |
| | DRIVE | OPAMP Drive in Normal Mode | Level | Sync |

#### 16.3.2.5 Synchronization

Events can be either synchronous or asynchronous to the peripheral clock. Each Event System channel has two subchannels: one asynchronous and one synchronous.

The asynchronous subchannel is identical to the event output from the generator. If the event generator generates a signal asynchronous to the peripheral clock, the signal on the asynchronous subchannel will be asynchronous. If the event generator generates a signal synchronous to the peripheral clock, the signal on the asynchronous subchannel will also be synchronous.

The synchronous subchannel is identical to the event output from the generator, if the event generator generates a signal synchronous to the peripheral clock. If the event generator generates a signal asynchronous to the peripheral clock, this signal is first synchronized before being routed onto the synchronous subchannel. Depending on when it occurs, synchronization will delay the event by two to three clock cycles. The Event System automatically performs this synchronization if an asynchronous generator is selected for an event channel.

#### 16.3.2.6 Software Event

The application can generate a software event. Software events on Channel n are issued by writing a '1' to the Software Event Channel Select (CHANNEL[n]) bit in the Software Events (EVSYS.SWEVENTx) register. A software event appears as a pulse on the Event System channel, inverting the current event signal for one clock cycle.

Event users see software events as no different from those produced by event generating peripherals.

### 16.3.3 Sleep Mode Operation

When configured, the Event System will work in all Sleep modes. Software events represent one exception since they require a peripheral clock.

Asynchronous event users are able to respond to an event without their clock running in Standby Sleep mode. Synchronous event users require their clock to be running to be able to respond to events. Such users will only work in Idle Sleep mode or in Standby Sleep mode, if configured to run in Standby mode by setting the RUNSTDBY bit in the appropriate register.

Asynchronous event generators are able to generate an event without their clock running, that is, in Standby Sleep mode. Synchronous event generators require their clock to be running to be able to generate events. Such generators will only work in Idle Sleep mode or in Standby Sleep mode, if configured to run in Standby mode by setting the RUNSTDBY bit in the appropriate register.

### 16.3.4 Debug Operation

This peripheral is unaffected by entering Debug mode.

## 16.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | SWEVENTA | 7:0 | SWEVENTA[7:0] | | | | | | | |
| 0x01 | SWEVENTB | 7:0 | SWEVENTB[7:0] | | | | | | | |
| 0x02<br>...<br>0x0F | Reserved | | | | | | | | | |
| 0x10 | CHANNEL0 | 7:0 | CHANNEL0[7:0] | | | | | | | |
| 0x11 | CHANNEL1 | 7:0 | CHANNEL1[7:0] | | | | | | | |
| 0x12 | CHANNEL2 | 7:0 | CHANNEL2[7:0] | | | | | | | |
| 0x13 | CHANNEL3 | 7:0 | CHANNEL3[7:0] | | | | | | | |
| 0x14 | CHANNEL4 | 7:0 | CHANNEL4[7:0] | | | | | | | |
| 0x15 | CHANNEL5 | 7:0 | CHANNEL5[7:0] | | | | | | | |
| 0x16 | CHANNEL6 | 7:0 | CHANNEL6[7:0] | | | | | | | |
| 0x17 | CHANNEL7 | 7:0 | CHANNEL7[7:0] | | | | | | | |
| 0x18 | CHANNEL8 | 7:0 | CHANNEL8[7:0] | | | | | | | |
| 0x19 | CHANNEL9 | 7:0 | CHANNEL9[7:0] | | | | | | | |
| 0x1A<br>...<br>0x1F | Reserved | | | | | | | | | |
| 0x20 | USERCCLLUT0A | 7:0 | USER[7:0] | | | | | | | |
| ... | | | | | | | | | | |
| 0x55 | USEROPAMP2DRIVE | 7:0 | USER[7:0] | | | | | | | |

## 16.5 Register Description

### 16.5.1 Software Events

**Name:** SWEVENTx
**Offset:** 0x00 + x*0x01 [x=0..1]
**Reset:** 0x00
**Property:** -

Write bits in this register to create a software event on the corresponding event channels.
Bits 0-7 in the EVSYS.SWEVENTA register correspond to event channels 0-7. If the number of available event channels is between eight and 15, these are available in the EVSYS.SWEVENTB register, where bit n corresponds to event channel 8+n.

Refer to the *Peripheral Overview* section for the available number of Event System channels.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SWEVENTx[7:0] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – SWEVENTx[7:0]** Software Event Channel Select
Writing a bit in this bit group to '1' will generate a single-pulse event on the corresponding event channel by inverting the signal on the event channel for one peripheral clock cycle.

### 16.5.2 Channel n Generator Selection

**Name:** CHANNELn
**Offset:** 0x10 + n*0x01 [n=0..9]
**Reset:** 0x00
**Property:** -

Each channel can be connected to one event generator. Not all generators can be connected to all channels. Refer to the table below to see which generator sources can be routed onto each channel and the generator value to be written to EVSYS.CHANNELn to achieve this routing. Writing the value 0x00 to EVSYS.CHANNELn turns the channel off.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CHANNELn[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 – CHANNELn[7:0]** Channel Generator Selection
The specific generator name corresponding to each bit group configuration is given by combining Peripheral and Output from the table below in the following way: PERIPHERAL_OUTPUT.

| Generator | | | Async/Sync | Description | Channel Availability |
|---|---|---|---|---|---|
| Value | Name | | | | |
| | Peripheral | Output | | | |
| 0x01 | UPDI | SYNCH | Sync | Rising edge of SYNCH character detection | All channels |
| 0x05 | MVIO | VDDIO2OK | Async | VDDIO2 OK | All Channels |
| 0x06 | RTC | OVF | Async | Counter overflow | All channels |
| 0x07 | | CMP | | Compare match | |
| 0x08 | | PIT_DIV8192 | | Prescaled RTC clock divided by 8192 | Even numbered channels only |
| 0x09 | | PIT_DIV4096 | | Prescaled RTC clock divided by 4096 | |
| 0x0A | | PIT_DIV2048 | | Prescaled RTC clock divided by 2048 | |
| 0x0B | | PIT_DIV1024 | | Prescaled RTC clock divided by 1024 | |
| 0x08 | | PIT_DIV512 | | Prescaled RTC clock divided by 512 | Odd numbered channels only |
| 0x09 | | PIT_DIV256 | | Prescaled RTC clock divided by 256 | |
| 0x0A | | PIT_DIV128 | | Prescaled RTC clock divided by 128 | |
| 0x0B | | PIT_DIV64 | | Prescaled RTC clock divided by 64 | |
| 0x10 | CCL | LUT0 | Async | LUT output level | All channels |
| 0x11 | | LUT1 | | | |
| 0x12 | | LUT2 | | | |
| 0x13 | | LUT3 | | | |
| 0x14 | | LUT4(1) | | | |
| 0x15 | | LUT5(1) | | | |
| 0x20 | AC0 | OUT | Async | Comparator output level | All channels |
| 0x21 | AC1 | | | | |
| 0x22 | AC2 | | | | |
| 0x24 | ADC0 | RESRDY | Sync | Result ready | All channels |
| 0x30 | ZCD0 | OUT | Async | ZCD output level | All channels |
| 0x31 | ZCD1(1) | | | | |
| 0x32 | ZCD2(1) | | | | |
| 0x34 | OPAMP0 | READY | Sync | OPAMP Ready | All Channels |
| 0x35 | OPAMP1 | | | | |
| 0x36 | OPAMP2 | | | | |
| 0x40-0x47 | PORTA | PIN0-PIN7 | Async | Pin level(2) | CHANNEL0 and CHANNEL1 only |
| 0x48-0x4F | PORTB(1) | | | | |
| 0x40-0x47 | PORTC | PIN0-PIN7 | Async | PIN level(2) | CHANNEL2 and CHANNEL3 only |
| 0x48-0x4F | PORTD | | | | |
| 0x40-0x47 | PORTE (1) | PIN0-PIN7 | Async | Pin level (2) | CHANNEL4 and CHANNEL5 only |
| 0x48-0x4F | PORTF | | | | |
| 0x40-0x47 | PORTG(1) | PIN0-PIN7 | Async | Pin level (2) | CHANNEL6 and CHANNEL7 only |

.....continued

| Generator | | | Async/Sync | Description | Channel Availability |
|---|---|---|---|---|---|
| Value | Name | | | | |
| | Peripheral | Output | | | |
| 0x60 | USART0 | XCK | Sync | Clock signal in SPI Master mode and synchronous USART Master mode | All channels |
| 0x61 | USART1 | | | | |
| 0x62 | USART2 | | | | |
| 0x63 | USART3(1) | | | | |
| 0x64 | USART4(1) | | | | |
| 0x65 | USART5(1) | | | | |
| 0x68 | SPI0 | SCK | Sync | SPI master clock signal | All channels |
| 0x69 | SPI1 | | | | |
| 0x80 | TCA0 | OVF_LUNF | Sync | Overflow/Low byte timer underflow | All channels |
| 0x81 | | HUNF | | High byte timer underflow | |
| 0x84 | | CMP0_LCMP0 | | Compare channel 0 match/Low byte timer compare channel 0 match | |
| 0x85 | | CMP1_LCMP1 | | Compare channel 1 match/Low byte timer compare channel 1 match | |
| 0x86 | | CMP2_LCMP2 | | Compare channel 2 match/Low byte timer compare channel 2 match | |
| 0x88 | TCA1(1) | OVF_LUNF | Sync | Overflow/Low byte timer underflow | All channels |
| 0x89 | | HUNF | | High byte timer underflow | |
| 0x8C | | CMP0_LCMP0 | | Compare channel 0 match/Low byte timer compare channel 0 match | |
| 0x8D | | CMP1_LCMP1 | | Compare channel 1 match/Low byte timer compare channel 1 match | |
| 0x8E | | CMP2_LCMP2 | | Compare channel 2 match/Low byte timer compare channel 2 match | |
| 0xA0 | TCB0 | CAPT | Sync | CAPT Interrupt flag set(3) | All channels |
| 0xA1 | | OVF | | Counter overflow | |
| 0xA2 | TCB1 | CAPT | Sync | CAPT Interrupt flag set(3) | All channels |
| 0xA3 | | OVF | | Counter overflow | |
| 0xA4 | TCB2 | CAPT | Sync | CAPT interrupt flag set(3) | All channels |
| 0xA5 | | OVF | | Counter overflow | |
| 0xA6 | TCB(1) | CAPT | Sync | CAPT interrupt flag set(3) | All channels |
| 0xA7 | | OVF | | Counter overflow | |
| 0xA8 | TCB4 (1) | CAPT | Sync | CAPT interrupt flag set (3) | All channels |
| 0xA9 | | OVF | | Counter overflow | |
| 0xB0 | TCD0 | CMPBCLR | Async | Counter matches CMPBCLR | All channels |
| 0xB1 | | CMPASET | | Counter matches CMPASET | |
| 0xB2 | | CMPBSET | | Counter matches CMPBSET | |
| 0xB3 | | PROGEV | | Programmable event output | |

**Notes:**

1. Not all peripheral instances are available for all pin-counts. Refer to the *Peripherals and Architecture* section for details.
2. Event from PORT pin will be zero if the input driver is disabled.
3. The operational mode of the timer decides when the CAPT flag is raised. See the *16-bit Timer/Counter Type B (TCB)* section for details.

### 16.5.3 User Channel MUX

**Name:** USER
**Offset:** 0x20 + n*0x01 [n=0..53]
**Reset:** 0x00
**Property:** -

Each event user can be connected to one channel and several users can be connected to the same channel. The following table lists all Event System users with their corresponding user ID number and name. The user name is given by combining USER with Peripheral and Input from the table below in the following way: USERPERIPHERALINPUT.

| USER # | User Name | | Async/ Sync | Description |
|---|---|---|---|---|
| | Peripheral | Input | | |
| 0x00 | CCL | LUT0A | Async | CCL LUT0 event input A |
| 0x01 | | LUT0B | | CCL LUT0 event input B |
| 0x02 | | LUT1A | | CCL LUT1 event input A |
| 0x03 | | LUT1B | | CCL LUT1 event input B |
| 0x04 | | LUT2A | | CCL LUT2 event input A |
| 0x05 | | LUT2B | | CCL LUT2 event input B |
| 0x06 | | LUT3A | | CCL LUT3 event input A |
| 0x07 | | LUT3B | | CCL LUT3 event input B |
| 0x08 | | LUT4A(1) | | CCL LUT4 event input A |
| 0x09 | | LUT4B(1) | | CCL LUT4 event input B |
| 0x0A | | LUT5A(1) | | CCL LUT5 event input A |
| 0x0B | | LUT5B (1) | | CCL LUT5 event input B |
| 0x0C | ADC0 | START | Async | ADC start on event |
| 0x0D | EVSYS | EVOUTA | Async | Event output A |
| 0x0E | | EVOUTB(1) | | Event output B |
| 0x0F | | EVOUTC | | Event output C |
| 0x10 | | EVOUTD | | Event output D |
| 0x11 | | EVOUTE (1) | | Event output E |
| 0x12 | | EVOUTF (1) | | Event output F |
| 0x13 | | EVOUTG (1) | | Event output G |
| 0x14 | USART0 | IRDA | Sync | USART0 IrDA event input |
| 0x15 | USART1 | IRDA | | USART1 IrDA event input |
| 0x16 | USART2 | IRDA | | USART2 IrDA event input |
| 0x17 | USART3(1) | IRDA | | USART3 IrDA event input |
| 0x18 | USART4(1) | IRDA | | USART4 IrDA event input |
| 0x19 | USART5 | IRDA | | USART5 IrDA event input |

.....continued

| USER # | User Name | | Async/ Sync | Description |
|---|---|---|---|---|
| | Peripheral | Input | | |
| 0x1A | TCA0 | CNTA | Sync | Count on event or control count direction |
| 0x1B | | CNTB | | Restart on event or control count direction |
| 0x1C | TCA1(1) | CNTA | Sync | Count on event or control count direction |
| 0x1D | | CNTB | | Restart on event or control count direction |
| 0x1E | TCB0 | CAPT | Both(2) | Start, stop, capture, restart or clear counter |
| 0x1F | | COUNT | Sync | Count on event |
| 0x20 | TCB1 | CAPT | Both(2) | Start, stop, capture, restart or clear counter |
| 0x21 | | COUNT | Sync | Count on event |
| 0x22 | TCB2 | CAPT | Both(2) | Start, stop, capture, restart or clear counter |
| 0x23 | | COUNT | Sync | Count on event |
| 0x24 | TCB3(1) | CAPT | Both(2) | Start, stop, capture, restart or clear counter |
| 0x25 | | COUNT | Sync | Count on event |
| 0x26 | TCB4(1) | CAPT | Both(2) | Start, stop, capture, restart or clear counter |
| 0x27 | | COUNT | Sync | Count on event |
| 0x28 | TCD0 | INPUTA | Async | Fault or capture |
| 0x29 | | INPUTB | | Fault or capture |
| 0x2A | OPAMP0 | ENABLE | Async | Enable OPAMP |
| 0x2B | OPAMP0 | DISABLE | Sync | Disable OPAMP |
| 0x2C | OPAMP0 | DUMP | Async | OPAMP $V_{OUT}$ dump for Integrator mode |
| 0x2D | OPAMP0 | DRIVE | Async | OPAMP drive in Normal mode |
| 0x2E | OPAMP1 | ENABLE | Async | Enable OPAMP |
| 0x2F | OPAMP1 | DISABLE | Sync | Disable OPAMP |
| 0x30 | OPAMP1 | DUMP | Async | OPAMP $V_{OUT}$ dump for Integrator mode |
| 0x31 | OPAMP1 | DRIVE | Async | OPAMP drive in Normal mode |
| 0x32 | OPAMP2 | ENABLE | Async | Enable OPAMP |
| 0x33 | OPAMP2 | DISABLE | Sync | Disable OPAMP |
| 0x34 | OPAMP2 | DUMP | Async | OPAMP $V_{OUT}$ dump for Integrator mode |
| 0x35 | OPAMP2 | DRIVE | Async | OPAMP drive in Normal mode |

**Notes:**

1. Not all peripheral instances are available for all pin-counts. Refer to the *Peripherals and Architecture* section for details.
2. Depends on timer operational mode.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | USER[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 – USER[7:0]** User Channel Selection
Configures which Event System channel the user is connected to.

| Value | Description |
|---|---|
| 0 | OFF, no channel is connected to this Event System user |
| n | The event user is connected to CHANNEL(n-1) |

# 17. PORTMUX - Port Multiplexer

## 17.1 Overview

The Port Multiplexer (PORTMUX) can either enable or disable the functionality of the pins, or change between default and alternative pin positions. Available options are described in detail in the PORTMUX register map and depend on the actual pin and its properties.

For available pins and functionality, refer to the *I/O Multiplexing and Considerations* section.

## 17.2 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | EVSYSROUTEA | 7:0 | | EVOUTG | EVOUTF | EVOUTE | EVOUTD | EVOUTC | EVOUTB | EVOUTA |
| 0x01 | CCLROUTEA | 7:0 | | | LUT5 | LUT4 | LUT3 | LUT2 | LUT1 | LUT0 |
| 0x02 | USARTROUTEA | 7:0 | USART3[1:0] | | USART2[1:0] | | USART1[1:0] | | USART0[1:0] | |
| 0x03 | USARTROUTEB | 7:0 | | | | | USART5[1:0] | | USART4[1:0] | |
| 0x04 | SPIROUTEA | 7:0 | | | | | SPI1[1:0] | | SPI0[1:0] | |
| 0x05 | TWIROUTEA | 7:0 | | | | | TWI1[1:0] | | TWI0[1:0] | |
| 0x06 | TCAROUTEA | 7:0 | | | TCA1[2:0] | | | TCA0[2:0] | | |
| 0x07 | TCBROUTEA | 7:0 | | | | TCB4 | TCB3 | TCB2 | TCB1 | TCB0 |
| 0x08 | TCDROUTEA | 7:0 | | | | | | TCD0[2:0] | | |
| 0x09 | ACROUTEA | 7:0 | | | | | | AC2 | AC1 | AC0 |
| 0x0A | ZCDROUTEA | 7:0 | | | | | | ZCD2 | ZCD1 | ZCD0 |

## 17.3 Register Description

### 17.3.1 EVSYS Pin Position

**Name:** EVSYSROUTEA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | EVOUTG | EVOUTF | EVOUTE | EVOUTD | EVOUTC | EVOUTB | EVOUTA |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 6 – EVOUTG** Event Output G
This bit controls pin position for event output G.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | EVOUT on PG2 |
| 1 | ALT1 | EVOUT on PG7 |

**Bit 5 – EVOUTF** Event Output F
This bit controls pin position for event output F.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | EVOUT on PF2 |
| 1 | - | Reserved |

**Bit 4 – EVOUTE** Event Output E
This bit controls pin position for event output E.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | EVOUT on PE2 |
| 1 | ALT1 | EVOUT on PE7 |

**Bit 3 – EVOUTD** Event Output D
This bit controls pin position for event output D.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | EVOUT on PD2 |
| 1 | ALT1 | EVOUT on PD7 |

**Bit 2 – EVOUTC** Event Output C
This bit controls pin position for event output C.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | EVOUT on PC2 |
| 1 | ALT1 | EVOUT on PC7 |

**Bit 1 – EVOUTB** Event Output B
This bit controls pin position for event output B.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | EVOUT on PB2 |
| 1 | ALT1 | EVOUT on PB7 |

**Bit 0 – EVOUTA** Event Output A
This bit controls pin position for event output A.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | EVOUT on PA2 |
| 1 | ALT1 | EVOUT on PA7 |

### 17.3.2 CCL LUTn Pin Position

**Name:** CCLROUTEA
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | LUT5 | LUT4 | LUT3 | LUT2 | LUT1 | LUT0 |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 5 – LUT5** CCL LUT 5 Signals
This bit field controls the pin positions for CCL LUT 5 signals

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | OUT | IN0 | IN1 | IN2 |
| 0 | DEFAULT | PG3 | PG0 | PG1 | PG2 |
| 1 | ALT1 | PG6 | PG0 | PG1 | PG2 |

**Bit 4 – LUT4** CCL LUT 4 Signals
This bit field controls the pin positions for CCL LUT 4 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | OUT | IN0 | IN1 | IN2 |
| 0 | DEFAULT | PB3 | PB0 | PB1 | PB2 |
| 1 | ALT1 | PB6 | PB0 | PB1 | PB2 |

**Bit 3 – LUT3** CCL LUT 3 Signals
This bit field controls the pin positions for CCL LUT 3 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | OUT | IN0 | IN1 | IN2 |
| 0 | DEFAULT | PF3 | PF0 | PF1 | PF2 |
| 1 | - | Reserved | Reserved | Reserved | Reserved |

**Bit 2 – LUT2** CCL LUT 2 Signals
This bit field controls the pin positions for CCL LUT 2 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | OUT | IN0 | IN1 | IN2 |
| 0 | DEFAULT | PD3 | PD0 | PD1 | PD2 |
| 1 | ALT1 | PD6 | PD0 | PD1 | PD2 |

**Bit 1 – LUT1** CCL LUT 1 Signals
This bit field controls the pin positions for CCL LUT 1 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | OUT | IN0 | IN1 | IN2 |
| 0 | DEFAULT | PC3 | PC0 | PC1 | PC2 |
| 1 | ALT1 | PC6 | PC0 | PC1 | PC2 |

**Bit 0 – LUT0** CCL LUT 0 Signals
This bit field controls the pin positions for CCL LUT 0 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | OUT | IN0 | IN1 | IN2 |
| 0 | DEFAULT | PA3 | PA0 | PA1 | PA2 |
| 1 | ALT1 | PA6 | PA0 | PA1 | PA2 |

### 17.3.3 USARTn Pin Position

**Name:** USARTROUTEA
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | USART3[1:0] | | USART2[1:0] | | USART1[1:0] | | USART0[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 7:6 – USART3[1:0]** USART 3 Signals
This bit field controls the pin positions for USART 3 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | TxD | RxD | XCK | XDIR |
| 0x0 | DEFAULT | PB0 | PB1 | PB2 | PB3 |
| 0x1 | ALT1 | PB4 | PB5 | PB6 | PB7 |
| 0x2 | - | Reserved | | | |
| 0x3 | NONE | Not connected to any pins | | | |

**Bits 5:4 – USART2[1:0]** USART 2 Signals
This bit field controls the pin positions for USART 2 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | TxD | RxD | XCK | XDIR |
| 0x0 | DEFAULT | PF0 | PF1 | PF2 | PF3 |
| 0x1 | ALT1 | PF4 | PF5 | Not available | Not available |
| 0x2 | - | Reserved | | | |
| 0x3 | NONE | Not connected to any pins | | | |

**Bits 3:2 – USART1[1:0]** USART 1 Signals
This bit field controls the pin positions for USART 1 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | TxD | RxD | XCK | XDIR |
| 0x0 | DEFAULT | PC0 | PC1 | PC2 | PC3 |
| 0x1 | ALT1 | PC4 | PC5 | PC6 | PC7 |
| 0x2 | - | Reserved | | | |
| 0x3 | NONE | Not connected to any pins | | | |

**Bits 1:0 – USART0[1:0]** USART 0 Signals
This bit field controls the pin positions for USART 0 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | TxD | RxD | XCK | XDIR |
| 0x0 | DEFAULT | PA0 | PA1 | PA2 | PA3 |
| 0x1 | ALT1 | PA4 | PA5 | PA6 | PA7 |
| 0x2 | - | Reserved | | | |
| 0x3 | NONE | Not connected to any pins | | | |

### 17.3.4 USARTn Pin Position

**Name:** USARTROUTEB
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | USART5[1:0] | | USART4[1:0] | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:2 – USART5[1:0]** USART 5 Signals
This bit field controls the pin positions for USART 5 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | TxD | RxD | XCK | XDIR |
| 0x0 | DEFAULT | PG0 | PG1 | PG2 | PG3 |
| 0x1 | ALT1 | PG4 | PG5 | PG6 | PG7 |
| 0x2 | - | Reserved | | | |
| 0x3 | NONE | Not connected to any pins | | | |

**Bits 1:0 – USART4[1:0]** USART 4 Signals
This bit field controls the pin positions for USART 4 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | TxD | RxD | XCK | XDIR |
| 0x0 | DEFAULT | PE0 | PE1 | PE2 | PE3 |
| 0x1 | ALT1 | PE4 | PE5 | PE6 | PE7 |
| 0x2 | - | Reserved | | | |
| 0x3 | NONE | Not connected to any pins | | | |

### 17.3.5 SPIn Pin Position

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SPI1[1:0] | | SPI0[1:0] | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:2 – SPI1[1:0]** SPI 1 Signals
This bit field controls the pin positions for SPI 1 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | MOSI | MISO | SCK | $\overline{SS}$ |
| 0x0 | DEFAULT | PC0 | PC1 | PC2 | PC3 |
| 0x1 | ALT1 | PC4 | PC5 | PC6 | PC7 |
| 0x2 | ALT2 | PB4 | PB5 | PB6 | PB7 |
| 0x3 | NONE | Not connected to any pins | | | |

**Bits 1:0 – SPI0[1:0]** SPI 0 Signals
This bit field controls the pin positions for SPI 0 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | MOSI | MISO | SCK | $\overline{SS}$ |
| 0x0 | DEFAULT | PA4 | PA5 | PA6 | PA7 |
| 0x1 | ALT1 | PE0 | PE1 | PE2 | PE3 |
| 0x2 | ALT2 | PG4 | PG5 | PG6 | PG7 |
| 0x3 | NONE | Not connected to any pins | | | |

### 17.3.6 TWIn Pin Position

**Name:** TWIROUTEA
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | TWI1[1:0] | | TWI0[1:0] | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:2 – TWI1[1:0]** TWI 1 Signals
This bit field controls the pin positions for TWI 1 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | Master/Slave | | Dual mode (Slave) | |
| | | SDA | SCL | SDA | SCL |
| 0x0 | DEFAULT | PF2 | PF3 | PB2 | PB3 |
| 0x1 | ALT1 | PF2 | PF3 | PB6(1) | PB7(1) |
| 0x2 | ALT2 | PB2 | PB3 | PB6(1) | PB7(1) |
| 0x3 | - | Reserved | | | |

**Note:**
1. Normal and Fast Mode only. Standard General Purpose I/O driver specification apply.

**Bits 1:0 – TWI0[1:0]** TWI 0 Signals
This bit field controls the pin positions for TWI 0 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | Master/Slave | | Dual mode (Slave) | |
| | | SDA | SCL | SDA | SCL |
| 0x0 | DEFAULT | PA2 | PA3 | PC2 | PC3 |
| 0x1 | ALT1 | PA2 | PA3 | PC6(1) | PC7(1) |
| 0x2 | ALT2 | PC2 | PC3 | PC6(1) | PC7(1) |
| 0x3 | - | Reserved | | | |

**Note:**
1. Normal and Fast Mode only. Standard General Purpose I/O driver specification apply.

### 17.3.7 TCAn Pin Position

**Name:** TCAROUTEA
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | TCA1[2:0] | | | TCA0[2:0] | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:3 – TCA1[2:0]** TCA1 Signals
This bit field controls the pin positions for TCA1 signals.

| Value | Name | Description | | | | | |
|---|---|---|---|---|---|---|---|
| | | WO0 | WO1 | WO2 | WO3 | WO4 | WO5 |
| 0x0 | PORTB | PB0 | PB1 | PB2 | PB3 | PB4 | PB5 |
| 0x1 | PORTC | PC4 | PC5 | PC6 | - | - | - |
| 0x2 | PORTE | PE4 | PE5 | PE6 | - | - | - |
| 0x3 | PORTG | PG0 | PG1 | PG2 | PG3 | PG4 | PG5 |
| Other | - | Reserved | | | | | |

**Bits 2:0 – TCA0[2:0]** TCA0 Signals
This bit field controls the pin positions for TCA0 signals.

| Value | Name | Description | | | | | |
|---|---|---|---|---|---|---|---|
| | | WO0 | WO1 | WO2 | WO3 | WO4 | WO5 |
| 0x0 | PORTA | PA0 | PA1 | PA2 | PA3 | PA4 | PA5 |
| 0x1 | PORTB | PB0 | PB1 | PB2 | PB3 | PB4 | PB5 |
| 0x2 | PORTC | PC0 | PC1 | PC2 | PC3 | PC4 | PC5 |
| 0x3 | PORTD | PD0 | PD1 | PD2 | PD3 | PD4 | PD5 |
| 0x4 | PORTE | PE0 | PE1 | PE2 | PE3 | PE4 | PE5 |
| 0x5 | PORTF | PF0 | PF1 | PF2 | PF3 | PF4 | PF5 |
| 0x6 | PORTG | PG0 | PG1 | PG2 | PG3 | PG4 | PG5 |
| 0x7 | - | Reserved | | | | | |

### 17.3.8 TCBn Pin Position

**Name:** TCBROUTEA
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | TCB4 | TCB3 | TCB2 | TCB1 | TCB0 |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bit 4 – TCB4** TCB4 Output
This bit controls pin position for TCB4 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | WO on PG3 |
| 1 | ALT1 | WO on PC6 |

**Bit 3 – TCB3** TCB3 Output
This bit controls pin position for TCB3 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | WO on PB5 |
| 1 | ALT1 | WO on PC1 |

**Bit 2 – TCB2** TCB2 Output
This bit controls pin position for TCB2 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | WO on PC0 |
| 1 | ALT1 | WO on PB4 |

**Bit 1 – TCB1** TCB1 Output
This bit controls pin position for TCB1 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | WO on PA3 |
| 1 | ALT1 | WO on PF5 |

**Bit 0 – TCB0** TCB0 Output
This bit controls pin position for TCB0 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | WO on PA2 |
| 1 | ALT1 | WO on PF4 |

### 17.3.9 TCDn Pin Position

**Name:** TCDROUTEA
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | TCD0[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – TCD0[2:0]** TCD0 Signals
This bit field controls the pin positions for TCD0 signals.

| Value | Name | Description | | | |
|---|---|---|---|---|---|
| | | WOA | WOB | WOC | WOD |
| 0x0 | DEFAULT | PA4 | PA5 | PA6 | PA7 |
| 0x1 | ALT1 | PB4 | PB5 | PB6 | PB7 |
| 0x2 | ALT2 | PF0 | PF1 | PF2 | PF3 |
| 0x3 | ALT3 | PG4 | PG5 | PG6 | PG7 |
| Other | - | Reserved | | | |

### 17.3.10 ACn Pin Position

**Name:** ACROUTEA
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | AC2 | AC1 | AC0 |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bit 2 – AC2** Analog Comparator 2 Output
This bit controls pin position for AC2 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | OUT on PA7 |
| 1 | ALT1 | OUT on PC6 |

**Bit 1 – AC1** Analog comparator 1 Output
This bit controls pin position for AC1 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | OUT on PA7 |
| 1 | ALT1 | OUT on PC6 |

**Bit 0 – AC0** Analog Comparator 0 Output
This bit controls pin position for AC0 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | OUT on PA7 |
| 1 | ALT1 | OUT on PC6 |

### 17.3.11 ZCDn Pin Position

**Name:** ZCDROUTEA
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | ZCD2 | ZCD1 | ZCD0 |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bit 2 – ZCD2** Zero Cross Detector 2 Output
This bit controls pin position for ZCD2 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | OUT on PA7 |
| 1 | ALT1 | OUT on PC7 |

**Bit 1 – ZCD1** Zero Cross Detector 1 Output
This bit controls pin position for ZCD1 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | OUT on PA7 |
| 1 | ALT1 | OUT on PC7 |

**Bit 0 – ZCD0** Zero Cross Detector 0 Output
This bit controls pin position for ZCD0 output.

| Value | Name | Description |
|---|---|---|
| 0 | DEFAULT | OUT on PA7 |
| 1 | ALT1 | OUT on PC7 |

# 18. PORT - I/O Pin Configuration

## 18.1 Features

- General Purpose Input and Output Pins with Individual Configuration:
  - Pull-up
  - Inverted I/O
  - Input voltage threshold
- Interrupts and Events:
  - Sense both edges
  - Sense rising edges
  - Sense falling edges
  - Sense low level
- Optional Slew Rate Control per I/O Port
- Asynchronous Pin Change Sensing that Can Wake the Device From all Sleep Modes
- Efficient and Safe Access to Port Pins
  - Hardware Read-Modify-Write (RMW) through dedicated toggle/clear/set registers
  - Mapping of often-used PORT registers into bit-accessible I/O memory space (virtual ports)

## 18.2 Overview

The I/O pins of the device are controlled by instances of the PORT peripheral registers. Each PORT instance has up to eight I/O pins. The PORTs are named PORTA, PORTB, PORTC, etc. Refer to the *I/O Multiplexing and Considerations* section to see which pins are controlled by what instance of PORT. The base addresses of the PORT instances and the corresponding Virtual PORT instances are listed in the *Peripherals and Architecture* section.

Each PORT pin has a corresponding bit in the Data Direction (PORTx.DIR) and Data Output Value (PORTx.OUT) registers to enable that pin as an output and to define the output state. For example, pin PA3 is controlled by DIR[3] and OUT[3] of the PORTA instance.

The input value of a PORT pin is synchronized to the Peripheral Clock (CLK_PER) and then made accessible as the data input value (PORTx.IN). The value of the pin can be read whether the pin is configured as input or output.

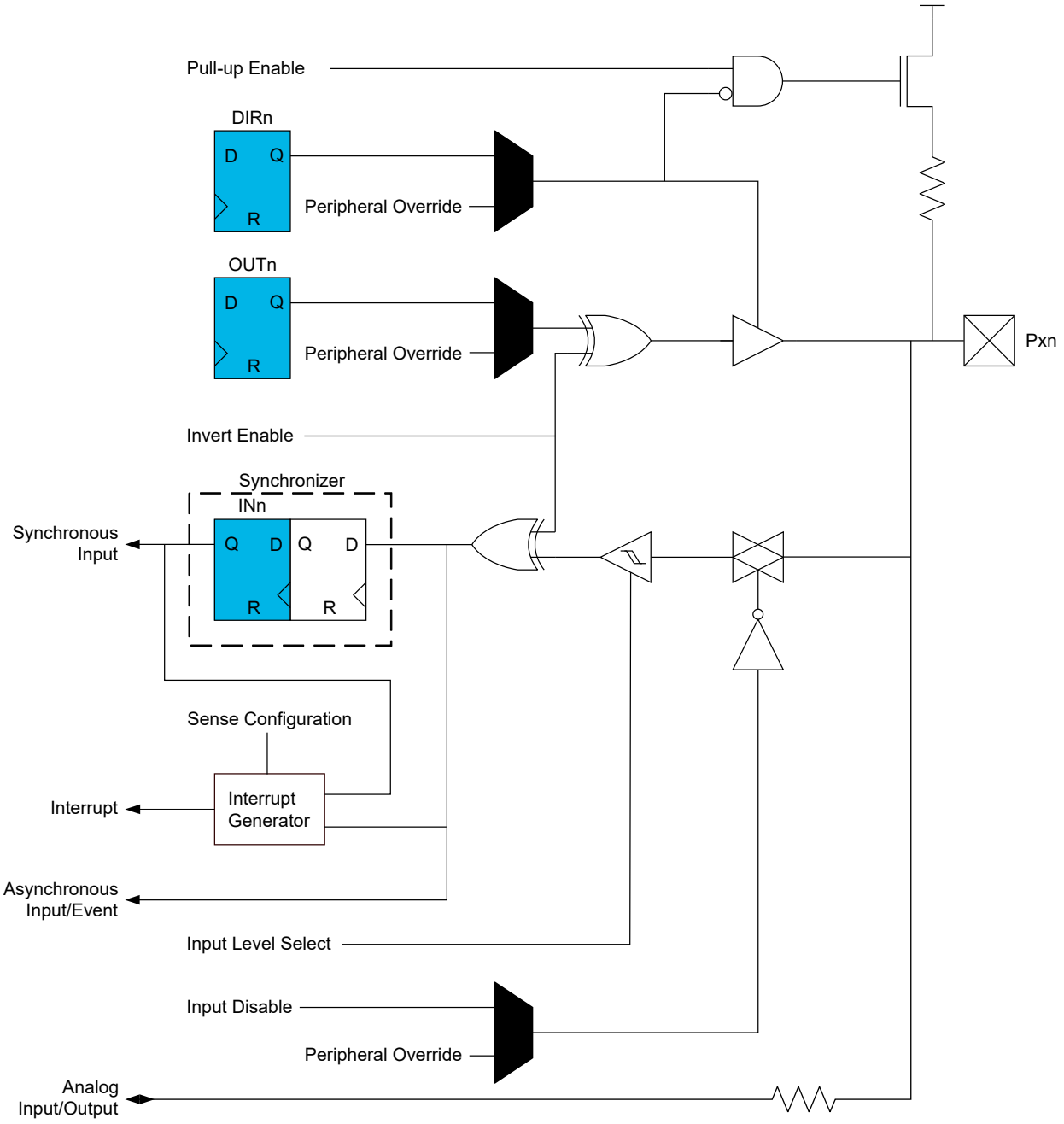
The PORT also supports asynchronous input sensing with interrupts and events for selectable pin change conditions. Asynchronous pin change sensing means that a pin change can trigger an interrupt and wake the device from sleep, including sleep modes where CLK_PER is stopped.

All pin functions are individually configurable per pin. The pins have hardware Read-Modify-Write functionality for a safe and correct change of the drive values and/or input and sense configuration.

The PORT pin configuration controls input and output selection of other device functions.

### 18.2.1 Block Diagram

**Figure 18-1. PORT Block Diagram**




### 18.2.2 Signal Description

| Signal | Type | Description |
|---|---|---|
| Pxn | I/O pin | I/O pin n on PORTx |

## 18.3 Functional Description

### 18.3.1 Initialization

After Reset, all outputs are tri-stated, and digital input buffers enabled even if there is no clock running.

The following steps are all optional when initializing PORT operation:

- Enable or disable the output driver for pin Pxn by respectively writing '1' to bit n in the PORTx.DIRSET or PORTx.DIRCLR register
- Set the output driver for pin Pxn to high or low level respectively by writing '1' to bit n in the PORTx.OUTSET or PORTx.OUTCLR register
- Read the input of pin Pxn by reading bit n in the PORTx.IN register
- Configure the individual pin configurations and interrupt control for pin Pxn in PORTx.PINnCTRL

**Important:** For lowest power consumption, disable the digital input buffer of unused pins and pins that are used as analog inputs or outputs.

Specific pins, such as those used to connect a debugger, may be configured differently, as required by their special function.

### 18.3.2 Operation

#### 18.3.2.1 Basic Functions

Each pin group x has its own set of PORT registers. I/O pin Pxn can be controlled by the registers in PORTx.

To use pin number n as an output, write bit n of the PORTx.DIR register to '1'. This can be done by writing bit n in the PORTx.DIRSET register to '1', which will avoid disturbing the configuration of other pins in that group. The $n^{th}$ bit in the PORTx.OUT register must be written to the desired output value.

Similarly, writing a PORTx.OUTSET bit to '1' will set the corresponding bit in the PORTx.OUT register to '1'. Writing a bit in PORTx.OUTCLR to '1' will clear that bit in PORTx.OUT to '0'. Writing a bit in PORTx.OUTTGL or PORTx.IN to '1' will toggle that bit in PORTx.OUT.

To use pin n as an input, bit n in the PORTx.DIR register must be written to '0' to disable the output driver. This can be done by writing bit n in the PORTx.DIRCLR register to '1', which will avoid disturbing the configuration of other pins in that group. The input value can be read from bit n in the PORTx.IN register as long as the ISC bit is not set to INPUT_DISABLE.

Writing a bit to '1' in PORTx.DIRTGL will toggle that bit in PORTx.DIR and toggle the direction of the corresponding pin.

#### 18.3.2.2 Port Configuration

The Port Control (PORTx.PORTCTRL) register is used to configure the slew rate limitation for all the PORTx pins.

The slew rate limitation is enabled by writing a '1' to the Slew Rate Limit Enable (SLR) bit in PORTx.PORTCTRL. Refer to the *Electrical Characteristics* section for further details.

#### 18.3.2.3 Pin Configuration

The Pin n Control (PORTx.PINnCTRL) register is used to configure inverted I/O, pull-up, and input sensing of a pin. The control register for pin n is at the byte address $\mathrm{PORTx} + 0\mathrm{x}10 + n$.

All input and output on the respective pin n can be inverted by writing a '1' to the Inverted I/O Enable (INVEN) bit in PORTx.PINnCTRL. When INVEN is '1', the PORTx.IN/OUT/OUTSET/OUTTGL registers will have inverted operation for this pin.

Toggling the INVEN bit causes an edge on the pin, which can be detected by all peripherals using this pin, and is seen by interrupts or events if enabled.

The Input Level Select (INLVL) bit controls the input voltage threshold for pin n in PORTx.PINnCTRL. A selection of Schmitt trigger thresholds derived from the supply voltage or TTL levels are available.

The input threshold is important in determining the value of bit n in the PORTx.IN register and also the level at which an interrupt condition occurs if that feature is enabled.

The input pull-up of pin n is enabled by writing a '1' to the Pull-up Enable (PULLUPEN) bit in PORTx.PINnCTRL. The pull-up is disconnected when the pin is configured as an output, even if PULLUPEN is '1'.

Pin interrupts can be enabled for pin n by writing to the Input/Sense Configuration (ISC) bit field in PORTx.PINnCTRL. Refer to 18.3.3 Interrupts for further details.

The digital input buffer for pin n can be disabled by writing the INPUT_DISABLE setting to ISC. This can reduce power consumption and may reduce noise if the pin is used as analog input. While configured to INPUT_DISABLE, bit n in PORTx.IN will not change since the input synchronizer is disabled.

#### 18.3.2.4 Multi-Pin Configuration

The multi-pin configuration function is used to configure multiple port pins in one operation. The wanted pin configuration is first written to the PORTx.PINCONFIG register, followed by a register write with the selected pins to modify. This allows changing the configuration (PORTx.PINnCTRL) for up to eight pins in one write.

**Tip:** The PORTx.PINCONFIG register is mirrored on all ports, which allows the use of a single setting across multiple ports. The PORTx.PINCTRLUPD/SET/CLR registers are not mirrored and need to be applied to each port.

For the multi-pin configuration, port pins can be configured and modified by writing to the following registers.

**Table 18-1. Multi-Pin Configuration Registers**

| Register | Description |
|---|---|
| PORTx.PINCONFIG | PINnCTRL (ISC, PULLUPEN, INLVL and INVEN) setting to load simultaneously to multiple PINnCTRL registers |
| PORTx.PINCTRLUPD | Writing a '1' to bit n in the PINCTRLUPD register will copy the PINCONFIG register content to the PINnCTRL register |
| PORTx.PINCTRLSET(1) | Writing a '1' to bit n in the PINCTRLSET register will set the individual bits in the PINnCTRL register, according to the bits set to '1' in the PINCONFIG register |
| PORTx.PINCTRLCLR(2) | Writing a '1' to bit n in the PINCTRLCLRn register will clear the individual bits in the PINnCTRL register, according to the bits set to '1' in the PINCONFIG register |

**Notes:**

1. Using PINCTRLSET to configure ISC bit fields that are non-zero will result in a bitwise OR with the PINCONFIG and PINnCTRL registers. This may give an unexpected setting.
2. Using PINCTRLCLR to configure ISC bit fields that are non-zero will result in a bitwise inverse AND with the PINCONFIG and PINnCTRL registers. This may give an unexpected setting.

#### 18.3.2.5 Virtual Ports

The Virtual PORT registers map the most frequently used regular PORT registers into the I/O Register space with single-cycle bit access. Access to the Virtual PORT registers has the same outcome as access to the regular registers but allows for memory specific instructions, such as bit manipulation instructions, which cannot be used in the extended I/O Register space where the regular PORT registers reside. The following table shows the mapping between the PORT and VPORT registers.

**Table 18-2. Virtual Port Mapping**

| Regular PORT Register | Mapped to Virtual PORT Register |
|---|---|
| PORTx.DIR | VPORTx.DIR |

| .....continued | |
|---|---|
| **Regular PORT Register** | **Mapped to Virtual PORT Register** |
| PORTx.OUT | VPORTx.OUT |
| PORTx.IN | VPORTx.IN |
| PORTx.INTFLAGS | VPORTx.INTFLAGS |

#### 18.3.2.6 Peripheral Override

Peripherals such as USARTs, ADCs and timers may be connected to I/O pins. Such peripherals will usually have a primary and, optionally, one or more alternate I/O pin connections, selectable by PORTMUX or a multiplexer inside the peripheral. By configuring and enabling such peripherals, the general purpose I/O pin behavior normally controlled by PORT will be overridden in a peripheral dependent way. Some peripherals may not override all the PORT registers, leaving the PORT module to control some aspects of the I/O pin operation.

Refer to the description of each peripheral for information on the peripheral override. Any pin in a PORT that is not overridden by a peripheral will continue to operate as a general purpose I/O pin.

#### 18.3.2.7 Multi-Voltage I/O

One or more PORT pin groups are connected to the VDDIO2 power domain, allowing a different I/O supply voltage on these pins. Refer to the *Multi-Voltage I/O (MVIO)* section for further information.

### 18.3.3 Interrupts

**Table 18-3. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| PORTx | PORT interrupt | INTn in PORTx.INTFLAGS is raised as configured by the Input/Sense Configuration (ISC) bit in PORTx.PINnCTRL |

Each PORT pin n can be configured as an interrupt source. Each interrupt can be individually enabled or disabled by writing to ISC in PORTx.PINnCTRL.

When an interrupt condition occurs, the corresponding interrupt flag is set in the Interrupt Flags register of the peripheral (*peripheral*.INTFLAGS).

An interrupt request is generated when the corresponding interrupt source is enabled and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

When setting or changing interrupt settings, take these points into account:

- If an Inverted I/O Enable (INVEN) bit is toggled in the same cycle as ISC is changed, the edge caused by the inversion toggling may not cause an interrupt request
- Changing INLVL for a pin must be performed while relevant interrupts and peripheral modules are disabled. Changing the threshold while a module is active may generate a temporary state transition on the input, regardless of the actual voltage level on that pin.
- If an input is disabled by writing to ISC while synchronizing an interrupt, that interrupt may be requested on re-enabling the input, even if it is re-enabled with a different interrupt setting
- If the interrupt setting is changed by writing to ISC while synchronizing an interrupt, that interrupt may not be requested

#### 18.3.3.1 Asynchronous Sensing Pin Properties

All PORT pins support asynchronous input sensing with interrupts for selectable pin change conditions. Fully asynchronous pin change sensing can trigger an interrupt and wake the device from all sleep modes, including modes where the Peripheral Clock (CLK_PER) is stopped, while partially asynchronous pin change sensing is limited as per the table below. See the *I/O Multiplexing and Considerations* section for further details on which pins support fully asynchronous pin change sensing.

**Table 18-4. Behavior Comparison of Sense Pins**

| Property | Partially Asynchronous Pins | Fully Asynchronous Pins |
|---|---|---|
| Waking the device from sleep modes with CLK_PER running | From all interrupt sense configurations | From all interrupt sense configurations |
| Waking the device from sleep modes with CLK_PER stopped | Only from BOTHEDGES or LEVEL interrupt sense configurations | |
| Minimum pulse-width to trigger an interrupt with CLK_PER running | Minimum one CLK_PER cycle | Less than one CLK_PER cycle |
| Minimum pulse-width to trigger an interrupt with CLK_PER stopped | The pin value must be kept until CLK_PER has restarted(1) | |
| Interrupt “dead-time” | No new interrupt for three CLK_PER cycles after the previous | |

**Note:**

1. If a partially asynchronous input pin is used for wake-up from sleep with CLK_PER stopped, the required level must be held long enough for the MCU to complete the wake-up to trigger the interrupt. If the level disappears, the MCU can wake up without any interrupt generated.

### 18.3.4 Events

PORT can generate the following events:

**Table 18-5. Event Generators in PORTx**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| PORTx | PINn | Pin level | Level | Asynchronous | Given by pin level |

All PORT pins are asynchronous event system generators. PORT has as many event generators as there are PORT pins in the device. Each event system output from PORT is the value present on the corresponding pin if the digital input buffer is enabled. If a pin input buffer is disabled, the corresponding event system output is zero.

PORT has no event inputs. Refer to the *Event System (EVSYS)* section for more details regarding event types and Event System configuration.

### 18.3.5 Sleep Mode Operation

Except for interrupts and input synchronization, all pin configurations are independent of sleep modes. All pins can wake the device from sleep, see the PORT Interrupt section for further details.

Peripherals connected to the PORTs can be affected by sleep modes, described in the respective peripherals’ data sheet section.

**Important:** The PORTs will always use the Peripheral Clock (CLK_PER). Input synchronization will halt when this clock stops.

### 18.3.6 Debug Operation

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

## 18.4 Register Summary - PORTx

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | DIR | 7:0 | DIR[7:0] | | | | | | | |
| 0x01 | DIRSET | 7:0 | DIRSET[7:0] | | | | | | | |
| 0x02 | DIRCLR | 7:0 | DIRCLR[7:0] | | | | | | | |
| 0x03 | DIRTGL | 7:0 | DIRTGL[7:0] | | | | | | | |
| 0x04 | OUT | 7:0 | OUT[7:0] | | | | | | | |
| 0x05 | OUTSET | 7:0 | OUTSET[7:0] | | | | | | | |
| 0x06 | OUTCLR | 7:0 | OUTCLR[7:0] | | | | | | | |
| 0x07 | OUTTGL | 7:0 | OUTTGL[7:0] | | | | | | | |
| 0x08 | IN | 7:0 | IN[7:0] | | | | | | | |
| 0x09 | INTFLAGS | 7:0 | INT[7:0] | | | | | | | |
| 0x0A | PORTCTRL | 7:0 | | | | | | | | SRL |
| 0x0B | PINCONFIG | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| 0x0C | PINCTRLUPD | 7:0 | PINCTRLUPD[7:0] | | | | | | | |
| 0x0D | PINCTRLSET | 7:0 | PINCTRLSET[7:0] | | | | | | | |
| 0x0E | PINCTRLCLR | 7:0 | PINCTRLCLR[7:0] | | | | | | | |
| 0x0F | Reserved | | | | | | | | | |
| 0x10 | PIN0CTRL | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| 0x11 | PIN1CTRL | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| 0x12 | PIN2CTRL | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| 0x13 | PIN3CTRL | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| 0x14 | PIN4CTRL | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| 0x15 | PIN5CTRL | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| 0x16 | PIN6CTRL | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| 0x17 | PIN7CTRL | 7:0 | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |

## 18.5 Register Description - PORTx

### 18.5.1 Data Direction

**Name:** DIR
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| 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 7:0 – DIR[7:0]** Data Direction
This bit field controls the output driver for each PORTx pin.
This bit field does not control the digital input buffer. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.
The available configuration for each bit n in this bit field is shown in the table below.

| Value | Description |
|---|---|
| 0 | Pxn is configured as an input-only pin, and the output driver is disabled |
| 1 | Pxn is configured as an output pin, and the output driver is enabled |

### 18.5.2 Data Direction Set

**Name:** DIRSET
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| 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 7:0 – DIRSET[7:0]** Data Direction Set
This bit field controls the output driver for each PORTx pin, without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will set the corresponding bit in PORTx.DIR, which will configure pin n (Pxn) as an output pin and enable the output driver.
Reading this bit field will return the value of PORTx.DIR.

### 18.5.3 Data Direction Clear

**Name:** DIRCLR
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| 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 7:0 – DIRCLR[7:0]** Data Direction Clear
This bit field controls the output driver for each PORTx pin, without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will clear the corresponding bit in PORTx.DIR, which will configure pin n (Pxn) as an input-only pin and disable the output driver.
Reading this bit field will return the value of PORTx.DIR.

### 18.5.4 Data Direction Toggle

**Name:** DIRTGL
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| 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 7:0 – DIRTGL[7:0]** Data Direction Toggle
This bit field controls the output driver for each PORTx pin, without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will toggle the corresponding bit in PORTx.DIR.
Reading this bit field will return the value of PORTx.DIR.

### 18.5.5 Output Value

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

| 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 7:0 – OUT[7:0]** Output Value
This bit field controls the output driver level for each PORTx pin.
This configuration only has an effect when the output driver (PORTx.DIR) is enabled for the corresponding pin.
The available configuration for each bit n in this bit field is shown in the table below.

| Value | Description |
|---|---|
| 0 | The pin n (Pxn) output is driven low |
| 1 | The Pxn output is driven high |

### 18.5.6 Output Value Set

**Name:** OUTSET
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

| 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 7:0 – OUTSET[7:0]** Output Value Set
This bit field controls the output driver level for each PORTx pin, without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will set the corresponding bit in PORTx.OUT, which will configure the output for pin n (Pxn) to be driven high.
Reading this bit field will return the value of PORTx.OUT.

### 18.5.7 Output Value Clear

**Name:** OUTCLR
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

| 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 7:0 – OUTCLR[7:0]** Output Value Clear
This bit field controls the output driver level for each PORTx pin, without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will clear the corresponding bit in PORTx.OUT, which will configure the output for pin n (Pxn) to be driven low.
Reading this bit field will return the value of PORTx.OUT.

### 18.5.8 Output Value Toggle

**Name:** OUTTGL
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

| 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 7:0 – OUTTGL[7:0]** Output Value Toggle
This bit field controls the output driver level for each PORTx pin, without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will toggle the corresponding bit in PORTx.OUT.
Reading this bit field will return the value of PORTx.OUT.

### 18.5.9 Input Value

**Name:** IN
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IN[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 – IN[7:0]** Input Value
This bit field shows the state of the PORTx pins when the digital input buffer is enabled.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will toggle the corresponding bit in PORTx.OUT.
If the digital input buffer is disabled, the input is not sampled, and the bit value will not change. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.
The available states of each bit n in this bit field is shown in the table below.

| Value | Description |
|---|---|
| 0 | The voltage level on Pxn is low |
| 1 | The voltage level on Pxn is high |

### 18.5.10 Interrupt Flags

**Name:** INTFLAGS
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INT[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 – INT[7:0]** Pin Interrupt Flag
Pin interrupt flag n is cleared by writing a '1' to it.
Pin interrupt flag n is set when the change or state of pin n (Pxn) matches the pin's Input/Sense Configuration (ISC) in PORTx.PINnCTRL.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will clear Pin interrupt flag n.

### 18.5.11 Port Control

**Name:** PORTCTRL
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

This register contains the slew rate limit enable bit for this port.

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

**Bit 0 – SRL** Slew Rate Limit Enable
This bit controls the slew rate limitation for all pins in PORTx.

| Value | Description |
|---|---|
| 0 | Slew rate limitation is disabled for all pins in PORTx |
| 1 | Slew rate limitation is enabled for all pins in PORTx |

### 18.5.12 Multi-Pin Configuration

**Name:** PINCONFIG
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

The multi-pin configuration write enables the configuration of several pins of a port in a single cycle, for faster configuration of the port module. Especially with large pin count devices, this function can significantly speed up port pin configuration operations.

Writing to this register may be followed by a write to either of the Multi-Pin Control (PORTx.PINCTRLUPD/SET/CLR) registers to update the Pin n Control (PORTx.PINnCTRL) registers for PORTx.

This register is mirrored across all PORTx modules.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| Access | R/W | R/W | | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | | | 0 | 0 | 0 | 0 |

**Bit 7 – INVEN** Inverted I/O Enable
This bit controls whether the input and output for pin n are inverted or not.

| Value | Description |
|---|---|
| 0 | Input and output values are not inverted |
| 1 | Input and output values are inverted |

**Bit 6 – INLVL** Input Level Select
This bit controls the input voltage threshold for pin n, used for port input reads and interrupt conditions.

| Value | Name | Description |
|---|---|---|
| 0 | ST | Schmitt Trigger derived from supply level |
| 1 | TTL | TTL Levels |

**Bit 3 – PULLUPEN** Pull-up Enable
This bit controls whether the internal pull-up of pin n is enabled or not when the pin is configured as input-only.

| Value | Description |
|---|---|
| 0 | Pull-up disabled |
| 1 | Pull-up enabled |

**Bits 2:0 – ISC[2:0]** Input/Sense Configuration
This bit field controls the input and sense configuration of pin n. The sense configuration determines how a port interrupt can be triggered.

| Value | Name | Description |
|---|---|---|
| 0x0 | INTDISABLE | Interrupt disabled but digital input buffer enabled |
| 0x1 | BOTHEDGES | Interrupt enabled with sense on both edges |
| 0x2 | RISING | Interrupt enabled with sense on rising edge |
| 0x3 | FALLING | Interrupt enabled with sense on falling edge |
| 0x4 | INPUT_DISABLE | Interrupt and digital input buffer disabled[1] |
| 0x5 | LEVEL | Interrupt enabled with sense on low level |
| other | — | Reserved |

**Note:**
1. If the digital input buffer for pin n is disabled, bit n in the Input Value (PORTx.IN) register will not be updated.

### 18.5.13 Multi-Pin Control Update Mask

**Name:** PINCTRLUPD
**Offset:** 0x0C
**Reset:** 0x00
**Property:** -

The multi-pin configuration write enables the configuration of several pins of a port in a single cycle, for faster configuration of the port module. Especially with large pin count devices, this function can significantly speed up port pin configuration operations.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PINCTRLUPD[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 – PINCTRLUPD[7:0]** Multi-Pin Control Update Mask
This bit field controls the copy of the Multi-Pin Configuration (PORTx.PINCONFIG) register content to the individual Pin n Control (PORTx.PINnCTRL) registers, without using an individual write operation for each register.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will copy the PORTx.PINCONFIG register content to the corresponding PORTx.PINnCTRL register.
Reading this bit field will always return zero.

### 18.5.14 Multi-Pin Control Set Mask

**Name:** PINCTRLSET
**Offset:** 0x0D
**Reset:** 0x00
**Property:** -

The multi-pin configuration write enables the configuration of several pins of a port in a single cycle, for faster configuration of the port module. Especially with large pin count devices, this function can significantly speed up port pin configuration operations.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PINCTRLSET[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 – PINCTRLSET[7:0]** Multi-Pin Control Set Mask
This bit field controls the setting of bits in the individual Pin n Control (PORTx.PINnCTRL) registers, without using an individual read-modify-write operation for each register.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will set the individual bits in the PORTx.PINnCTRL register, according to the bits set to '1' in the Multi-Pin Configuration (PORTx.PINCONFIG) register.
Reading this bit field will always return zero.

### 18.5.15 Multi-Pin Control Clear Mask

**Name:** PINCTRLCLR
**Offset:** 0x0E
**Reset:** 0x00
**Property:** -

The multi-pin configuration write enables the configuration of several pins of a port in a single cycle, for faster configuration of the port module. Especially with large pin count devices, this function can significantly speed up port pin configuration operations.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PINCTRLCLR[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 – PINCTRLCLR[7:0]** Multi-Pin Control Clear Mask
This bit field controls the clearing of bits in the individual Pin n Control (PORTx.PINnCTRL) registers, without using an individual read-modify-write operation for each register.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will clear the individual bits in the PORTx.PINnCTRL register, according to the bits set to '1' in the Multi-Pin Configuration (PORTx.PINCONFIG) register.
Reading this bit field will always return zero.

### 18.5.16 Pin n Control

**Name:** PINnCTRL
**Offset:** 0x10 + n*0x01 [n=0..7]
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INVEN | INLVL | | | PULLUPEN | ISC[2:0] | | |
| Access | R/W | R/W | | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | | | 0 | 0 | 0 | 0 |

**Bit 7 – INVEN** Inverted I/O Enable
This bit controls whether the input and output for pin n are inverted or not.

| Value | Description |
|---|---|
| 0 | Input and output values are not inverted |
| 1 | Input and output values are inverted |

**Bit 6 – INLVL** Input Level Select
This bit controls the input voltage threshold for pin n, used for port input reads and interrupt conditions.

| Value | Name | Description |
|---|---|---|
| 0 | ST | Schmitt Trigger derived from supply level |
| 1 | TTL | TTL Levels |

**Bit 3 – PULLUPEN** Pull-up Enable
This bit controls whether the internal pull-up of pin n is enabled or not when the pin is configured as input-only.

| Value | Description |
|---|---|
| 0 | Pull-up disabled |
| 1 | Pull-up enabled |

**Bits 2:0 – ISC[2:0]** Input/Sense Configuration
This bit field controls the input and sense configuration of pin n. The sense configuration determines how a port interrupt can be triggered.

| Value | Name | Description |
|---|---|---|
| 0x0 | INTDISABLE | Interrupt disabled but digital input buffer enabled |
| 0x1 | BOTHEDGES | Interrupt enabled with sense on both edges |
| 0x2 | RISING | Interrupt enabled with sense on rising edge |
| 0x3 | FALLING | Interrupt enabled with sense on falling edge |
| 0x4 | INPUT_DISABLE | Interrupt and digital input buffer disabled(1) |
| 0x5 | LEVEL | Interrupt enabled with sense on low level |
| other | — | Reserved |

**Note:**
1. If the digital input buffer for pin n is disabled, bit n in the Input Value (PORTx.IN) register will not be updated.

## 18.6 Register Summary - VPORTx

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | DIR | 7:0 | DIR[7:0] | | | | | | | |
| 0x01 | OUT | 7:0 | OUT[7:0] | | | | | | | |
| 0x02 | IN | 7:0 | IN[7:0] | | | | | | | |
| 0x03 | INTFLAGS | 7:0 | INT[7:0] | | | | | | | |

## 18.7 Register Description - VPORTx

### 18.7.1 Data Direction

**Name:** DIR
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

Access to the Virtual PORT registers has the same outcome as access to the regular registers but allows for memory specific instructions, such as bit manipulation instructions, which cannot be used in the extended I/O Register space where the regular PORT registers reside.

| 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 7:0 – DIR[7:0]** Data Direction
This bit field controls the output driver for each PORTx pin.
This bit field does not control the digital input buffer. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.
The available configuration for each bit n in this bit field is shown in the table below.

| Value | Description |
|---|---|
| 0 | Pxn is configured as an input-only pin, and the output driver is disabled |
| 1 | Pxn is configured as an output pin, and the output driver is enabled |

### 18.7.2 Output Value

**Name:** OUT
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

Access to the Virtual PORT registers has the same outcome as access to the regular registers but allows for memory specific instructions, such as bit manipulation instructions, which cannot be used in the extended I/O Register space where the regular PORT registers reside.

| 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 7:0 – OUT[7:0]** Output Value
This bit field controls the output driver level for each PORTx pin.
This configuration only has an effect when the output driver (PORTx.DIR) is enabled for the corresponding pin.
The available configuration for each bit n in this bit field is shown in the table below.

| Value | Description |
|---|---|
| 0 | The pin n (Pxn) output is driven low |
| 1 | The Pxn output is driven high |

### 18.7.3 Input Value

**Name:** IN
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

Access to the Virtual PORT registers has the same outcome as access to the regular registers but allows for memory specific instructions, such as bit manipulation instructions, which cannot be used in the extended I/O Register space where the regular PORT registers reside.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IN[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 – IN[7:0]** Input Value
This bit field shows the state of the PORTx pins when the digital input buffer is enabled.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will toggle the corresponding bit in PORTx.OUT.
If the digital input buffer is disabled, the input is not sampled, and the bit value will not change. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.
The available states of each bit n in this bit field is shown in the table below.

| Value | Description |
|---|---|
| 0 | The voltage level on Pxn is low |
| 1 | The voltage level on Pxn is high |

### 18.7.4 Interrupt Flags

**Name:** INTFLAGS
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

Access to the Virtual PORT registers has the same outcome as access to the regular registers but allows for memory specific instructions, such as bit manipulation instructions, which cannot be used in the extended I/O Register space where the regular PORT registers reside.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INT[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 – INT[7:0]** Pin Interrupt Flag
Pin interrupt flag n is cleared by writing a '1' to it.
Pin interrupt flag n is set when the change or state of pin n (Pxn) matches the pin's Input/Sense Configuration (ISC) in PORTx.PINnCTRL.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will clear Pin interrupt flag n.

# 19. MVIO - Multi-Voltage I/O

## 19.1 Features

- A Subset of the Device I/O Pins Can Be Powered by $V_{DDIO2}$
- The $V_{DDIO2}$ Supply Can Ramp Up and Down Independently of the $V_{DD}$ Supply
- Single- or Dual-Supply Configuration Determined by Fuse
- PORT Access and Peripheral Override Independent of the Supply Configuration
- $V_{DDIO2}$ Supply Status Bit
- Interrupt and Event for $V_{DDIO2}$ Supply Status Change
- ADC Channel for Measuring $V_{DDIO2}$ Supply Voltage

## 19.2 Overview

The MVIO feature allows a subset of the I/O pins to be powered by a different I/O voltage domain than the rest of the I/O pins. This eliminates the need of having external level shifters for communication or control of external components running on a different voltage level. The MVIO-capable I/O pads are supplied by a voltage applied to the VDDIO2 power pin(s), while the regular I/O pins are supplied by the voltage applied to the VDD pin(s).

The MVIO can be configured in one of two supply modes:

- Single-Supply mode, where the MVIO-capable I/O pins are powered at the same voltage level as the non-MVIO capable pins, i.e., $V_{DD}$. The user must connect the VDDIO2 pin(s) to the VDD pin(s).
- Dual-Supply mode, where the MVIO-capable I/O pins are supplied by the $V_{DDIO2}$ voltage, which may be different from the voltage supplied to the VDD pin(s).

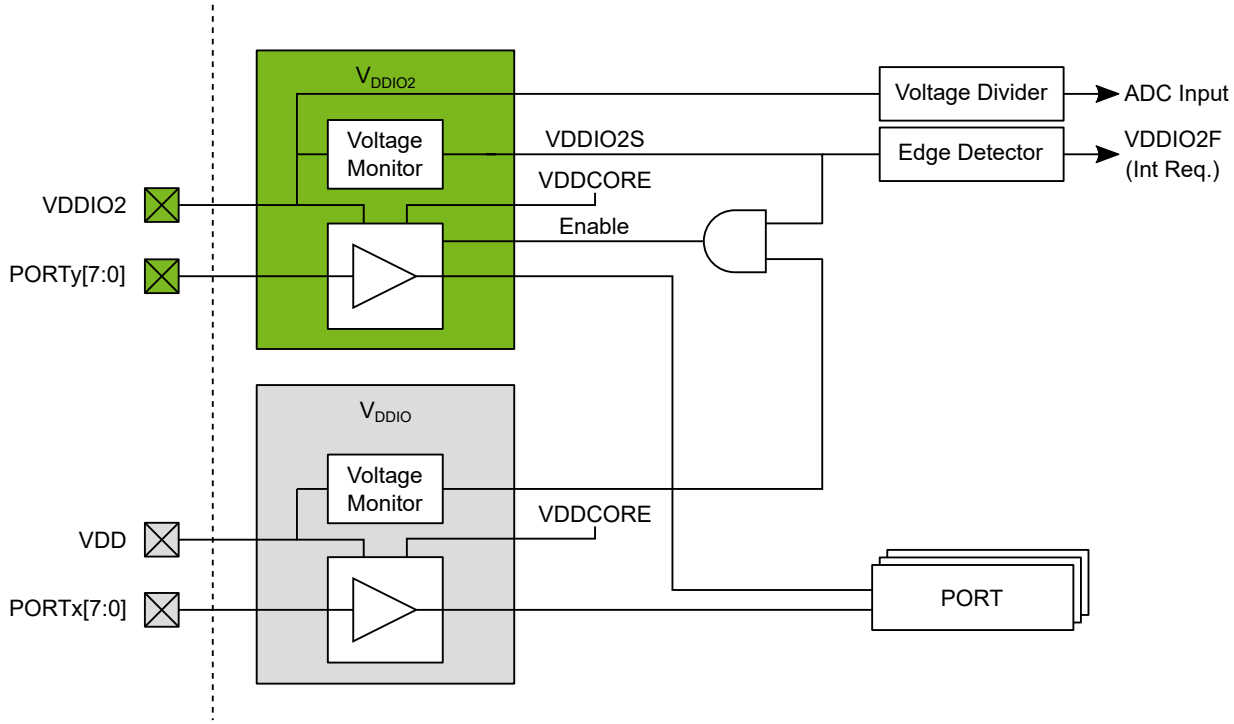
A configuration fuse determines the MVIO supply mode. The loss or gain of power on $V_{DDIO2}$ is signaled by a status register bit. This status bit has corresponding interrupt and event functionality.

The MVIO pins are capable of the same digital behavior as regular I/O pins, e.g., GPIO, serial communication (USART, SPI, $I^2C$), or connected to PWM peripherals. The input Schmitt trigger levels are scaled according to the $V_{DDIO2}$ voltage, as described in the *Electrical Characteristics* section of the data sheet.

A divided-down $V_{DDIO2}$ voltage is available as input to the ADC.

### 19.2.1 Block Diagram

**Figure 19-1. MVIO Block Diagram**




### 19.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| VDD | Power pin for $V_{DDIO}$ and other power domains | Supply |
| VDDIO2 | Power pin for $V_{DDIO2}$ | Supply |
| PORTx[7:0] | PORT pins powered by $V_{DDIO}$ | Input/Output |
| PORTy[7:0] | PORT pins powered by $V_{DDIO2}$ | Input/Output |

## 19.3 Functional Description

### 19.3.1 Initialization

Initialize the MVIO in Dual-Supply configuration by following these steps:

1. Program the Multi-Voltage System Configuration (MVSYSCFG) fuse to the Dual-Supply configuration.
2. Optional: Write the VDDIO2 Interrupt Enable (VDDIO2IE) bit to '1' in the Interrupt Control (MVIO.INTCTRL) register.
3. Read the VDDIO2 Status (VDDIO2S) bit in the Status (MVIO.STATUS) register to check if the $V_{DDIO2}$ voltage is within the acceptable range for operation.
4. Configure and use the PORT pins powered by $V_{DDIO2}$.

If the MVSYSCFG fuse is programmed to the Single-Supply configuration, the VDDIO2 Status bit is read as '1', and the VDDIO2 Interrupt Flag is read as '0'.

### 19.3.2 Operation

#### 19.3.2.1 Power Sequencing

The system supports the following power ramp scenarios for MVIO when configured in Dual-Supply mode:

- Supply ramp of $V_{DDIO}$ before $V_{DDIO2}$
- Supply ramp of $V_{DDIO2}$ before $V_{DDIO}$
- $V_{DDIO2}$ loses and regains power
- $V_{DDIO}$ loses and regains power

When either voltage domain loses power, the MVIO I/O pins are tri-stated. If $V_{DDIO2}$ regains power, the pins will reload the current configuration of the PORT registers. If $V_{DDIO}$ loses power, the device will reset, and the PORTs will have to be reinitialized. Refer to the *Electrical Characteristics* section for $V_{DD}$ and $V_{DDIO2}$ power supply thresholds.

#### 19.3.2.2 Voltage Measurement

$V_{DDIO2}$ is available as an internal input channel to the ADC. The voltage is divided by 10 to allow the use of any internal ADC reference. To measure $V_{DDIO2}$, follow these steps:

- Configure the voltage reference for the ADC
- Select $V_{DDIO2}$ as the positive input to the ADC
- Run a single-ended ADC conversion
- Calculate the voltage using the following equation:

$$V_{\mathrm{DDIO2}} = \frac{\mathrm{ADC\ Result} \times V_{\mathrm{REF}} \times 10}{\mathrm{ADC\ Resolution}}$$

### 19.3.3 Events

The MVIO can generate the following events:

**Table 19-1. Event Generators in MVIO**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| MVIO | VDDIO2OK | $V_{DDIO2}$ level is above threshold | Level | Asynchronous | Given by the VDDIO2 Status (VDDIO2S) bit in the Status (MVIO.STATUS) register |

The MVIO has no event users. Refer to the *Event System (EVSYS)* section for more details regarding event types and Event System configuration.

### 19.3.4 Interrupts

**Table 19-2. Available Interrupt Vectors and Sources**

| Name | Vector Description | Interrupt Flag | Conditions |
|---|---|---|---|
| MVIO | VDDIO2 interrupt | VDDIO2IF | VDDIO2S toggles |

A change in the VDDIO2 Status (VDDIO2S) bit in the Status (MVIO.STATUS) register can be configured to trigger an interrupt. This can be enabled or disabled by writing to the VDDIO2 Interrupt Enable (VDDIO2IE) bit in the Interrupt Control (MVIO.INTCTRL) register.

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

### 19.3.5 Sleep Mode Operation

When enabled by the Multi-Voltage System Configuration (MVSYSCFG) fuse, the module will operate in all sleep modes.

### 19.3.6 Debug Operation

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

## 19.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | INTCTRL | 7:0 | | | | | | | | VDDIO2IE |
| 0x01 | INTFLAGS | 7:0 | | | | | | | | VDDIO2IF |
| 0x02 | STATUS | 7:0 | | | | | | | | VDDIO2S |

## 19.5 Register Description

### 19.5.1 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | VDDIO2IE |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – VDDIO2IE** VDDIO2 Interrupt Enable
This bit controls whether the interrupt for a VDDIO2 Status change is enabled or not.

| Value | Description |
|---|---|
| 0 | The VDDIO2 interrupt is disabled |
| 1 | The VDDIO2 interrupt is enabled |

### 19.5.2 Interrupt Flags

**Name:** INTFLAGS
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | VDDIO2IF |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – VDDIO2IF** VDDIO2 Interrupt Flag
This flag is cleared by writing a '1' to it.
This flag is set when the VDDIO2 Status (VDDIO2S) bit in MVIO.STATUS changes value.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the VDDIO2 Interrupt Flag.

### 19.5.3 Status

**Name:** STATUS
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | VDDIO2S |
| Access | | | | | | | | R |
| Reset | | | | | | | | 0 |

**Bit 0 – VDDIO2S** VDDIO2 Status
This bit shows the state of the $V_{DDIO2}$ voltage level.
Writing to this bit has no effect.

| Value | Description |
|---|---|
| 0 | The $V_{DDIO2}$ supply voltage is below the acceptable range for operation. The MVIO pins are tri-stated. |
| 1 | The $V_{DDIO2}$ supply voltage is within the acceptable range for operation. The MVIO pin configurations are loaded from the corresponding PORT registers. |

# 20. BOD - Brown-out Detector

## 20.1 Features

- Brown-out Detector Monitors the Power Supply to Avoid Operation Below a Programmable Level
- Three Available Modes:
  - Enabled mode (continuously active)
  - Sampled mode
  - Disabled
- Separate Selection of Mode for Active and Sleep Modes
- Voltage Level Monitor (VLM) with Interrupt
- Programmable VLM Level Relative to the BOD Level

## 20.2 Overview

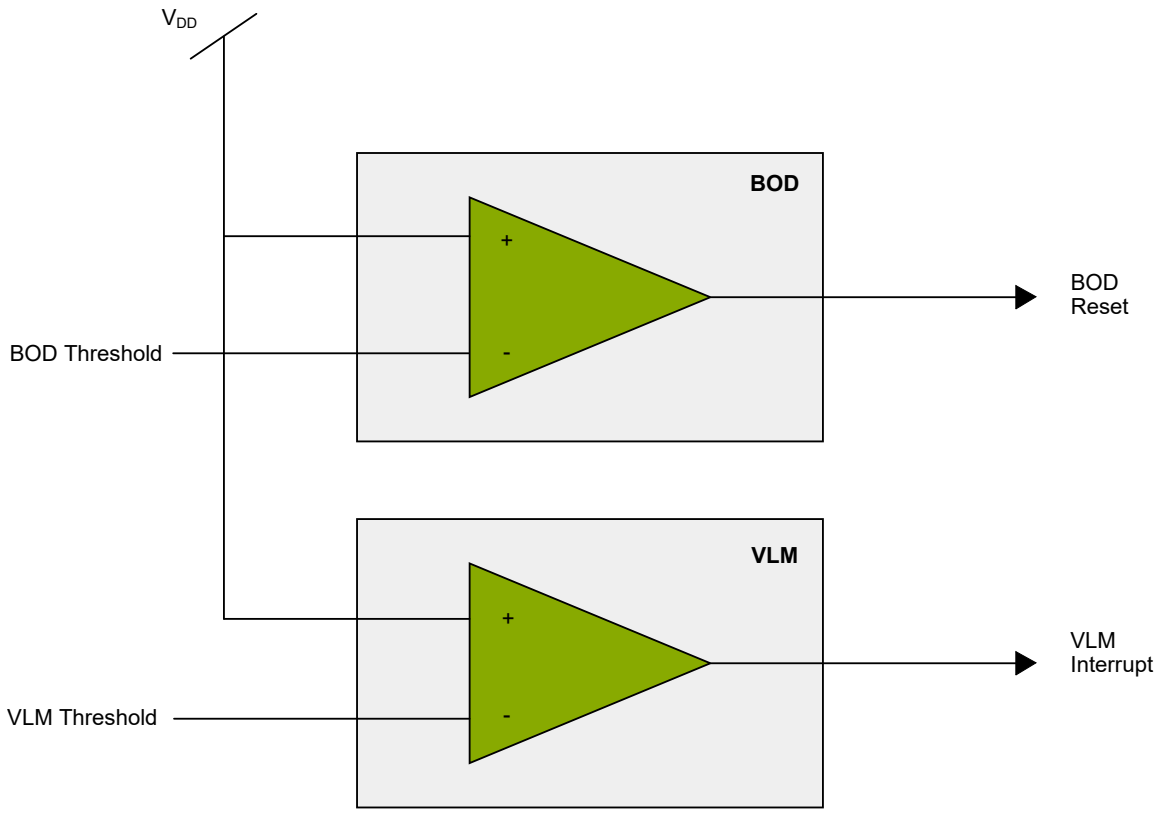
The Brown-out Detector (BOD) monitors the power supply and compares the supply voltage with the programmable brown-out threshold level. The brown-out threshold level defines when to generate a System Reset. The Voltage Level Monitor (VLM) monitors the power supply and compares it to a threshold higher than the BOD threshold. The VLM can then generate an interrupt as an “early warning” when the supply voltage is approaching the BOD threshold. The VLM threshold level is expressed as a percentage above the BOD threshold level.

The BOD is controlled mainly by fuses and has to be enabled by the user. The mode used in Standby sleep mode and Power-Down sleep mode can be altered in normal program execution. The VLM is controlled by I/O registers as well.

When activated, the BOD can operate in Enabled mode, where the BOD is continuously active, or in Sampled mode, where the BOD is activated briefly at a given period to check the supply voltage level.

### 20.2.1 Block Diagram

**Figure 20-1. BOD Block Diagram**




## 20.3 Functional Description

### 20.3.1 Initialization

The BOD settings are loaded from fuses during Reset. The BOD level and operating mode in Active mode and Idle sleep mode are set by fuses and cannot be changed by software. The operating mode in Standby and Power-Down sleep mode is loaded from fuses and can be changed by software.

The Voltage Level Monitor function can be enabled by writing a '`1`' to the VLM Interrupt Enable (VLMIE) bit in the Interrupt Control (BOD.INTCTRL) register. The VLM interrupt is configured by writing the VLM Configuration (VLMCFG) bits in BOD.INTCTRL. An interrupt is requested when the supply voltage crosses the VLM threshold from either above or below.

The VLM functionality will follow the BOD mode. If the BOD is disabled, the VLM will not be enabled, even if the VLMIE is '`1`'. If the BOD is using Sampled mode, the VLM will also be sampled. When enabling the VLM interrupt, the interrupt flag will always be set if VLMCFG equals `0x2`, and may be set if VLMCFG is configured to `0x0` or `0x1`.

The VLM threshold is defined by writing the VLM Level (VLMLVL) bits in the VLM Control (BOD.VLMCTRL) register.

### 20.3.2 Interrupts

**Table 20-1. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| VLM | Voltage Level Monitor | Supply voltage crossing the VLM threshold as configured by the VLM Configuration (VLMCFG) bit field in the Interrupt Control (BOD.INTCTRL) register |

The VLM interrupt will not be executed if the CPU is halted in Debug mode.

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

### 20.3.3 Sleep Mode Operation

The BOD configuration in the different sleep modes is defined by fuses. The mode used in Active mode and Idle sleep mode is defined by the ACTIVE fuses in FUSE.BODCFG, which is loaded into the ACTIVE bit field in the Control A (BOD.CTRLA) register. The mode used in Standby sleep mode and Power-Down sleep mode is defined by SLEEP in FUSE.BODCFG, which is loaded into the SLEEP bit field in the Control A (BOD.CTRLA) register.

The operating mode in Active mode and Idle sleep mode (i.e., ACTIVE in BOD.CTRLA) cannot be altered by software. The operating mode in Standby sleep mode and Power-Down sleep mode can be altered by writing to the SLEEP bit field in the Control A (BOD.CTRLA) register.

When the device is going into Standby or Power-Down sleep mode, the BOD will change the operation mode as defined by SLEEP in BOD.CTRLA. When the device is waking up from Standby or Power-Down sleep mode, the BOD will operate in the mode defined by the ACTIVE bit field in the Control A (BOD.CTRLA) register.

### 20.3.4 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a certain key must first be written to the CPU.CCP register, followed by a write access to the protected bits within four CPU instructions.

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged.

The following registers are under CCP:

**Table 20-2. Registers Under Configuration Change Protection**

| Register | Key |
|---|---|
| The SLEEP and SAMPFREQ bits in the BOD.CTRLA register | IOREG |

## 20.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | | | SAMPFREQ | ACTIVE[1:0] | | SLEEP[1:0] | |
| 0x01 | CTRLB | 7:0 | | | | | | LVL[2:0] | | |
| 0x02<br>...<br>0x07 | Reserved | | | | | | | | | |
| 0x08 | VLMCTRL | 7:0 | | | | | | | VLMLVL[1:0] | |
| 0x09 | INTCTRL | 7:0 | | | | | | VLMCFG[1:0] | | VLMIE |
| 0x0A | INTFLAGS | 7:0 | | | | | | | | VLMIF |
| 0x0B | STATUS | 7:0 | | | | | | | | VLMS |

## 20.5 Register Description

### 20.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | SAMPFREQ | ACTIVE[1:0] | | SLEEP[1:0] | |
| Access | | | | R | R | R | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bit 4 – SAMPFREQ** Sample Frequency
This bit controls the BOD sample frequency.
The Reset value is loaded from the SAMPFREQ bit in FUSE.BODCFG.
This bit is not under Configuration Change Protection (CCP).

| Value | Description |
|---|---|
| 0x0 | Sample frequency is 128 Hz |
| 0x1 | Sample frequency is 32 Hz |

**Bits 3:2 – ACTIVE[1:0]** Active
These bits select the BOD operation mode when the device is in Active mode or Idle sleep mode.
The Reset value is loaded from the ACTIVE bits in FUSE.BODCFG.
These bits are not under Configuration Change Protection (CCP).

| Value | Name | Description |
|---|---|---|
| 0x0 | DIS | Disabled |
| 0x1 | ENABLED | Enabled in Continuous mode |
| 0x2 | SAMPLE | Enabled in Sampled mode |
| 0x3 | ENWAKE | Enabled in Continuous mode. Execution is halted at wake-up until BOD is running. |

**Bits 1:0 – SLEEP[1:0]** Sleep
These bits select the BOD operation mode when the device is in Standby or Power-Down sleep mode. The Reset value is loaded from the SLEEP bits in FUSE.BODCFG.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIS | Disabled |
| 0x1 | ENABLED | Enabled in Continuous mode |
| 0x2 | SAMPLED | Enabled in Sampled mode |
| 0x3 | - | Reserved |

### 20.5.2 Control B

**Name:** CTRLB
**Offset:** 0x01
**Reset:** Loaded from fuse
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | LVL[2:0] | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | x | x | x |

**Bits 2:0 – LVL[2:0]** BOD Level
This bit field controls the BOD threshold level.
The Reset value is loaded from the BOD Level (LVL) bits in the BOD Configuration Fuse (FUSE.BODCFG).

| Value | Name | Typical Values |
|---|---|---|
| 0x0 | BODLEVEL0 | 1.9V |
| 0x1 | BODLEVEL1 | 2.45V |
| 0x2 | BODLEVEL2 | 2.7V |
| 0x3 | BODLEVEL3 | 2.85V |
| Other | — | Reserved |

**Note:** Refer to the *Reset, WDT, Oscillator, Start-up Timer, Power-up Timer, Brown-out Detector Specifications* section for BOD level characterization.

### 20.5.3 VLM Control

**Name:** VLMCTRL
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

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

**Bits 1:0 – VLMLVL[1:0]** VLM Level
These bits select the VLM threshold relative to the BOD threshold (LVL in BOD.CTRLB).

| Value | Name | Description |
|---|---|---|
| 0x00 | OFF | VLM disabled |
| 0x01 | 5ABOVE | VLM threshold 5% above the BOD threshold |
| 0x02 | 15ABOVE | VLM threshold 15% above the BOD threshold |
| 0x03 | 25ABOVE | VLM threshold 25% above the BOD threshold |

### 20.5.4 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | VLMCFG[1:0] | | VLMIE |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:1 – VLMCFG[1:0]** VLM Configuration
These bits select which incidents will trigger a VLM interrupt.

| Value | Name | Description |
|---|---|---|
| `0x0` | FALLING | $V_{DD}$ falls below VLM threshold |
| `0x1` | RISING | $V_{DD}$ rises above VLM threshold |
| `0x2` | BOTH | $V_{DD}$ crosses VLM threshold |
| `Other` | - | Reserved |

**Bit 0 – VLMIE** VLM Interrupt Enable
Writing a '`1`' to this bit enables the VLM interrupt.

### 20.5.5 VLM Interrupt Flags

**Name:** INTFLAGS
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

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

**Bit 0 – VLMIF** VLM Interrupt Flag
This flag is set when a trigger from the VLM is given, as configured by the VLMCFG bit in the BOD.INTCTRL register. The flag is only updated when the BOD is enabled.

### 20.5.6 VLM Status

**Name:** STATUS
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

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

**Bit 0 – VLMS** VLM Status
This bit is only valid when the BOD is enabled.

| Value | Name | Description |
|---|---|---|
| 0 | ABOVE | The voltage is above the VLM threshold level |
| 1 | BELOW | The voltage is below the VLM threshold level |

# 21. VREF - Voltage Reference

## 21.1 Features

- Programmable Voltage Reference Sources:
  - One reference for Analog to Digital Converter 0 (ADC0)
  - One reference for Digital to Analog Converter 0 (DAC0)
  - One reference shared between all Analog Comparators (ACs)
- Each Reference Source Supports the Following Voltages:
  - 1.024V
  - 2.048V
  - 4.096V
  - 2.500V
  - VDD
  - VREFA

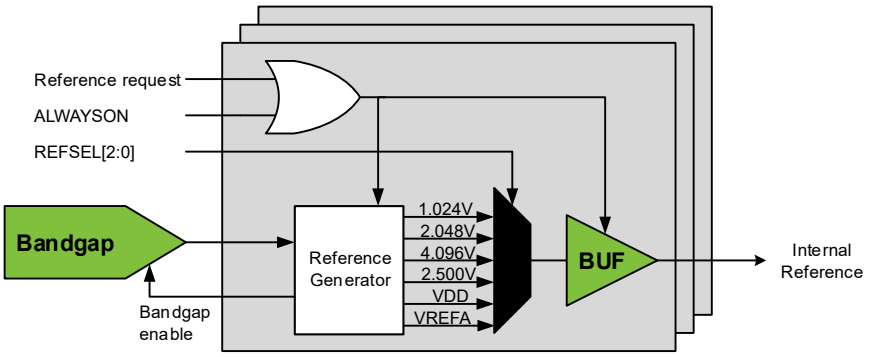
## 21.2 Overview

The Voltage Reference (VREF) peripheral provides control registers for the voltage reference sources used by several peripherals. The user can select the reference voltages for the ADC0, DAC0 and ACs by writing to the appropriate registers in the VREF peripheral.

A voltage reference source is enabled automatically when requested by a peripheral. The user can enable the reference voltage sources, and thus, override the automatic disabling of unused sources by writing to the respective ALWAYSON bit in VREF.ADC0REF, VREF.DAC0REF and VREF.ACREF. This will decrease the start-up time at the cost of increased power consumption.

### 21.2.1 Block Diagram

**Figure 21-1. VREF Block Diagram**




## 21.3 Functional Description

### 21.3.1 Initialization

The default configuration will enable the respective source when the ADC0, DAC0, or any of the ACs are requesting a reference voltage. The default reference voltage is 1.024V but can be configured by writing to the respective Reference Select (REFSEL) bit field in the ADC0 Reference (ADC0REF), DAC0 Reference (DAC0REF) or Analog Comparators (ACREF) registers.

## 21.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | ADC0REF | 7:0 | ALWAYSON | | | | | REFSEL[2:0] | | |
| 0x01 | Reserved | | | | | | | | | |
| 0x02 | DAC0REF | 7:0 | ALWAYSON | | | | | REFSEL[2:0] | | |
| 0x03 | Reserved | | | | | | | | | |
| 0x04 | ACREF | 7:0 | ALWAYSON | | | | | REFSEL[2:0] | | |

## 21.5 Register Description

### 21.5.1 ADC0 Reference

**Name:** ADC0REF
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ALWAYSON | | | | | REFSEL[2:0] | | |
| Access | R/W | | | | | R/W | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

**Bit 7 – ALWAYSON** Reference Always On
This bit controls whether the ADC0 reference is always on or not.

| Value | Description |
|---|---|
| 0 | The reference is automatically enabled when needed |
| 1 | The reference is always on |

**Bits 2:0 – REFSEL[2:0]** Reference Select
This bit field controls the reference voltage level for ADC0.

| Value | Name | Description |
|---|---|---|
| 0x0 | 1V024 | Internal 1.024V reference(1) |
| 0x1 | 2V048 | Internal 2.048V reference(1) |
| 0x2 | 4V096 | Internal 4.096V reference(1) |
| 0x3 | 2V500 | Internal 2.500V reference(1) |
| 0x4 | - | Reserved |
| 0x5 | VDD | VDD as reference |
| 0x6 | VREFA | External reference from the VREFA pin |
| 0x7 | - | Reserved |

**Note:**

1. The values given for internal references are only typical. Refer to the *Electrical Characteristics* section for further details.

### 21.5.2 DAC0 Reference

**Name:** DAC0REF
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ALWAYSON | | | | | REFSEL[2:0] | | |
| Access | R/W | | | | | R/W | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

**Bit 7 – ALWAYSON** Reference Always On
This bit controls whether the DAC0 reference is always on or not.

| Value | Description |
|---|---|
| 0 | The reference is automatically enabled when needed |
| 1 | The reference is always on |

**Bits 2:0 – REFSEL[2:0]** Reference Select
This bit field controls the reference voltage level for DAC0.

| Value | Name | Description |
|---|---|---|
| 0x0 | 1V024 | Internal 1.024V reference(1) |
| 0x1 | 2V048 | Internal 2.048V reference(1) |
| 0x2 | 4V096 | Internal 4.096V reference(1) |
| 0x3 | 2V500 | Internal 2.500V reference(1) |
| 0x4 | - | Reserved |
| 0x5 | VDD | VDD as reference |
| 0x6 | VREFA | External reference from the VREFA pin |
| 0x7 | - | Reserved |

**Note:**

1. The values given for internal references are only typical. Refer to the *Electrical Characteristics* section for further details.

### 21.5.3 Analog Comparator Reference

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ALWAYSON | | | | | REFSEL[2:0] | | |
| Access | R/W | | | | | R/W | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

**Bit 7 – ALWAYSON** Reference Always On
This bit controls whether the ACs reference is always on or not.

| Value | Description |
|---|---|
| 0 | The reference is automatically enabled when needed |
| 1 | The reference is always on |

**Bits 2:0 – REFSEL[2:0]** Reference Select
This bit field controls the reference voltage level for ACs.

| Value | Name | Description |
|---|---|---|
| 0x0 | 1V024 | Internal 1.024V reference(1) |
| 0x1 | 2V048 | Internal 2.048V reference(1) |
| 0x2 | 4V096 | Internal 4.096V reference(1) |
| 0x3 | 2V500 | Internal 2.500V reference(1) |
| 0x4 | - | Reserved |
| 0x5 | VDD | VDD as reference |
| 0x6 | VREFA | External reference from the VREFA pin |
| 0x7 | - | Reserved |

**Note:**

1. The values given for internal references are only typical. Refer to the *Electrical Characteristics* section for further details.

# 22. WDT - Watchdog Timer

## 22.1 Features

- Issues a System Reset if the Watchdog Timer is not Cleared Before its Time-out Period
- Operates Asynchronously from the Peripheral Clock Using an Independent Oscillator
- Uses the 1.024 kHz Output of the 32.768 kHz Ultra Low-Power Oscillator (OSC32K)
- 11 Selectable Time-out Periods, from 8 ms to 8s
- Two Operation Modes:
  - Normal mode
  - Window mode
- Configuration Lock to Prevent Unwanted Changes

## 22.2 Overview

The Watchdog Timer (WDT) is a system function for monitoring the correct program operation. When enabled, the WDT is a constantly running timer with a configurable time-out period. If the WDT is not reset within the time-out period, it will issue a system Reset. This allows the system to recover from situations such as runaway or deadlocked code. The WDT is reset by executing the `WDR` (Watchdog Timer Reset) instruction from software.

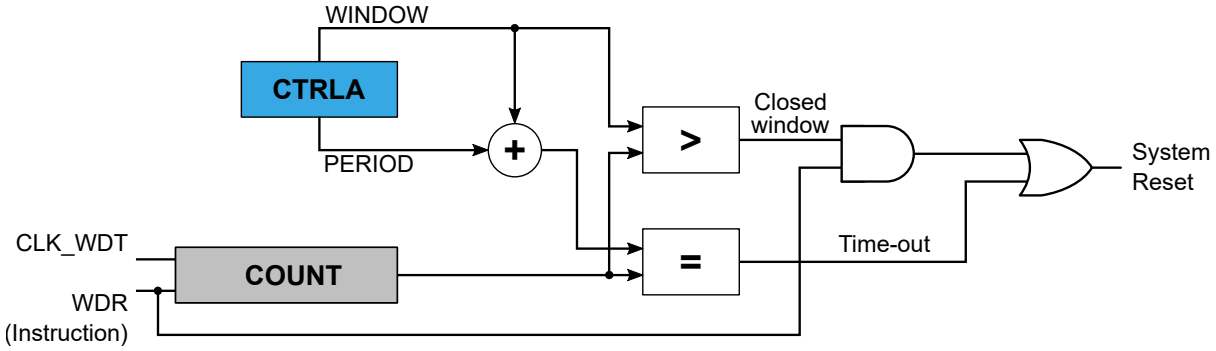
In addition to the Normal mode as described above, the WDT has a Window mode. The Window mode defines a time slot or “window” inside the time-out period during which the WDT must be reset. If the WDT is reset outside this window, either too early or too late, a system Reset will be issued. Compared to the Normal mode, the Window mode can catch situations where a code error causes constant `WDR` execution.

When enabled, the WDT will run in Active mode and all sleep modes. Since it is asynchronous (that is running from a CPU independent clock source), it will continue to operate and be able to issue a system Reset, even if the main clock fails.

The WDT has a Configuration Change Protection (CCP) mechanism and a lock functionality, ensuring the WDT settings cannot be changed by accident.

### 22.2.1 Block Diagram

**Figure 22-1. WDT Block Diagram**




## 22.3 Functional Description

### 22.3.1 Initialization

1. The WDT is enabled when a non-zero value is written to the Period (PERIOD) bit field in the Control A (WDT.CTRLA) register.

2. Optional: Write a non-zero value to the Window (WINDOW) bit field in WDT.CTRLA to enable the Window mode operation.

All bits in the Control A register and the Lock (LOCK) bit in the Status (WDT.STATUS) register are write-protected by the Configuration Change Protection (CCP) mechanism.

A fuse (FUSE.WDTCFG) defines the Reset value of the WDT.CTRLA register. If the value of the PERIOD bit field in the FUSE.WDTCFG fuse is different than zero, the WDT is enabled and the LOCK bit in the WDT.STATUS register is set at boot time.

### 22.3.2 Clocks

A 1.024 kHz clock (CLK_WDT) is sourced from the internal Ultra Low-Power Oscillator, OSC32K. Due to the ultra low-power design, the oscillator is less accurate than other oscillators featured in the device, and hence the exact time-out period may vary from device to device. This variation must be taken into consideration when designing software that uses the WDT, to ensure that the time-out periods used are valid for all devices. Refer to the *Electrical Characteristics* section for more specific information.

The WDT clock (CLK_WDT) is asynchronous to the peripheral clock. Due to this asynchronicity, writing to the WDT Control A (WDT.CTRLA) register will require synchronization between the clock domains. Refer to 22.3.6 Synchronization for further details.

### 22.3.3 Operation

#### 22.3.3.1 Normal Mode

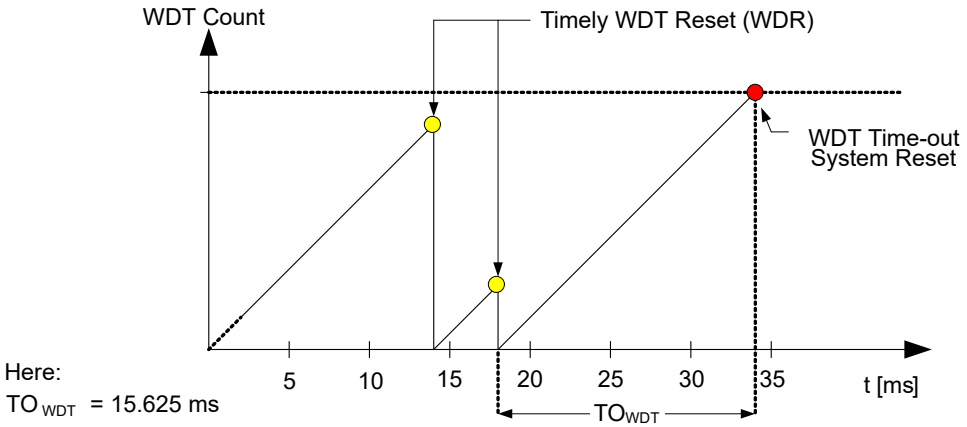
In the Normal mode operation, a single time-out period is set for the WDT. If the WDT is not reset from software using the `WDR` instruction during the defined time-out period, the WDT will issue a system Reset.

A new WDT time-out period starts each time the WDT is reset by software using the `WDR` instruction.

There are 11 possible WDT time-out periods ($TO_{WDT}$), selectable from 8 ms to 8s by writing to the Period (PERIOD) bit field in the Control A (WDT.CTRLA) register.

The figure below shows a typical timing scheme for the WDT operating in Normal mode.

**Figure 22-2. Normal Mode Operation**




The Normal mode is enabled as long as the Window (WINDOW) bit field in the WDT.CTRLA register is '`0x0`'.

#### 22.3.3.2 Window Mode

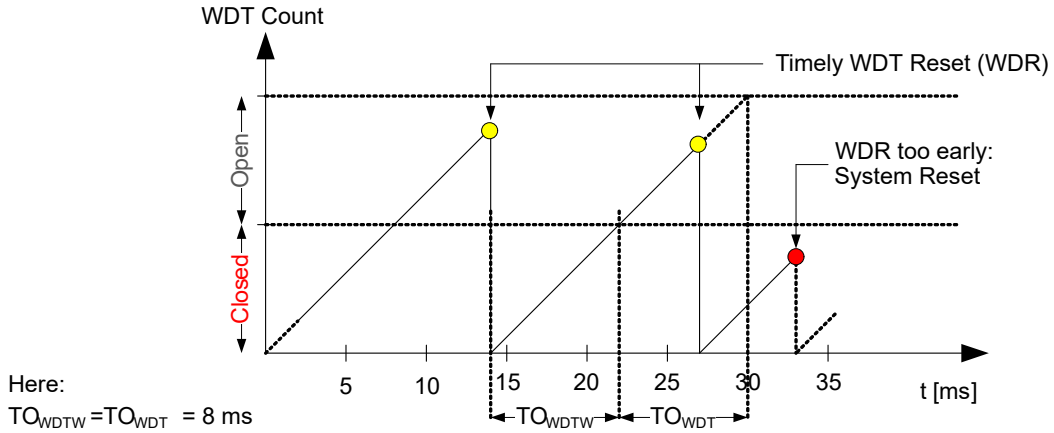
In the Window mode operation, the WDT uses two different time-out periods:

- The closed window time-out period ($TO_{WDTW}$) defines a duration, from 8 ms to 8s, where the WDT cannot be reset. If the WDT is reset during this period, the WDT will issue a system Reset.
- The open window time-out period ($TO_{WDT}$), which is also 8 ms to 8s, defines the duration of the open period during which the WDT can (and needs to) be reset. The open period will always follow the closed period, so the total duration of the time-out period is the sum of the closed window and the open window time-out periods.

When enabling the Window mode or when going out of the Debug mode, the window is activated after the first `WDR` instruction.

The figure below shows a typical timing scheme for the WDT operating in Window mode.

**Figure 22-3. Window Mode Operation**




The Window mode is enabled by writing a non-zero value to the Window (WINDOW) bit field in the Control A (WDT.CTRLA) register. The Window mode is disabled by writing the WINDOW bit field to '`0x0`'.

#### 22.3.3.3 Preventing Unintentional Changes

The WDT provides two security mechanisms to avoid unintentional changes to the WDT settings:

- The CCP mechanism, employing a timed write procedure for changing the WDT control registers. Refer to 22.3.7 Configuration Change Protection for further details.
- Locking the configuration by writing a '`1`' to the Lock (LOCK) bit in the Status (WDT.STATUS) register. When this bit is '`1`', the Control A (WDT.CTRLA) register cannot be changed. The LOCK bit can only be written to '`1`' in software, while the device needs to be in Debug mode to be able to write it to '`0`'. Consequently, the WDT cannot be disabled from software.

**Note:** The WDT configuration is loaded from fuses after Reset. If the PERIOD bit field is set to a non-zero value, the LOCK bit is automatically set in WDT.STATUS.

### 22.3.4 Sleep Mode Operation

The WDT will continue to operate in any sleep mode where the source clock is active.

### 22.3.5 Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in Debugging mode will halt the normal operation of the peripheral.

When halting the CPU in Debug mode, the WDT counter is reset.

When starting the CPU and when the WDT is operating in Window mode, the first closed window time-out period will be disabled, and a Normal mode time-out period is executed.

### 22.3.6 Synchronization

The Control A (WDT.CTRLA) register is synchronized when written, due to the asynchronicity between the WDT clock domain and the peripheral clock domain. The Synchronization Busy (SYNCBUSY) flag in the STATUS (WDT.STATUS) register indicates if there is an ongoing synchronization.

Writing to WDT.CTRLA while SYNCBUSY = `1` is not allowed.

The following bit fields are synchronized when written:

- The Period (PERIOD) bit field in Control A (WDT.CTRLA) register
- The Window (WINDOW) bit field in Control A (WDT.CTRLA) register

The `WDR` instruction will need two to three cycles of the WDT clock to be synchronized.

### 22.3.7 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a certain key must first be written to the CPU.CCP register, followed by a write access to the protected bits within four CPU instructions.

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged.

The following registers are under CCP:

**Table 22-1. WDT - Registers Under Configuration Change Protection**

| Register | Key |
|---|---|
| WDT.CTRLA | IOREG |
| LOCK bit in WDT.STATUS | IOREG |

## 22.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | WINDOW[3:0] | | | | PERIOD[3:0] | | | |
| 0x01 | STATUS | 7:0 | LOCK | | | | | | | SYNCBUSY |

## 22.5 Register Description

### 22.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** From FUSE.WDTCFG
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WINDOW[3:0] | | | | PERIOD[3:0] | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:4 – WINDOW[3:0]** Window
Writing a non-zero value to these bits enables the Window mode and selects the duration of the closed period accordingly.
The bits are optionally lock-protected:

- If the LOCK bit in WDT.STATUS is '1', all bits are change-protected (Access = R)
- If the LOCK bit in WDT.STATUS is '0', all bits can be changed (Access = R/W)

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | - |
| 0x1 | 8CLK | 7.8125 ms |
| 0x2 | 16CLK | 15.625 ms |
| 0x3 | 32CLK | 31.25 ms |
| 0x4 | 64CLK | 62.5 ms |
| 0x5 | 128CLK | 0.125s |
| 0x6 | 256CLK | 0.250s |
| 0x7 | 512CLK | 0.500s |
| 0x8 | 1KCLK | 1.0s |
| 0x9 | 2KCLK | 2.0s |
| 0xA | 4KCLK | 4.0s |
| 0xB | 8KCLK | 8.0s |
| Other | - | Reserved |

**Note:** Refer to the *Electrical Characteristics* section for specific information regarding the accuracy of the 32.768 kHz Ultra Low-Power Oscillator (OSC32K).

**Bits 3:0 – PERIOD[3:0]** Period
Writing a non-zero value to this bit enables the WDT and selects the time-out period in the Normal mode accordingly. In the Window mode, these bits select the duration of the open window.
The bits are optionally lock-protected:

- If the LOCK bit in WDT.STATUS is '1', all bits are change-protected (Access = R)
- If the LOCK bit in WDT.STATUS is '0', all bits can be changed (Access = R/W)

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | - |
| 0x1 | 8CLK | 7.8125 ms |
| 0x2 | 16CLK | 15.625 ms |
| 0x3 | 32CLK | 31.25 ms |
| 0x4 | 64CLK | 62.5 ms |
| 0x5 | 128CLK | 0.125s |
| 0x6 | 256CLK | 0.250s |
| 0x7 | 512CLK | 0.500s |
| 0x8 | 1KCLK | 1.0s |
| 0x9 | 2KCLK | 2.0s |
| 0xA | 4KCLK | 4.0s |

| Value | Name | Description |
|---|---|---|
| 0xB | 8KCLK | 8.0s |
| Other | - | Reserved |

**Note:** Refer to the *Electrical Characteristics* section for specific information regarding the accuracy of the 32.768 kHz Ultra Low-Power Oscillator (OSC32K).

### 22.5.2 Status

**Name:** STATUS
**Offset:** 0x01
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LOCK | | | | | | | SYNCBUSY |
| Access | R/W | | | | | | | R |
| Reset | 0 | | | | | | | 0 |

**Bit 7 – LOCK** Lock
Writing this bit to '1' write-protects the WDT.CTRLA register.
It is only possible to write this bit to '1'. This bit can be cleared in Debug mode only.
If the PERIOD bits in WDT.CTRLA are different from zero after boot code, the lock will automatically be set.
This bit is under CCP.

**Bit 0 – SYNCBUSY** Synchronization Busy
This bit is set after writing to the WDT.CTRLA register, while the data is being synchronized from the peripheral clock domain to the WDT clock domain.
This bit is cleared after the synchronization is finished.
This bit is not under CCP.

# 23. TCA - 16-bit Timer/Counter Type A

## 23.1 Features

- 16-Bit Timer/Counter
- Three Compare Channels
- Double-Buffered Timer Period Setting
- Double-Buffered Compare Channels
- Waveform Generation:
  - Frequency generation
  - Single-slope PWM (Pulse-Width Modulation)
  - Dual-slope PWM
- Count on Event
- Timer Overflow Interrupts/Events
- One Compare Match per Compare Channel
- Two 8-Bit Timer/Counters in Split Mode

## 23.2 Overview

The flexible 16-bit PWM Timer/Counter type A (TCA) provides accurate program execution timing, frequency and waveform generation, and command execution.

A TCA consists of a base counter and a set of compare channels. The base counter can be used to count clock cycles or events, or let events control how it counts clock cycles. It has direction control and period setting that can be used for timing. The compare channels can be used together with the base counter to do compare match control, frequency generation, and pulse-width waveform modulation.

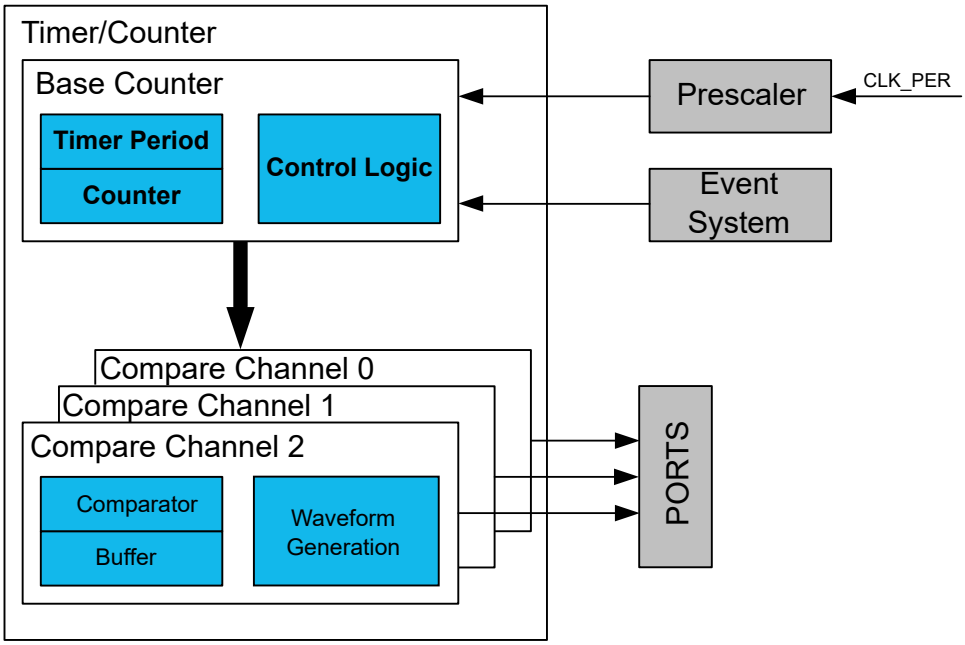
Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each timer/counter clock or event input.

A timer/counter can be clocked and timed from the peripheral clock, with optional prescaling, or from the Event System. The Event System can also be used for direction control or to synchronize operations.

By default, the TCA is a 16-bit timer/counter. The timer/counter has a Split mode feature that splits it into two 8-bit timer/counters with three compare channels each.

A block diagram of the 16-bit timer/counter with closely related peripheral modules (in grey) is shown in the figure below.

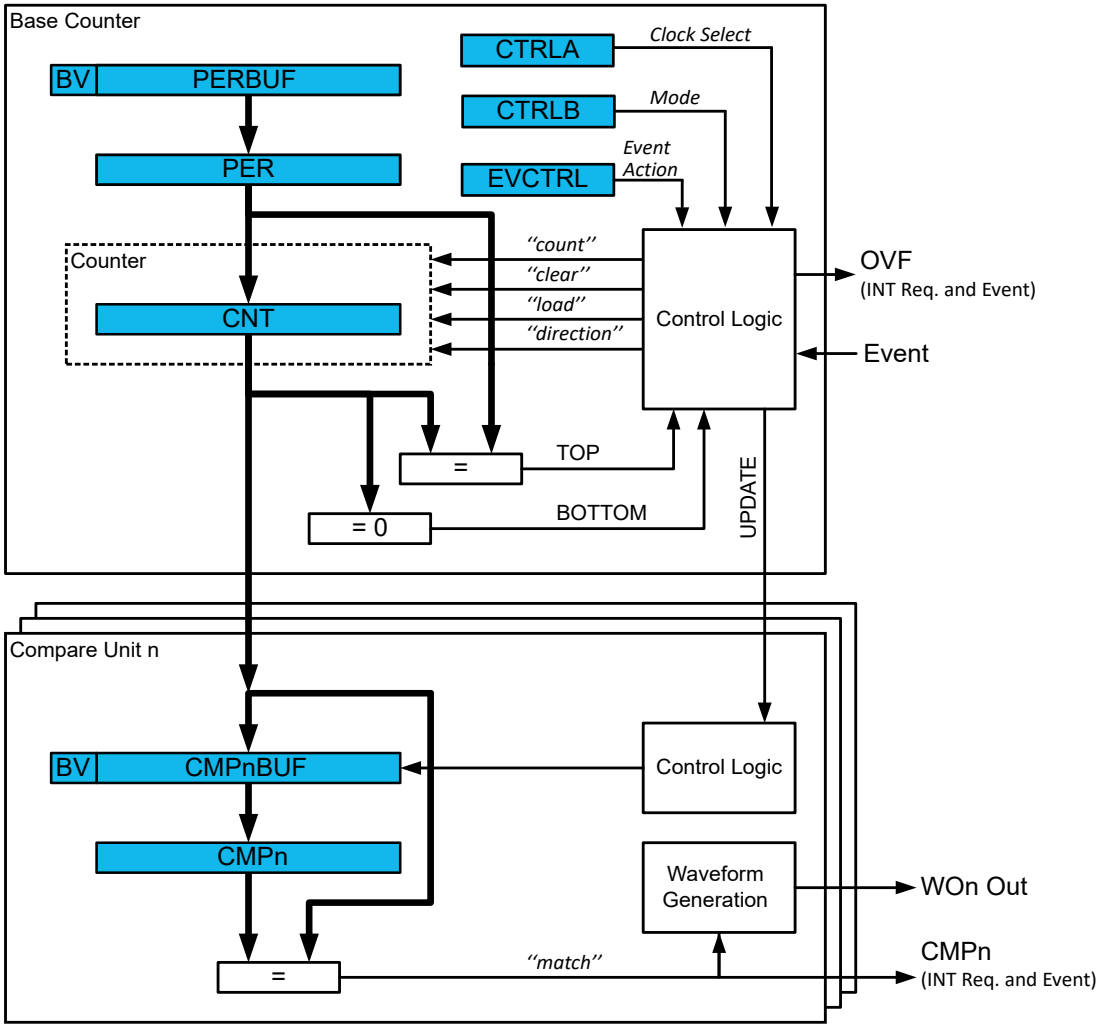
**Figure 23-1. 16-bit Timer/Counter and Closely Related Peripherals**




### 23.2.1 Block Diagram

The figure below shows a detailed block diagram of the timer/counter.

**Figure 23-2. Timer/Counter Block Diagram**



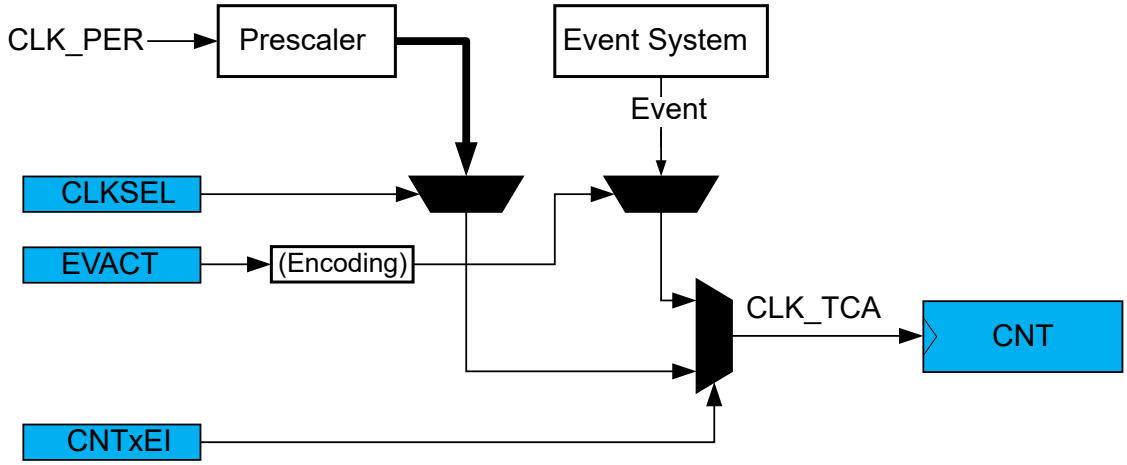

The Counter (TCAn.CNT) register, Period and Compare (TCAn.PER and TCAn.CMPn) registers, and their corresponding buffer registers (TCAn.PERBUF and TCAn.CMPBUFn) are 16-bit registers. All buffer registers have a Buffer Valid (BV) flag that indicates when the buffer contains a new value.

During normal operation, the counter value is continuously compared to zero and the period (PER) value to determine whether the counter has reached TOP or BOTTOM. The counter value can also be compared to the TCAn.CMPn registers.

The timer/counter can generate interrupt requests, events, or change the waveform output after being triggered by the Counter (TCAn.CNT) register reaching TOP, BOTTOM, or CMPn. The interrupt requests, events, or waveform output changes will occur on the next CLK_TCA cycle after the triggering.

CLK_TCA is either the prescaled peripheral clock or events from the Event System, as shown in the figure below.

**Figure 23-3. Timer/Counter Clock Logic**




### 23.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| WOn | Digital output | Waveform output |

## 23.3 Functional Description

### 23.3.1 Definitions

The following definitions are used throughout the documentation:

**Table 23-1. Timer/Counter Definitions**

| Name | Description |
|---|---|
| BOTTOM | The counter reaches BOTTOM when it becomes 0x0000. |
| MAX | The counter reaches MAXimum when it becomes all ones. |
| TOP | The counter reaches TOP when it becomes equal to the highest value in the count sequence. |
| UPDATE | The update condition is met when the timer/counter reaches BOTTOM or TOP, depending on the Waveform Generator mode. Buffered registers with valid buffer values will be updated unless the Lock Update (LUPD) bit in the TCAn.CTRLE register has been set. |
| CNT | Counter register value. |
| CMP | Compare register value. |
| PER | Period register value. |

In general, the term timer is used when the timer/counter is counting periodic clock ticks. The term counter is used when the input signal has sporadic or irregular ticks. The latter can be the case when counting events.

### 23.3.2 Initialization

To start using the timer/counter in a basic mode, follow these steps:

1. Write a TOP value to the Period (TCAn.PER) register.
2. Enable the peripheral by writing a '1' to the Enable (ENABLE) bit in the Control A (TCAn.CTRLA) register. The counter will start counting clock ticks according to the prescaler setting in the Clock Select (CLKSEL) bit field in TCAn.CTRLA.

3. Optional: By writing a '1' to the Enable Counter Event Input A (CNTAEI) bit in the Event Control (TCAn.EVCTRL) register, events are counted instead of clock ticks.
4. The counter value can be read from the Counter (CNT) bit field in the Counter (TCAn.CNT) register.

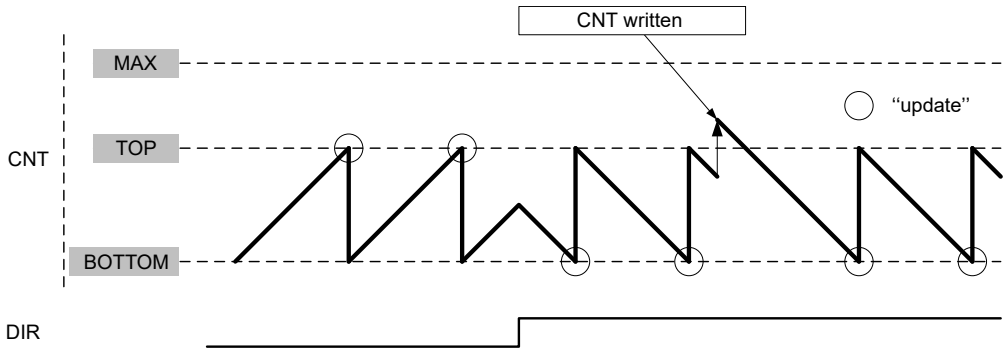
### 23.3.3 Operation

#### 23.3.3.1 Normal Operation

In normal operation, the counter is counting clock ticks in the direction selected by the Direction (DIR) bit in the Control E (TCAn.CTRLE) register, until it reaches TOP or BOTTOM. The clock ticks are given by the peripheral clock (CLK_PER), prescaled according to the Clock Select (CLKSEL) bit field in the Control A (TCAn.CTRLA) register.

When TOP is reached while the counter is counting up, the counter will wrap to '0' at the next clock tick. When counting down, the counter is reloaded with the Period (TCAn.PER) register value when BOTTOM is reached.

**Figure 23-4. Normal Operation**



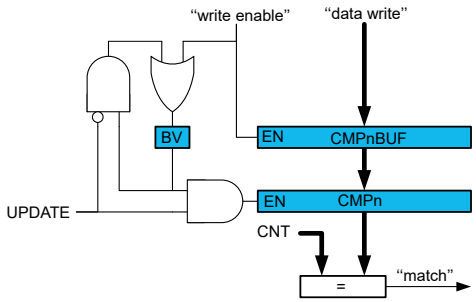

It is possible to change the counter value in the Counter (TCAn.CNT) register when the counter is running. The write access to TCAn.CNT has higher priority than count, clear or reload, and will be immediate. The direction of the counter can also be changed during normal operation by writing to DIR in TCAn.CTRLE.

#### 23.3.3.2 Double Buffering

The Period (TCAn.PER) register value and the Compare n (TCAn.CMPn) register values are all double-buffered (TCAn.PERBUF and TCAn.CMPnBUF).

Each buffer register has a Buffer Valid (BV) flag (PERBV, CMPnBV) in the Control F (TCAn.CTRLF) register, which indicates that the buffer register contains a valid (new) value that can be copied into the corresponding Period or Compare register. When the Period register and Compare n registers are used for a compare operation, the BV flag is set when data are written to the buffer register and cleared on an UPDATE condition. This is shown for a Compare (CMPn) register in the figure below.

**Figure 23-5. Period and Compare Double Buffering**



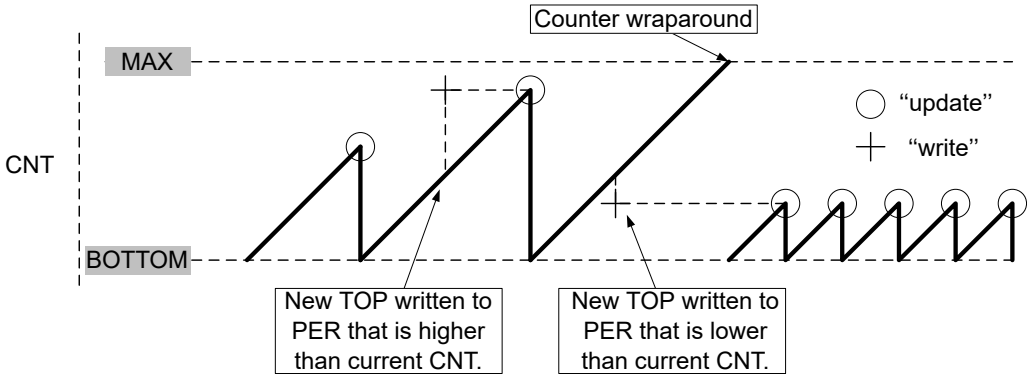

Both the TCAn.CMPn and TCAn.CMPnBUF registers are available as I/O registers. This allows initialization and bypassing of the buffer register and the double-buffering function.

#### 23.3.3.3 Changing the Period

The Counter period is changed by writing a new TOP value to the Period (TCAn.PER) register.

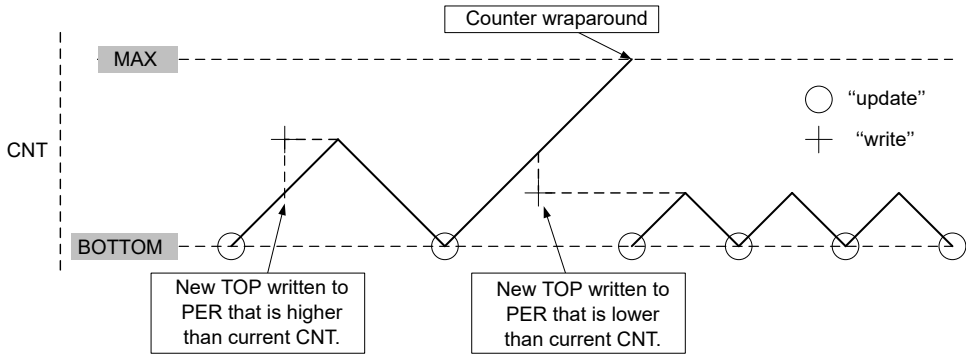
**No Buffering:** If double-buffering is not used, any period update is immediate.

**Figure 23-6. Changing the Period Without Buffering**



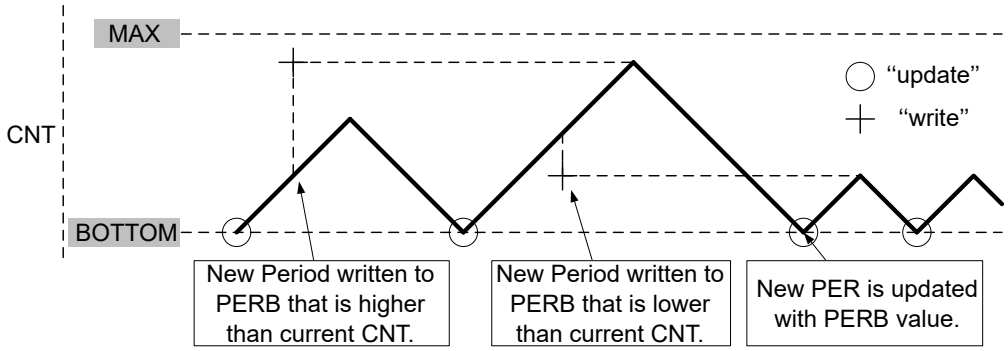

A counter wraparound can occur in any mode of operation when counting up without buffering, as the TCAn.CNT and TCAn.PER registers are continuously compared. If a new TOP value is written to TCAn.PER that is lower than the current TCAn.CNT, the counter will wrap first, before a compare match occurs.

**Figure 23-7. Unbuffered Dual-Slope Operation**




**With Buffering:** When double-buffering is used, the buffer can be written at any time and still maintain correct operation. The TCAn.PER is always updated on the UPDATE condition, as shown for dual-slope operation in the figure below. This prevents wraparound and the generation of odd waveforms.

**Figure 23-8. Changing the Period Using Buffering**




**Note:** Buffering is used in figures illustrating TCA operation if not otherwise specified.

### 23.3.3.4 Compare Channel

Each Compare Channel n continuously compares the counter value (TCAn.CNT) with the Compare n (TCAn.CMPn) register. If TCAn.CNT equals TCAn.CMPn, the Comparator n signals a match. The match will set the Compare Channel's interrupt flag at the next timer clock cycle, and the optional interrupt is generated.

The Compare n Buffer (TCAn.CMPnBUF) register provides double-buffer capability equivalent to that for the period buffer. The double-buffering synchronizes the update of the TCAn.CMPn register with the buffer value to either the TOP or BOTTOM of the counting sequence, according to the UPDATE condition. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses for glitch-free output.

The value in CMPnBUF is moved to CMPn at the UPDATE condition and is compared to the counter value (TCAn.CNT) from the next count.

#### 23.3.3.4.1 Waveform Generation

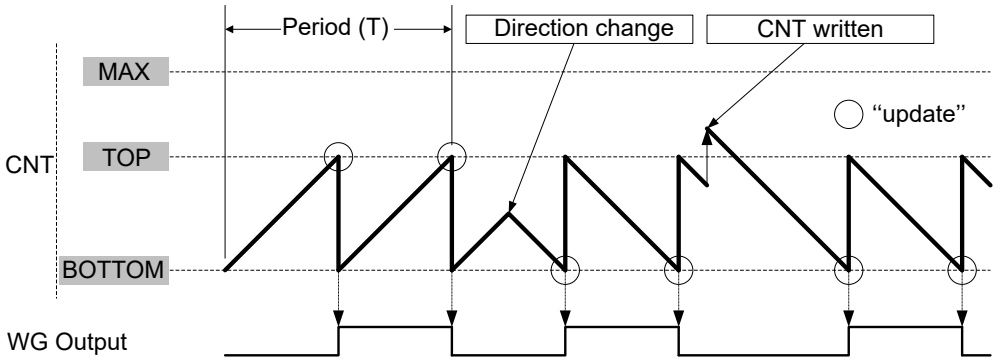
The compare channels can be used for waveform generation on the corresponding port pins. The following requirements must be met to make the waveform visible on the connected port pin:

1. A Waveform Generation mode must be selected by writing the Waveform Generation Mode (WGMODE) bit field in the TCAn.CTRLB register.
2. The compare channels used must be enabled (CMPnEN = `1` in TCAn.CTRLB). This will override the output value for the corresponding pin. An alternative pin can be selected by configuring the Port Multiplexer (PORTMUX). Refer to the *PORTMUX* section for details.
3. The direction for the associated port pin n must be configured in the Port peripheral as an output.
4. Optional: Enable the inverted waveform output for the associated port pin n. Refer to the *PORT* section for details.

#### 23.3.3.4.2 Frequency (FRQ) Waveform Generation

For frequency generation, the period time (T) is controlled by the TCAn.CMP0 register instead of the Period (TCAn.PER) register. The corresponding waveform generator output is toggled on each compare match between the TCAn.CNT and TCAn.CMPn registers.

**Figure 23-9. Frequency Waveform Generation**




The waveform frequency ($f_{FRQ}$) is defined by the following equation:

$$f_{\mathrm{FRQ}} = \frac{\mathrm{f_{CLK_PER}}}{2N(\mathrm{CMPn}+1)}$$

where *N* represents the prescaler divider used (see CLKSEL bit field in the TCAn.CTRLA register), and $f_{CLK_PER}$ is the peripheral clock frequency.

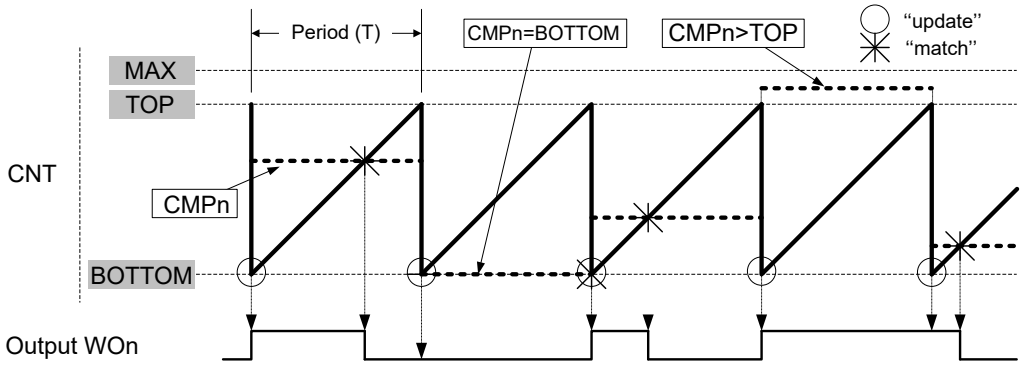
The maximum frequency of the waveform generated is half of the peripheral clock frequency ($f_{CLK_PER}/2$) when TCAn.CMP0 is written to 0x0000 and no prescaling is used (*N* = `1`, CLKSEL = `0x0` in TCAn.CTRLA).

#### 23.3.3.4.3 Single-Slope PWM Generation

For single-slope Pulse-Width Modulation (PWM) generation the period (T) is controlled by the TCAn.PER register, while the values of the TCAn.CMPn registers control the duty cycles of the generated waveforms. The figure below shows how the counter counts from BOTTOM to TOP and then restarts from BOTTOM. The waveform generator output is set at BOTTOM and cleared on the compare match between the TCAn.CNT and TCAn.CMPn registers.

CMPn = BOTTOM will produce a static low signal on WOn while CMPn > TOP will produce a static high signal on WOn.

**Figure 23-10. Single-Slope Pulse-Width Modulation**




**Note:** The representation in the figure above is valid for when CMPn is updated using CMPnBUF.

The TCAn.PER register defines the PWM resolution. The minimum resolution is 2 bits (TCAn.PER = 0x0002), and the maximum resolution is 16 bits (TCAn.PER = MAX-1).

The following equation calculates the exact resolution in bits for single-slope PWM ($R_{PWM_SS}$):

$$R_{\mathrm{PWM_SS}} = \frac{\log(\mathrm{PER}+2)}{\log(2)}$$

The single-slope PWM frequency ($f_{PWM_SS}$) depends on the period setting (TCAn.PER), the peripheral clock frequency $f_{CLK_PER}$ and the TCA prescaler (the CLKSEL bit field in the TCAn.CTRLA register). It is calculated by the following equation where *N* represents the prescaler divider used:

$$f_{\mathrm{PWM_SS}} = \frac{f_{\mathrm{CLK_PER}}}{N(\mathrm{PER}+1)}$$

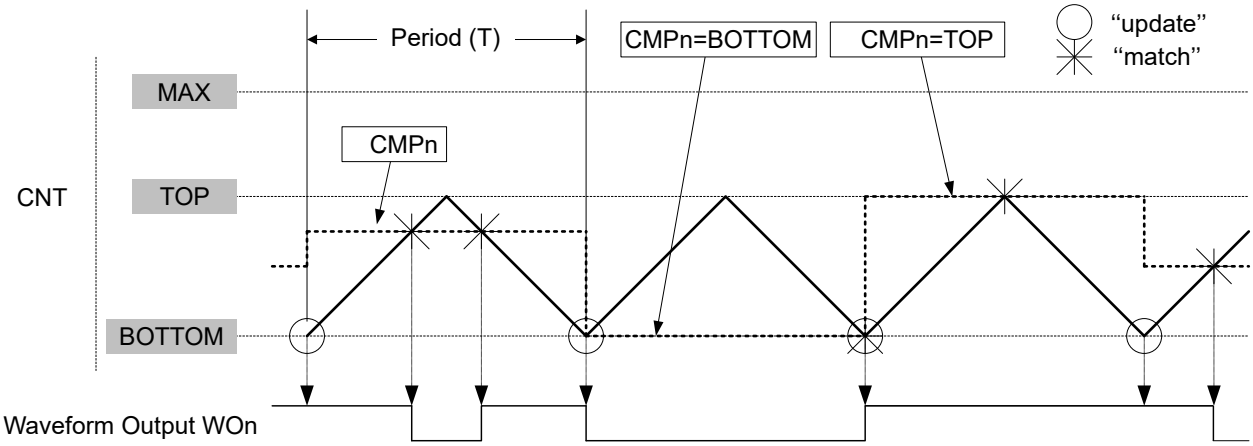
#### 23.3.3.4.4 Dual-Slope PWM

For dual-slope PWM generation, the period (T) is controlled by TCAn.PER, while the values of TCAn.CMPn control the duty cycle of the WG output.

The figure below shows how, for dual-slope PWM, the counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. The waveform generator output is set at BOTTOM, cleared on compare match when up-counting and set on compare match when down-counting.

CMPn = BOTTOM will produce a static low signal on WOn, while CMPn = TOP will produce a static high signal on WOn.

**Figure 23-11. Dual-Slope Pulse-Width Modulation**




**Note:** The representation in the figure above is valid for when CMPn is updated using CMPnBUF.

Using dual-slope PWM results in half the maximum operation frequency compared to single-slope PWM operation, due to twice the number of timer increments per period.

The Period (TCAn.PER) register defines the PWM resolution. The minimum resolution is 2 bits (TCAn.PER = 0x0003), and the maximum resolution is 16 bits (TCAn.PER = MAX).

The following equation calculates the exact resolution in bits for dual-slope PWM ($R_{PWM_DS}$):

$$R_{\mathrm{PWM_DS}} = \frac{\log(\mathrm{PER}+1)}{\log(2)}$$

The PWM frequency depends on the period setting in the TCAn.PER register, the peripheral clock frequency ($f_{CLK_PER}$) and the prescaler divider selected in the CLKSEL bit field in the TCAn.CTRLA register. It is calculated by the following equation:

$$f_{\mathrm{PWM_DS}} = \frac{f_{\mathrm{CLK_PER}}}{2N \cdot \mathrm{PER}}$$

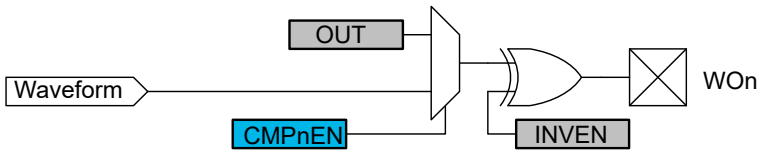
*N* represents the prescaler divider used.

#### 23.3.3.4.5 Port Override for Waveform Generation

To make the waveform generation available on the port pins, the corresponding port pin direction must be set as output (PORTx.DIR[n] = `1`). The TCA will override the port pin values when the compare channel is enabled (CMPnEN = `1` in TCAn.CTRLB) and a Waveform Generation mode is selected.

The figure below shows the port override for TCA. The timer/counter compare channel will override the port pin output value (OUT) on the corresponding port pin. Enabling inverted I/O on the port pin (INVEN = `1` in PORT.PINn) inverts the corresponding WG output.

**Figure 23-12. Port Override for Timer/Counter Type A**




### 23.3.3.5 Timer/Counter Commands

A set of commands can be issued by software to immediately change the state of the peripheral. These commands give direct control of the UPDATE, RESTART and RESET signals. A command is issued by writing the respective value to the Command (CMD) bit field in the Control E (TCAn.CTRLESET) register.

An UPDATE command has the same effect as when an UPDATE condition occurs, except that the UPDATE command is not affected by the state of the Lock Update (LUPD) bit in the Control E (TCAn.CTRLE) register.

The software can force a restart of the current waveform period by issuing a RESTART command. In this case, the counter, direction, and all compare outputs are set to '`0`'.

A RESET command will set all timer/counter registers to their initial values. A RESET command can be issued only when the timer/counter is not running (ENABLE = `0` in the TCAn.CTRLA register).

### 23.3.3.6 Split Mode - Two 8-Bit Timer/Counters

#### Split Mode Overview

To double the number of timers and PWM channels in the TCA, a Split mode is provided. In this Split mode, the 16-bit timer/counter acts as two separate 8-bit timers, which each have three compare channels for PWM generation. The Split mode will only work with single-slope down-count. Event controlled operation is not supported in Split mode.

Activating Split mode results in changes to the functionality of some registers and register bits. The modifications are described in a separate register map (see 23.6 Register Summary - Split Mode).

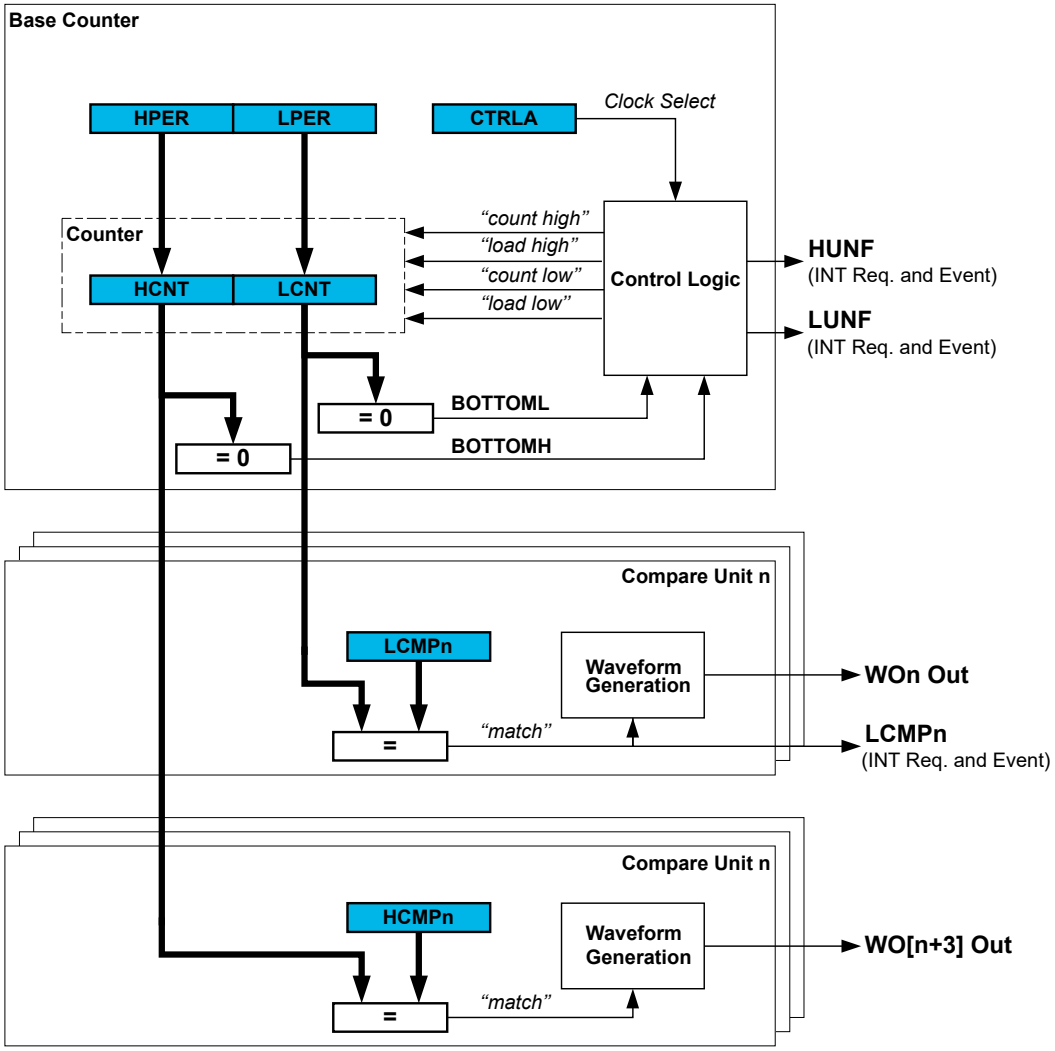
#### Split Mode Differences Compared to Normal Mode

- Count:
  - Down-count only
  - Low Byte Timer Counter (TCAn.LCNT) register and High Byte Timer Counter (TCAn.HCNT) register are independent

- Waveform Generation:
  - Single-slope PWM only (WGMODE = SINGLESLOPE in TCAn.CTRLB)
- Interrupt:
  - No change for Low Byte Timer Counter (TCAn.LCNT) register
  - Underflow interrupt for High Byte Timer Counter (TCAn.HCNT) register
  - No compare interrupt or flag for High Byte Compare n (TCAn.HCMPn) register
- Event Actions: Not Compatible
- Buffer Registers and Buffer Valid Flags: Unused
- Register Access: Byte Access to All Registers

**Block Diagram**

**Figure 23-13. Timer/Counter Block Diagram Split Mode**




**Split Mode Initialization**

When shifting between Normal mode and Split mode, the functionality of some registers and bits changes, but their values do not. For this reason, disabling the peripheral (ENABLE = 0 in TCAn.CTRLA) and doing a hard Reset (CMD = RESET in TCAn.CTRLESET) is recommended when changing the mode to avoid unexpected behavior.

To start using the timer/counter in basic Split mode after a hard Reset, follow these steps:

1. Enable Split mode by writing a '1' to the Split mode enable (SPLITM) bit in the Control D (TCAn.CTRLD) register.
2. Write a TOP value to the Period (TCAn.PER) registers.
3. Enable the peripheral by writing a '1' to the Enable (ENABLE) bit in the Control A (TCAn.CTRLA) register. The counter will start counting clock ticks according to the prescaler setting in the Clock Select (CLKSEL) bit field in the TCAn.CTRLA register.
4. The counter values can be read from the Counter bit field in the Counter (TCAn.CNT) registers.

### 23.3.4 Events

The TCA can generate the events described in the table below. All event generators except TCAn_HUNF are shared between Normal mode and Split mode operation, and the generator name indicates what specific signal the generator represents in each mode in the following way: OVF_LUNF corresponds to overflow in Normal mode and Low byte timer underflow in Split mode. The same applies to CMPn_LCMPn.

**Table 23-2. Event Generators in TCA**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| TCAn | OVF_LUNF | Normal mode: Overflow<br>Split mode: Low byte timer underflow | Pulse | CLK_PER | One CLK_PER period |
| | HUNF | Normal mode: Not available<br>Split mode: High byte timer underflow | Pulse | CLK_PER | One CLK_PER period |
| | CMP0_LCMP0 | Normal mode: Compare Channel 0 match<br>Split mode: Low byte timer Compare Channel 0 match | Pulse | CLK_PER | One CLK_PER period |
| | CMP1_LCMP1 | Normal mode: Compare Channel 1 match<br>Split mode: Low byte timer Compare Channel 1 match | Pulse | CLK_PER | One CLK_PER period |
| | CMP2_LCMP2 | Normal mode: Compare Channel 2 match<br>Split mode: Low byte timer Compare Channel 2 match | Pulse | CLK_PER | One CLK_PER period |

The conditions for generating an event are identical to those that will raise the corresponding interrupt flag in the TCAn.INTFLAGS register for both Normal mode and Split mode.

The TCA has two event users for detecting and acting upon input events. The table below describes the event users and their associated functionality.

**Table 23-3. Event Users in TCA**

| User Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| TCAn | CNTA | Count on a positive event edge | Edge | Sync |
| | | Count on any event edge | Edge | Sync |
| | | Count while the event signal is high | Level | Sync |
| | | The event level controls the count direction, up when low and down when high | Level | Sync |
| | CNTB | The event level controls count direction, up when low and down when high | Level | Sync |
| | | Restart counter on a positive event edge | Edge | Sync |
| | | Restart counter on any event edge | Edge | Sync |
| | | Restart counter while the event signal is high | Level | Sync |

The specific actions described in the table above are selected by writing to the Event Action (EVACTA, EVACTB) bits in the Event Control (TCAn.EVCTRL) register. Input events are enabled by writing a '`1`' to the Enable Counter Event Input (CNTAEI and CNTBEI) bits in the TCAn.EVCTRL register.

If both EVACTA and EVACTB are configured to control the count direction, the event signals will be `OR`'ed to determine the count direction. Both event inputs must then be low for the counter to count upwards.

Event inputs are not used in Split mode.

Refer to the *Event System (EVSYS)* section for more details regarding event types and Event System configuration.

### 23.3.5 Interrupts

**Table 23-4. Available Interrupt Vectors and Sources in Normal Mode**

| Name | Vector Description | Conditions |
|---|---|---|
| OVF | Overflow or underflow interrupt | The counter has reached TOP or BOTTOM |
| CMP0 | Compare Channel 0 interrupt | Match between the counter value and the Compare 0 register |
| CMP1 | Compare Channel 1 interrupt | Match between the counter value and the Compare 1 register |
| CMP2 | Compare Channel 2 interrupt | Match between the counter value and the Compare 2 register |

**Table 23-5. Available Interrupt Vectors and Sources in Split Mode**

| Name | Vector Description | Conditions |
|---|---|---|
| LUNF | Low-byte Underflow interrupt | Low byte timer reaches BOTTOM |
| HUNF | High-byte Underflow interrupt | High byte timer reaches BOTTOM |
| LCMP0 | Compare Channel 0 interrupt | Match between the counter value and the low byte of the Compare 0 register |
| LCMP1 | Compare Channel 1 interrupt | Match between the counter value and the low byte of the Compare 1 register |
| LCMP2 | Compare Channel 2 interrupt | Match between the counter value and the low byte of the Compare 2 register |

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

### 23.3.6 Sleep Mode Operation

TCA is by default disabled in Standby Sleep mode. It will be halted as soon as the Sleep mode is entered.

The module can stay fully operational in Standby Sleep mode if the Run Standby (RUNSTDBY) bit in the TCAn.CTRLA register is written to '`1`'.

All operation is halted in Power-Down Sleep mode.

## 23.4 Register Summary - Normal Mode

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | RUNSTDBY | | | | CLKSEL[2:0] | | | ENABLE |
| 0x01 | CTRLB | 7:0 | | CMP2EN | CMP1EN | CMP0EN | ALUPD | WGMODE[2:0] | | |
| 0x02 | CTRLC | 7:0 | | | | | | CMP2OV | CMP1OV | CMP0OV |
| 0x03 | CTRLD | 7:0 | | | | | | | | SPLITM |
| 0x04 | CTRLECLR | 7:0 | | | | | CMD[1:0] | | LUPD | DIR |
| 0x05 | CTRLESET | 7:0 | | | | | CMD[1:0] | | LUPD | DIR |
| 0x06 | CTRLFCLR | 7:0 | | | | | CMP2BV | CMP1BV | CMP0BV | PERBV |
| 0x07 | CTRLFSET | 7:0 | | | | | CMP2BV | CMP1BV | CMP0BV | PERBV |
| 0x08 | Reserved | | | | | | | | | |
| 0x09 | EVCTRL | 7:0 | EVACTB[2:0] | | | CNTBEI | EVACTA[2:0] | | | CNTAEI |
| 0x0A | INTCTRL | 7:0 | | CMP2 | CMP1 | CMP0 | | | | OVF |
| 0x0B | INTFLAGS | 7:0 | | CMP2 | CMP1 | CMP0 | | | | OVF |
| 0x0C ... 0x0D | Reserved | | | | | | | | | |
| 0x0E | DBGCTRL | 7:0 | | | | | | | | DBGRUN |
| 0x0F | TEMP | 7:0 | TEMP[7:0] | | | | | | | |
| 0x10 ... 0x1F | Reserved | | | | | | | | | |
| 0x20 | CNT | 7:0 | CNT[7:0] | | | | | | | |
| | | 15:8 | CNT[15:8] | | | | | | | |
| 0x22 ... 0x25 | Reserved | | | | | | | | | |
| 0x26 | PER | 7:0 | PER[7:0] | | | | | | | |
| | | 15:8 | PER[15:8] | | | | | | | |
| 0x28 | CMP0 | 7:0 | CMP[7:0] | | | | | | | |
| | | 15:8 | CMP[15:8] | | | | | | | |
| 0x2A | CMP1 | 7:0 | CMP[7:0] | | | | | | | |
| | | 15:8 | CMP[15:8] | | | | | | | |
| 0x2C | CMP2 | 7:0 | CMP[7:0] | | | | | | | |
| | | 15:8 | CMP[15:8] | | | | | | | |
| 0x2E ... 0x35 | Reserved | | | | | | | | | |
| 0x36 | PERBUF | 7:0 | PERBUF[7:0] | | | | | | | |
| | | 15:8 | PERBUF[15:8] | | | | | | | |
| 0x38 | CMP0BUF | 7:0 | CMPBUF[7:0] | | | | | | | |
| | | 15:8 | CMPBUF[15:8] | | | | | | | |
| 0x3A | CMP1BUF | 7:0 | CMPBUF[7:0] | | | | | | | |
| | | 15:8 | CMPBUF[15:8] | | | | | | | |
| 0x3C | CMP2BUF | 7:0 | CMPBUF[7:0] | | | | | | | |
| | | 15:8 | CMPBUF[15:8] | | | | | | | |

## 23.5 Register Description - Normal Mode

### 23.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | | | | CLKSEL[2:0] | | | ENABLE |
| Access | R/W | | | | R/W | R/W | R/W | R/W |
| Reset | 0 | | | | 0 | 0 | 0 | 0 |

**Bit 7 – RUNSTDBY** Run Standby
Writing a '1' to this bit will enable the peripheral to run in Standby Sleep mode.

**Bits 3:1 – CLKSEL[2:0]** Clock Select
These bits select the clock frequency for the timer/counter.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV1 | $f_{TCA} = f_{CLK_PER}$ |
| 0x1 | DIV2 | $f_{TCA} = f_{CLK_PER}/2$ |
| 0x2 | DIV4 | $f_{TCA} = f_{CLK_PER}/4$ |
| 0x3 | DIV8 | $f_{TCA} = f_{CLK_PER}/8$ |
| 0x4 | DIV16 | $f_{TCA} = f_{CLK_PER}/16$ |
| 0x5 | DIV64 | $f_{TCA} = f_{CLK_PER}/64$ |
| 0x6 | DIV256 | $f_{TCA} = f_{CLK_PER}/256$ |
| 0x7 | DIV1024 | $f_{TCA} = f_{CLK_PER}/1024$ |

**Bit 0 – ENABLE** Enable

| Value | Description |
|---|---|
| 0 | The peripheral is disabled |
| 1 | The peripheral is enabled |

### 23.5.2 Control B - Normal Mode

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | CMP2EN | CMP1EN | CMP0EN | ALUPD | WGMODE[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 4, 5, 6 – CMPEN** Compare n Enable
In the FRQ and PWM Waveform Generation modes the Compare n Enable (CMPnEN) bits will make the waveform output available on the pin corresponding to WOn, overriding the value in the corresponding PORT output register. The corresponding pin direction must be configured as an output in the PORT peripheral.

| Value | Description |
|---|---|
| 0 | Waveform output WOn will not be available on the corresponding pin |
| 1 | Waveform output WOn will override the output value of the corresponding pin |

**Bit 3 – ALUPD** Auto-Lock Update
The Auto-Lock Update bit controls the Lock Update (LUPD) bit in the TCAn.CTRLE register. When ALUPD is written to '1', LUPD will be set to '1' until the Buffer Valid (CMPnBV) bits of all enabled compare channels are '1'. This condition will clear LUPD.
It will remain cleared until the next UPDATE condition, where the buffer values will be transferred to the CMPn registers and LUPD will be set to '1' again. This makes sure that the CMPnBUF register values are not transferred to the CMPn registers until all enabled compare buffers are written.

| Value | Description |
|---|---|
| 0 | LUPD in TCA.CTRLE is not altered by the system |
| 1 | LUPD in TCA.CTRLE is set and cleared automatically |

**Bits 2:0 – WGMODE[2:0]** Waveform Generation Mode
These bits select the Waveform Generation mode and control the counting sequence of the counter, TOP value, UPDATE condition, Interrupt condition, and the type of waveform generated.
No waveform generation is performed in the Normal mode of operation. For all other modes, the waveform generator output will only be directed to the port pins if the corresponding CMPnEN bit has been set. The port pin direction must be set as output.

**Table 23-6. Timer Waveform Generation Mode**

| Value | Group Configuration | Mode of Operation | TOP | UPDATE | OVF |
|---|---|---|---|---|---|
| 0x0 | NORMAL | Normal | PER | TOP(1) | TOP(1) |
| 0x1 | FRQ | Frequency | CMP0 | TOP(1) | TOP(1) |
| 0x2 | - | Reserved | - | - | - |
| 0x3 | SINGLESLOPE | Single-slope PWM | PER | BOTTOM | BOTTOM |
| 0x4 | - | Reserved | - | - | - |
| 0x5 | DSTOP | Dual-slope PWM | PER | BOTTOM | TOP |
| 0x6 | DSBOTH | Dual-slope PWM | PER | BOTTOM | TOP and BOTTOM |
| 0x7 | DSBOTTOM | Dual-slope PWM | PER | BOTTOM | BOTTOM |

**Note:**
1. When counting up.

### 23.5.3 Control C - Normal Mode

**Name:** CTRLC
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | CMP2OV | CMP1OV | CMP0OV |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bit 2 – CMP2OV** Compare Output Value 2
See CMP0OV.

**Bit 1 – CMP1OV** Compare Output Value 1
See CMP0OV.

**Bit 0 – CMP0OV** Compare Output Value 0
The CMPnOV bits allow direct access to the waveform generator's output compare value when the timer/counter is not enabled. This is used to set or clear the WG output value when the timer/counter is not running.

### 23.5.4 Control D

**Name:** CTRLD
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

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

**Bit 0 – SPLITM** Enable Split Mode
This bit sets the timer/counter in Split mode operation. It will then work as two 8-bit timer/counters. The register map will change compared to normal 16-bit mode.

### 23.5.5 Control Register E Clear - Normal Mode

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

This register can be used instead of a Read-Modify-Write (RMW) to clear individual bits by writing a '1' to its bit location.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMD[1:0] | | LUPD | DIR |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:2 – CMD[1:0]** Command
These bits are used for software control of update, restart and Reset of the timer/counter. The command bits are always read as '0'.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No command |
| 0x1 | UPDATE | Force update |
| 0x2 | RESTART | Force restart |
| 0x3 | RESET | Force hard Reset (ignored if the timer/counter is enabled) |

**Bit 1 – LUPD** Lock Update
Lock update can be used to ensure that all buffers are valid before an update is performed.

| Value | Description |
|---|---|
| 0 | The buffered registers are updated as soon as an UPDATE condition has occurred |
| 1 | No update of the buffered registers is performed, even though an UPDATE condition has occurred |

**Bit 0 – DIR** Counter Direction
Normally this bit is controlled in hardware by the Waveform Generation mode or by event actions, but it can also be changed from software.

| Value | Description |
|---|---|
| 0 | The counter is counting up (incrementing) |
| 1 | The counter is counting down (decrementing) |

### 23.5.6 Control Register E Set - Normal Mode

**Name:** CTRLESET
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

This register can be used instead of a Read-Modify-Write (RMW) to set individual bits by writing a '1' to its bit location.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMD[1:0] | | LUPD | DIR |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:2 – CMD[1:0]** Command
These bits are used for software control of update, restart and Reset the timer/counter. The command bits are always read as '0'.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No command |
| 0x1 | UPDATE | Force update |
| 0x2 | RESTART | Force restart |
| 0x3 | RESET | Force hard Reset (ignored if the timer/counter is enabled) |

**Bit 1 – LUPD** Lock Update
Locking the update ensures that all buffers are valid before an update is performed.

| Value | Description |
|---|---|
| 0 | The buffered registers are updated as soon as an UPDATE condition has occurred |
| 1 | No update of the buffered registers is performed, even though an UPDATE condition has occurred |

**Bit 0 – DIR** Counter Direction
Normally this bit is controlled in hardware by the Waveform Generation mode or by event actions, but it can also be changed from software.

| Value | Description |
|---|---|
| 0 | The counter is counting up (incrementing) |
| 1 | The counter is counting down (decrementing) |

### 23.5.7 Control Register F Clear

**Name:** CTRLFCLR
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

This register can be used instead of a Read-Modify-Write (RMW) to clear individual bits by writing a '1' to its bit location.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMP2BV | CMP1BV | CMP0BV | PERBV |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bit 3 – CMP2BV** Compare 2 Buffer Valid
See CMP0BV.

**Bit 2 – CMP1BV** Compare 1 Buffer Valid
See CMP0BV.

**Bit 1 – CMP0BV** Compare 0 Buffer Valid
The CMPnBV bits are set when a new value is written to the corresponding TCAn.CMPnBUF register. These bits are automatically cleared on an UPDATE condition.

**Bit 0 – PERBV** Period Buffer Valid
This bit is set when a new value is written to the TCAn.PERBUF register. This bit is automatically cleared on an UPDATE condition.

### 23.5.8 Control Register F Set

**Name:** CTRLFSET
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

This register can be used instead of a Read-Modify-Write (RMW) to set individual bits by writing a '1' to its bit location.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMP2BV | CMP1BV | CMP0BV | PERBV |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bit 3 – CMP2BV** Compare 2 Buffer Valid
See CMP0BV.

**Bit 2 – CMP1BV** Compare 1 Buffer Valid
See CMP0BV.

**Bit 1 – CMP0BV** Compare 0 Buffer Valid
The CMPnBV bits are set when a new value is written to the corresponding TCAn.CMPnBUF register. These bits are automatically cleared on an UPDATE condition.

**Bit 0 – PERBV** Period Buffer Valid
This bit is set when a new value is written to the TCAn.PERBUF register. This bit is automatically cleared on an UPDATE condition.

### 23.5.9 Event Control

**Name:** EVCTRL
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EVACTB[2:0] | | | CNTBEI | EVACTA[2:0] | | | CNTAEI |
| Access | R/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 – EVACTB[2:0]** Event Action B
These bits define what action the counter will take upon certain event conditions.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No action |
| 0x1 | - | Reserved |
| 0x2 | - | Reserved |
| 0x3 | UPDOWN | Count prescaled clock cycles or count events according to setting for event input A. The event signal controls the count direction, up when low and down when high. |
| 0x4 | RESTART_POSEDGE | Restart counter on positive event edge |
| 0x5 | RESTART_ANYEDGE | Restart counter on any event edge |
| 0x6 | RESTART_HIGHLVL | Restart counter while the event signal is high |
| Other | - | Reserved |

**Bit 4 – CNTBEI** Enable Counter Event Input B

| Value | Description |
|---|---|
| 0 | Counter Event input B is disabled |
| 1 | Counter Event input B is enabled according to EVACTB bit field |

**Bits 3:1 – EVACTA[2:0]** Event Action A
These bits define what action the counter will take upon certain event conditions.

| Value | Name | Description |
|---|---|---|
| 0x0 | CNT_POSEDGE | Count on positive event edge |
| 0x1 | CNT_ANYEDGE | Count on any event edge |
| 0x2 | CNT_HIGHLVL | Count prescaled clock cycles while the event signal is high |
| 0x3 | UPDOWN | Count prescaled clock cycles. The event signal controls the count direction, up when low and down when high. |
| Other | | Reserved |

**Bit 0 – CNTAEI** Enable Counter Event Input A

| Value | Description |
|---|---|
| 0 | Counter Event input A is disabled |
| 1 | Counter Event input A is enabled according to EVACTA bit field |

### 23.5.10 Interrupt Control Register - Normal Mode

**Name:** INTCTRL
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | CMP2 | CMP1 | CMP0 | | | | OVF |
| Access | | R/W | R/W | R/W | | | | R/W |
| Reset | | 0 | 0 | 0 | | | | 0 |

**Bit 6 – CMP2** Compare Channel 2 Interrupt Enable
See CMP0.

**Bit 5 – CMP1** Compare Channel 1 Interrupt Enable
See CMP0.

**Bit 4 – CMP0** Compare Channel 0 Interrupt Enable
Writing the CMPn bit to '1' enables the interrupt from Compare Channel n.

**Bit 0 – OVF** Timer Overflow/Underflow Interrupt Enable
Writing the OVF bit to '1' enables the overflow/underflow interrupt.

### 23.5.11 Interrupt Flag Register - Normal Mode

**Name:** INTFLAGS
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | CMP2 | CMP1 | CMP0 | | | | OVF |
| Access | | R/W | R/W | R/W | | | | R/W |
| Reset | | 0 | 0 | 0 | | | | 0 |

**Bit 6 – CMP2** Compare Channel 2 Interrupt Flag
See the CMP0 flag description.

**Bit 5 – CMP1** Compare Channel 1 Interrupt Flag
See the CMP0 flag description.

**Bit 4 – CMP0** Compare Channel 0 Interrupt Flag
The Compare Interrupt (CMPn) flag is set on a compare match on the corresponding compare channel.
For all modes of operation, the CMPn flag will be set when a compare match occurs between the Count (CNT) register and the corresponding Compare n (CMPn) register. The CMPn flag is not cleared automatically. It will be cleared only by writing a '1' to its bit location.

**Bit 0 – OVF** Overflow/Underflow Interrupt Flag
This flag is set either on a TOP (overflow) or BOTTOM (underflow) condition, depending on the WGMODE setting.
The OVF flag is not cleared automatically. It will be cleared only by writing a '1' to its bit location.

### 23.5.12 Debug Control Register

**Name:** DBGCTRL
**Offset:** 0x0E
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Run in Debug

| Value | Description |
|---|---|
| 0 | The peripheral is halted in Break Debug mode and ignores events. |
| 1 | The peripheral will continue to run in Break Debug mode when the CPU is halted. |

### 23.5.13 Temporary Bits for 16-Bit Access

**Name:** TEMP
**Offset:** 0x0F
**Reset:** 0x00
**Property:** -

The Temporary register is used by the CPU for 16-bit single-cycle access to the 16-bit registers of this peripheral. The register is common for all the 16-bit registers of this peripheral and can be read and written by software. For more details on reading and writing 16-bit registers, refer to *Accessing 16-Bit Registers* in the *Memories* section. There is one common Temporary register for all the 16-bit registers of this peripheral.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TEMP[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 – TEMP[7:0]** Temporary Bits for 16-bit Access

### 23.5.14 Counter Register - Normal Mode

**Name:** CNT
**Offset:** 0x20
**Reset:** 0x00
**Property:** -

The TCAn.CNTL and TCAn.CNTH register pair represents the 16-bit value, TCAn.CNT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

CPU and UPDI write access has priority over internal updates of the register.

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

**Bits 15:8 – CNT[15:8]** Counter High Byte
These bits hold the MSB of the 16-bit Counter register.

**Bits 7:0 – CNT[7:0]** Counter Low Byte
These bits hold the LSB of the 16-bit Counter register.

### 23.5.15 Period Register - Normal Mode

**Name:** PER
**Offset:** 0x26
**Reset:** 0xFFFF
**Property:** -

TCAn.PER contains the 16-bit TOP value in the timer/counter in all modes of operation, except Frequency Waveform Generation (FRQ).

The TCAn.PERL and TCAn.PERH register pair represents the 16-bit value, TCAn.PER. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

| 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

**Bits 15:8 – PER[15:8]** Periodic High Byte
These bits hold the MSB of the 16-bit Period register.

**Bits 7:0 – PER[7:0]** Periodic Low Byte
These bits hold the LSB of the 16-bit Period register.

### 23.5.16 Compare n Register - Normal Mode

**Name:** CMPn
**Offset:** 0x28 + n*0x02 [n=0..2]
**Reset:** 0x00
**Property:** -

This register is continuously compared to the counter value. Normally, the outputs from the comparators are used to generate waveforms.

TCAn.CMPn registers are updated with the buffer value from their corresponding TCAn.CMPnBUF register when an UPDATE condition occurs.

The TCAn.CMPnL and TCAn.CMPnH register pair represents the 16-bit value, TCAn.CMPn. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

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

**Bits 15:8 – CMP[15:8]** Compare High Byte
These bits hold the MSB of the 16-bit Compare register.

**Bits 7:0 – CMP[7:0]** Compare Low Byte
These bits hold the LSB of the 16-bit Compare register.

### 23.5.17 Period Buffer Register

**Name:** PERBUF
**Offset:** 0x36
**Reset:** 0xFFFF
**Property:** -

This register serves as the buffer for the Period (TCAn.PER) register. Writing to this register from the CPU or UPDI will set the Period Buffer Valid (PERBV) bit in the TCAn.CTRLF register.

The TCAn.PERBUFL and TCAn.PERBUFH register pair represents the 16-bit value, TCAn.PERBUF. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

| 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 15:8 – PERBUF[15:8]** Period Buffer High Byte
These bits hold the MSB of the 16-bit Period Buffer register.

**Bits 7:0 – PERBUF[7:0]** Period Buffer Low Byte
These bits hold the LSB of the 16-bit Period Buffer register.

### 23.5.18 Compare n Buffer Register

**Name:** CMPnBUF
**Offset:** 0x38 + n*0x02 [n=0..2]
**Reset:** 0x00
**Property:** -

This register serves as the buffer for the associated Compare n (TCAn.CMPn) register. Writing to this register from the CPU or UPDI will set the Compare Buffer valid (CMPnBV) bit in the TCAn.CTRLF register.

The TCAn.CMPnBUFL and TCAn.CMPnBUFH register pair represents the 16-bit value, TCAn.CMPnBUF. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

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

**Bits 15:8 – CMPBUF[15:8]** Compare High Byte
These bits hold the MSB of the 16-bit Compare Buffer register.

**Bits 7:0 – CMPBUF[7:0]** Compare Low Byte
These bits hold the LSB of the 16-bit Compare Buffer register.

## 23.6 Register Summary - Split Mode

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | RUNSTDBY | | | | CLKSEL[2:0] | | | ENABLE |
| 0x01 | CTRLB | 7:0 | | HCMP2EN | HCMP1EN | HCMP0EN | | LCMP2EN | LCMP1EN | LCMP0EN |
| 0x02 | CTRLC | 7:0 | | HCMP2OV | HCMP1OV | HCMP0OV | | LCMP2OV | LCMP1OV | LCMP0OV |
| 0x03 | CTRLD | 7:0 | | | | | | | | SPLITM |
| 0x04 | CTRLECLR | 7:0 | | | | | CMD[1:0] | | CMDEN[1:0] | |
| 0x05 | CTRLESET | 7:0 | | | | | CMD[1:0] | | CMDEN[1:0] | |
| 0x06 ... 0x09 | Reserved | | | | | | | | | |
| 0x0A | INTCTRL | 7:0 | | LCMP2 | LCMP1 | LCMP0 | | | HUNF | LUNF |
| 0x0B | INTFLAGS | 7:0 | | LCMP2 | LCMP1 | LCMP0 | | | HUNF | LUNF |
| 0x0C ... 0x0D | Reserved | | | | | | | | | |
| 0x0E | DBGCTRL | 7:0 | | | | | | | | DBGRUN |
| 0x0F ... 0x1F | Reserved | | | | | | | | | |
| 0x20 | LCNT | 7:0 | LCNT[7:0] | | | | | | | |
| 0x21 | HCNT | 7:0 | HCNT[7:0] | | | | | | | |
| 0x22 ... 0x25 | Reserved | | | | | | | | | |
| 0x26 | LPER | 7:0 | LPER[7:0] | | | | | | | |
| 0x27 | HPER | 7:0 | HPER[7:0] | | | | | | | |
| 0x28 | LCMP0 | 7:0 | LCMP[7:0] | | | | | | | |
| 0x29 | HCMP0 | 7:0 | HCMP[7:0] | | | | | | | |
| 0x2A | LCMP1 | 7:0 | LCMP[7:0] | | | | | | | |
| 0x2B | HCMP1 | 7:0 | HCMP[7:0] | | | | | | | |
| 0x2C | LCMP2 | 7:0 | LCMP[7:0] | | | | | | | |
| 0x2D | HCMP2 | 7:0 | HCMP[7:0] | | | | | | | |

## 23.7 Register Description - Split Mode

### 23.7.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | | | | CLKSEL[2:0] | | | ENABLE |
| Access | R/W | | | | R/W | R/W | R/W | R/W |
| Reset | 0 | | | | 0 | 0 | 0 | 0 |

**Bit 7 – RUNSTDBY** Run Standby
Writing a '1' to this bit will enable the peripheral to run in Standby Sleep mode.

**Bits 3:1 – CLKSEL[2:0]** Clock Select
These bits select the clock frequency for the timer/counter.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV1 | $f_{TCA} = f_{CLK_PER}$ |
| 0x1 | DIV2 | $f_{TCA} = f_{CLK_PER}/2$ |
| 0x2 | DIV4 | $f_{TCA} = f_{CLK_PER}/4$ |
| 0x3 | DIV8 | $f_{TCA} = f_{CLK_PER}/8$ |
| 0x4 | DIV16 | $f_{TCA} = f_{CLK_PER}/16$ |
| 0x5 | DIV64 | $f_{TCA} = f_{CLK_PER}/64$ |
| 0x6 | DIV256 | $f_{TCA} = f_{CLK_PER}/256$ |
| 0x7 | DIV1024 | $f_{TCA} = f_{CLK_PER}/1024$ |

**Bit 0 – ENABLE** Enable

| Value | Description |
|---|---|
| 0 | The peripheral is disabled |
| 1 | The peripheral is enabled |

### 23.7.2 Control B - Split Mode

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | HCMP2EN | HCMP1EN | HCMP0EN | | LCMP2EN | LCMP1EN | LCMP0EN |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 6 – HCMP2EN** High byte Compare 2 Enable
See HCMP0EN.

**Bit 5 – HCMP1EN** High byte Compare 1 Enable
See HCMP0EN.

**Bit 4 – HCMP0EN** High byte Compare 0 Enable
Setting the HCMPnEN bit in the FRQ or PWM Waveform Generation mode of operation will override the port output register for the corresponding WO[n+3] pin.

**Bit 2 – LCMP2EN** Low byte Compare 2 Enable
See LCMP0EN.

**Bit 1 – LCMP1EN** Low byte Compare 1 Enable
See LCMP0EN.

**Bit 0 – LCMP0EN** Low byte Compare 0 Enable
Setting the LCMPnEN bit in the FRQ or PWM Waveform Generation mode of operation will override the port output register for the corresponding WOn pin.

### 23.7.3 Control C - Split Mode

**Name:** CTRLC
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | HCMP2OV | HCMP1OV | HCMP0OV | | LCMP2OV | LCMP1OV | LCMP0OV |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 6 – HCMP2OV** High byte Compare 2 Output Value
See HCMP0OV.

**Bit 5 – HCMP1OV** High byte Compare 1 Output Value
See HCMP0OV.

**Bit 4 – HCMP0OV** High byte Compare 0 Output Value
The HCMPnOV bit allows direct access to the output compare value of the waveform generator when the timer/counter is not enabled. This is used to set or clear the WO[n+3] output value when the timer/counter is not running.

**Bit 2 – LCMP2OV** Low byte Compare 2 Output Value
See LCMP0OV.

**Bit 1 – LCMP1OV** Low byte Compare 1 Output Value
See LCMP0OV.

**Bit 0 – LCMP0OV** Low byte Compare 0 Output Value
The LCMPnOV bit allows direct access to the output compare value of the waveform generator when the timer/counter is not enabled. This is used to set or clear the WOn output value when the timer/counter is not running.

### 23.7.4 Control D

**Name:** CTRLD
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

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

**Bit 0 – SPLITM** Enable Split Mode
This bit sets the timer/counter in Split mode operation. It will then work as two 8-bit timer/counters. The register map will change compared to normal 16-bit mode.

### 23.7.5 Control Register E Clear - Split Mode

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

This register can be used instead of a Read-Modify-Write (RMW) to clear individual bits by writing a '1' to its bit location.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMD[1:0] | | CMDEN[1:0] | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:2 – CMD[1:0]** Command
These bits are used for software control of restart and reset of the timer/counter. The command bits are always read as '0'.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No command |
| 0x1 | - | Reserved |
| 0x2 | RESTART | Force restart |
| 0x3 | RESET | Force hard Reset (ignored if the timer/counter is enabled) |

**Bits 1:0 – CMDEN[1:0]** Command Enable
These bits configure what timer/counters the command given by the CMD-bits will be applied to.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | None |
| 0x1 | - | Reserved |
| 0x2 | - | Reserved |
| 0x3 | BOTH | Command (CMD) will be applied to both low byte and high byte timer/counter |

### 23.7.6 Control Register E Set - Split Mode

**Name:** CTRLESET
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

This register can be used instead of a Read-Modify-Write (RMW) to set individual bits by writing a '1' to its bit location.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMD[1:0] | | CMDEN[1:0] | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:2 – CMD[1:0]** Command
This bit field used for software control of restart and reset of the timer/counter. The command bits are always read as '0'. The CMD bit field must be used together with the Command Enable (CMDEN) bits. Using the RESET command requires that both low byte and high byte timer/counter are selected with CMDEN.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No command |
| 0x1 | - | Reserved |
| 0x2 | RESTART | Force restart |
| 0x3 | RESET | Force hard Reset (ignored if the timer/counter is enabled) |

**Bits 1:0 – CMDEN[1:0]** Command Enable
These bits configure what timer/counters the command given by the CMD-bits will be applied to.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | None |
| 0x1 | - | Reserved |
| 0x2 | - | Reserved |
| 0x3 | BOTH | Command (CMD) will be applied to both low byte and high byte timer/counter |

### 23.7.7 Interrupt Control Register - Split Mode

**Name:** INTCTRL
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | LCMP2 | LCMP1 | LCMP0 | | | HUNF | LUNF |
| Access | | R/W | R/W | R/W | | | R/W | R/W |
| Reset | | 0 | 0 | 0 | | | 0 | 0 |

**Bit 6 – LCMP2** Low byte Compare Channel 2 Interrupt Enable
See LCMP0.

**Bit 5 – LCMP1** Low byte Compare Channel 1 Interrupt Enable
See LCMP0.

**Bit 4 – LCMP0** Low byte Compare Channel 0 Interrupt Enable
Writing the LCMPn bit to '1' enables the low byte Compare Channel n interrupt.

**Bit 1 – HUNF** High byte Underflow Interrupt Enable
Writing the HUNF bit to '1' enables the high byte underflow interrupt.

**Bit 0 – LUNF** Low byte Underflow Interrupt Enable
Writing the LUNF bit to '1' enables the low byte underflow interrupt.

### 23.7.8 Interrupt Flag Register - Split Mode

**Name:** INTFLAGS
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | LCMP2 | LCMP1 | LCMP0 | | | HUNF | LUNF |
| Access | | R/W | R/W | R/W | | | R/W | R/W |
| Reset | | 0 | 0 | 0 | | | 0 | 0 |

**Bit 6 – LCMP2** Low byte Compare Channel 2 Interrupt Flag
See LCMP0 flag description.

**Bit 5 – LCMP1** Low byte Compare Channel 1 Interrupt Flag
See LCMP0 flag description.

**Bit 4 – LCMP0** Low byte Compare Channel 0 Interrupt Flag
The Low byte Compare Interrupt (LCMPn) flag is set on a compare match on the corresponding compare channel in the low byte timer.
For all modes of operation, the LCMPn flag will be set when a compare match occurs between the Low Byte Timer Counter (TCAn.LCNT) register and the corresponding Compare n (TCAn.LCMPn) register. The LCMPn flag will not be cleared automatically and has to be cleared by software. This is done by writing a '`1`' to its bit location.

**Bit 1 – HUNF** High byte Underflow Interrupt Flag
This flag is set on a high byte timer BOTTOM (underflow) condition. HUNF is not automatically cleared and needs to be cleared by software. This is done by writing a '`1`' to its bit location.

**Bit 0 – LUNF** Low byte Underflow Interrupt Flag
This flag is set on a low byte timer BOTTOM (underflow) condition. LUNF is not automatically cleared and needs to be cleared by software. This is done by writing a '`1`' to its bit location.

### 23.7.9 Debug Control Register

**Name:** DBGCTRL
**Offset:** 0x0E
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Run in Debug

| Value | Description |
|---|---|
| 0 | The peripheral is halted in Break Debug mode and ignores events. |
| 1 | The peripheral will continue to run in Break Debug mode when the CPU is halted. |

### 23.7.10 Low Byte Timer Counter Register - Split Mode

**Name:** LCNT
**Offset:** 0x20
**Reset:** 0x00
**Property:** -

TCAn.LCNT contains the counter value for the low byte timer. CPU and UPDI write access has priority over count, clear or reload of the counter.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LCNT[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 – LCNT[7:0]** Counter Value for Low Byte Timer
These bits define the counter value of the low byte timer.

### 23.7.11 High Byte Timer Counter Register - Split Mode

**Name:** HCNT
**Offset:** 0x21
**Reset:** 0x00
**Property:** -

TCAn.HCNT contains the counter value for the high byte timer. CPU and UPDI write access has priority over count, clear or reload of the counter.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | HCNT[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 – HCNT[7:0]** Counter Value for High Byte Timer
These bits define the counter value in high byte timer.

### 23.7.12 Low Byte Timer Period Register - Split Mode

**Name:** LPER
**Offset:** 0x26
**Reset:** 0xFF
**Property:** -

The TCAn.LPER register contains the TOP value for the low byte timer.

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

**Bits 7:0 – LPER[7:0]** Period Value Low Byte Timer
These bits hold the TOP value for the low byte timer.

### 23.7.13 High Byte Period Register - Split Mode

**Name:** HPER
**Offset:** 0x27
**Reset:** 0xFF
**Property:** -

The TCAn.HPER register contains the TOP value for the high byte timer.

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

**Bits 7:0 – HPER[7:0]** Period Value High Byte Timer
These bits hold the TOP value for the high byte timer.

### 23.7.14 Compare Register n For Low Byte Timer - Split Mode

**Name:** LCMPn
**Offset:** 0x28 + n*0x02 [n=0..2]
**Reset:** 0x00
**Property:** -

The TCAn.LCMPn register represents the compare value of Compare Channel n for the low byte timer. This register is continuously compared to the counter value of the low byte timer, TCAn.LCNT. Normally, the outputs from the comparators are then used to generate waveforms.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LCMP[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 – LCMP[7:0]** Compare Value of Channel n
These bits hold the compare value of channel n that is compared to TCAn.LCNT.

### 23.7.15 High Byte Compare Register n - Split Mode

**Name:** HCMPn
**Offset:** 0x29 + n*0x02 [n=0..2]
**Reset:** 0x00
**Property:** -

The TCAn.HCMPn register represents the compare value of Compare Channel n for the high byte timer. This register is continuously compared to the counter value of the high byte timer, TCAn.HCNT. Normally, the outputs from the comparators are then used to generate waveforms.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | HCMP[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 – HCMP[7:0]** Compare Value of Channel n
These bits hold the compare value of channel n that is compared to TCAn.HCNT.

# 24. TCB - 16-bit Timer/Counter Type B

## 24.1 Features

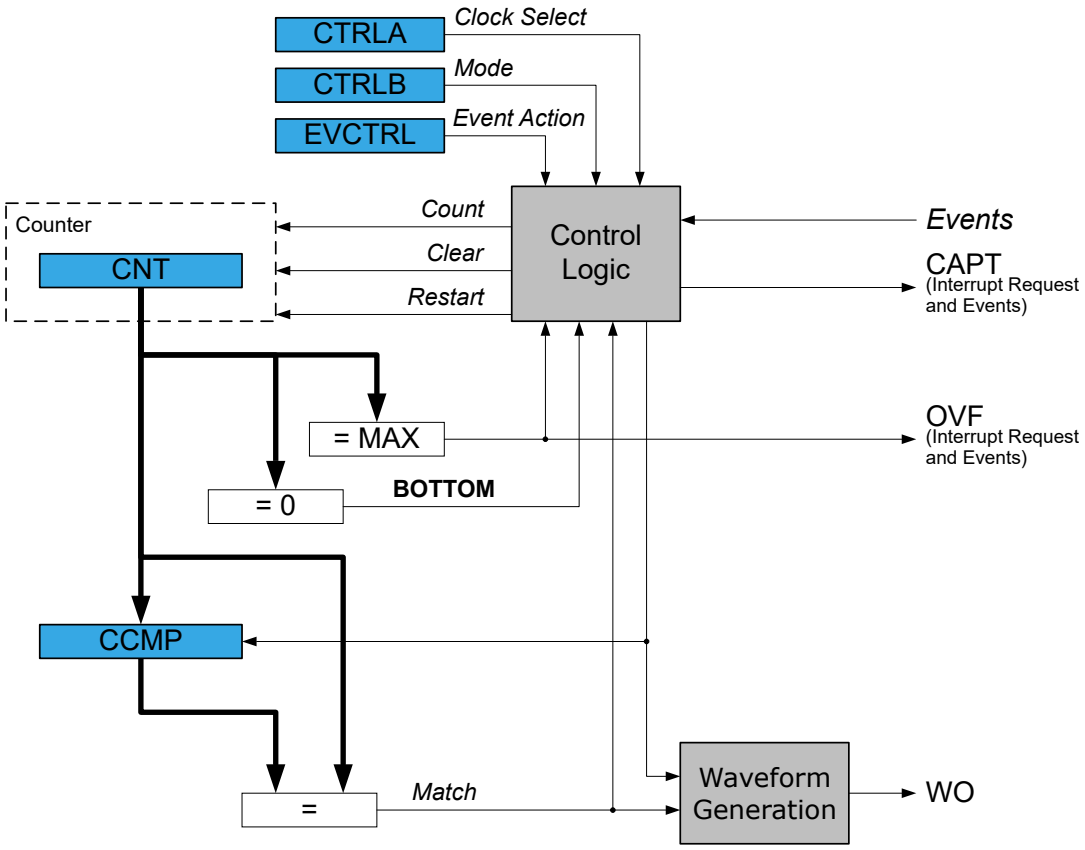
- 16-bit Counter Operation Modes:
  - Periodic interrupt
  - Time-out check
  - Input capture
    - On event
    - Frequency measurement
    - Pulse-width measurement
    - Frequency and pulse-width measurement
    - 32-bit capture
  - Single-shot
  - 8-bit Pulse-Width Modulation (PWM)
- Noise Canceler on Event Input
- Synchronize Operation with TCAn

## 24.2 Overview

The capabilities of the 16-bit Timer/Counter type B (TCB) include frequency and waveform generation, and input capture on event with time and frequency measurement of digital signals. The TCB consists of a base counter and control logic that can be set in one of eight different modes, each mode providing unique functionality. The base counter is clocked by the peripheral clock with optional prescaling.

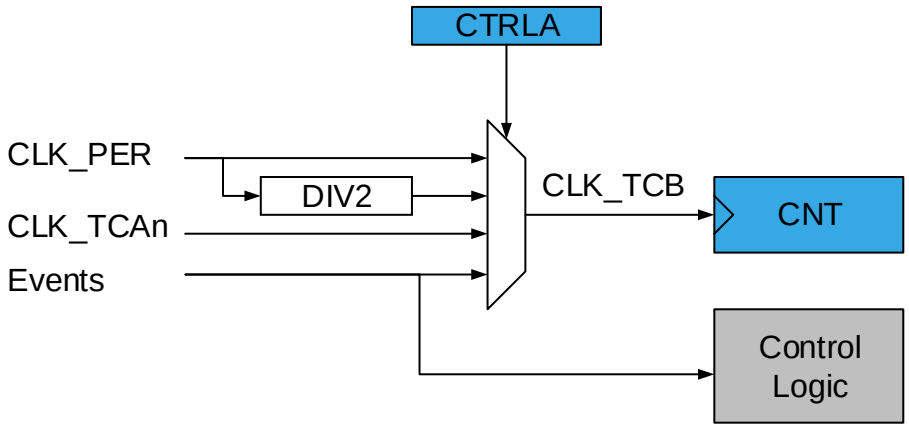
### 24.2.1 Block Diagram

**Figure 24-1. Timer/Counter Type B Block Diagram**




The timer/counter can be clocked from the Peripheral Clock (CLK_PER), from a 16-bit Timer/Counter type A (CLK_TCAn) or the Event System (EVSYS).

**Figure 24-2. Timer/Counter Clock Logic**



The Clock Select (CLKSEL) bit field in the Control A (TCBn.CTRLA) register selects one of the prescaler outputs directly, or an event channel as the clock (CLK_TCB) input.

Setting the timer/counter to use the clock from a TCAn allows the timer/counter to run in sync with that TCAn.

By using the EVSYS, any event source, such as an external clock signal on any I/O pin, may be used as the counter clock input or as a control logic input. When an event action controlled operation is used, the clock selection must be set to use an event channel as the counter input.

### 24.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| WO | Digital Asynchronous Output | Waveform Output |

## 24.3 Functional Description

### 24.3.1 Definitions

The following definitions are used throughout the documentation:

**Table 24-1. Timer/Counter Definitions**

| Name | Description |
|---|---|
| BOTTOM | The counter reaches BOTTOM when it becomes 0x0000 |
| MAX | The counter reaches the maximum when it becomes 0xFFFF |
| TOP | The counter reaches TOP when it becomes equal to the highest value in the count sequence |
| CNT | Count (TCBn.CNT) register value |
| CCMP | Capture/Compare (TCBn.CCMP) register value |

**Note:** In general, the term ‘timer’ is used when the timer/counter is counting periodic clock ticks. The term ‘counter’ is used when the input signal has sporadic or irregular ticks.

### 24.3.2 Initialization

By default, the TCB is in Periodic Interrupt mode. Follow these steps to start using it:

1. Write a TOP value to the Compare/Capture (TCBn.CCMP) register.
2. Optional: Write the Compare/Capture Output Enable (CCMPEN) bit in the Control B (TCBn.CTRLB) register to ‘`1`’. This will make the waveform output available on the corresponding pin, overriding the value in the corresponding PORT output register. The corresponding pin direction must be configured as an output in the PORT peripheral.
3. Enable the counter by writing a ‘`1`’ to the ENABLE bit in the Control A (TCBn.CTRLA) register.
   The counter will start counting clock ticks according to the prescaler setting in the Clock Select (CLKSEL) bit field in the Control A (TCBn.CTRLA) register.
4. The counter value can be read from the Count (TCBn.CNT) register. The peripheral will generate a CAPT interrupt and event when the CNT value reaches TOP.
   - 4.1. If the Compare/Capture register is modified to a value lower than the current CNT, the peripheral will count to MAX and wrap around.
   - 4.2. At MAX, an OVF interrupt and event will be generated.

### 24.3.3 Operation

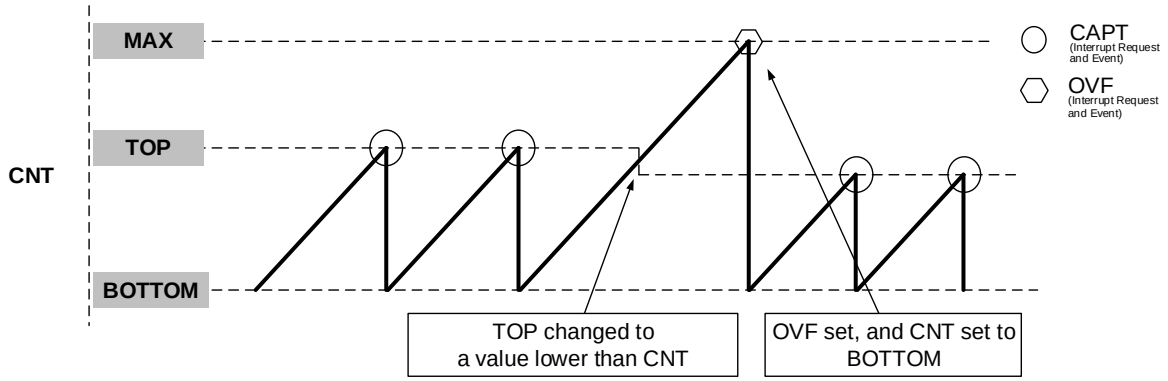
#### 24.3.3.1 Modes

The timer can be configured to run in one of the eight different modes described in the sections below. The event pulse needs to be longer than one peripheral clock cycle to ensure edge detection.

#### 24.3.3.1.1 Periodic Interrupt Mode

In the Periodic Interrupt mode, the counter counts to the capture value and restarts from BOTTOM. A CAPT interrupt and event is generated when the CNT is equal to TOP. If TOP is updated to a value lower than CNT, upon reaching MAX, an OVF interrupt and event is generated, and the counter restarts from BOTTOM.

**Figure 24-3. Periodic Interrupt Mode**



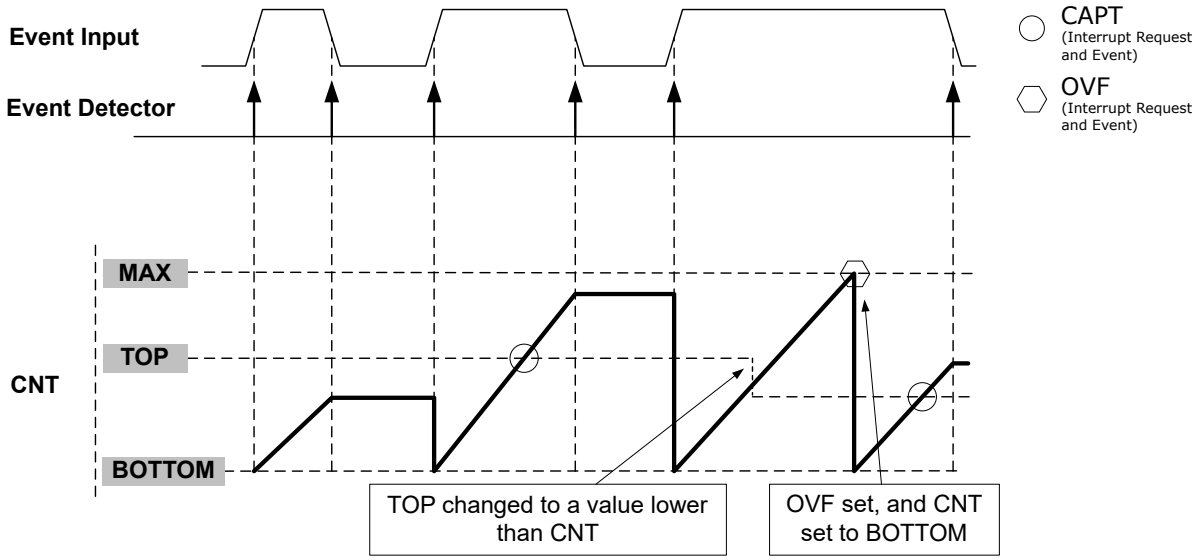

#### 24.3.3.1.2 Time-Out Check Mode

In the Time-Out Check mode, the peripheral starts counting on the first signal edge and stops on the next signal edge detected on the event input channel. CNT remains stationary after the Stop edge (Freeze state). In Freeze state, the counter will restart on a new Start edge.

This mode requires TCB to be configured as an event user, and is explained in Events section.

Start or Stop edge is determined by the Event Edge (EDGE) bit in the Event Control (TCBn.EVCTRL) register. If CNT reaches TOP before the second edge, a CAPT interrupt and event will be generated. If TOP is updated to a value lower than the CNT upon reaching MAX, an OVF interrupt and the simultaneous event is generated, and the counter restarts from BOTTOM. In Freeze state, reading the Count (TCBn.CNT) register or Compare/Capture (TCBn.CCMP) register, or writing the Run (RUN) bit in the Status (TCBn.STATUS) register has no effect.

**Figure 24-4. Time-Out Check Mode**



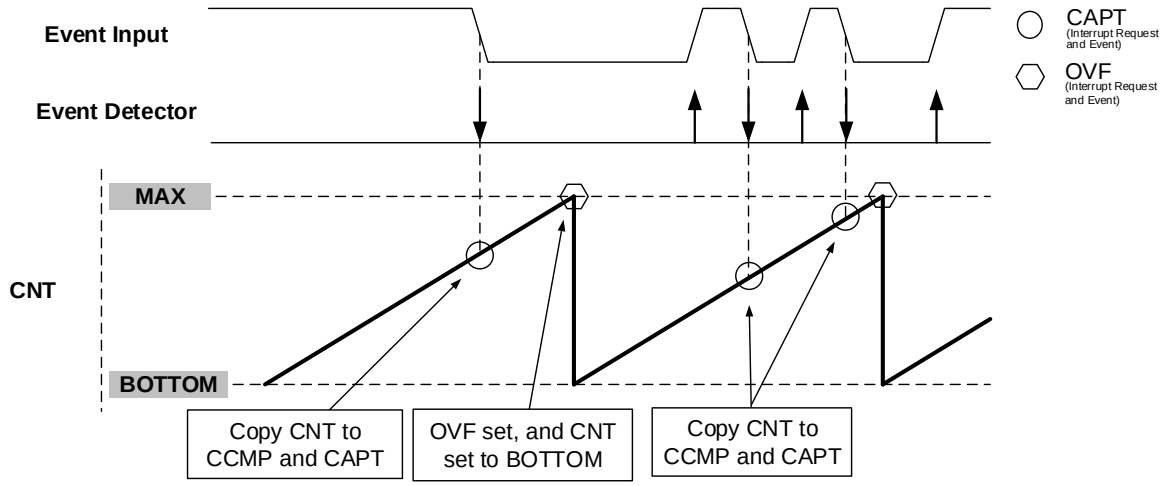
#### 24.3.3.1.3 Input Capture on Event Mode

In the Input Capture on Event mode, the counter will count from BOTTOM to MAX continuously. When an event is detected, the CNT is transferred to the Capture/Compare (TCBn.CCMP) register, and a CAPT interrupt and event is generated. The Event edge detector can be configured to trigger a capture on either rising or falling edges.

This mode requires TCB to be configured as an event user, and is explained in Events section.

The figure below shows the input capture unit configured to capture on the falling edge of the event input signal. The CAPT Interrupt flag is automatically cleared after the low byte of the Compare/Capture (TCBn.CCMP) register has been read. An OVF interrupt and event is generated when the CNT is MAX.

**Figure 24-5. Input Capture on Event**




**Important:** It is recommended to write 0x0000 to the Count (TCBn.CNT) register when entering this mode from any other mode.

#### 24.3.3.1.4 Input Capture Frequency Measurement Mode

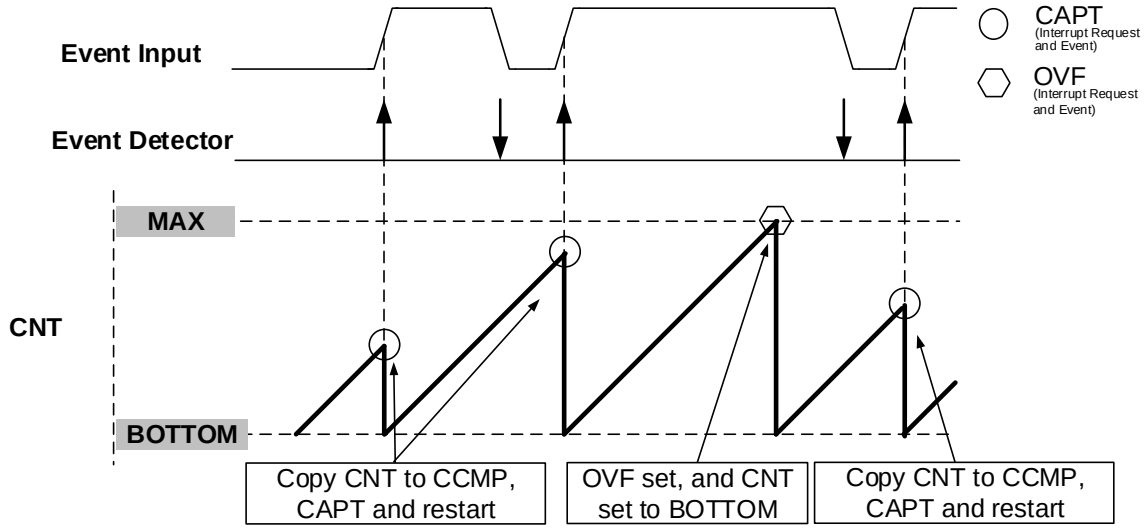
In the Input Capture Frequency Measurement mode, the TCB captures the counter value and restarts on either a positive or negative edge of the event input signal.

This mode requires TCB to be configured as an event user, and is explained in Events section.

The CAPT Interrupt flag is automatically cleared after the low byte of the Compare/Capture (TCBn.CCMP) register has been read. An OVF interrupt and event is generated when the CNT value is MAX.

The figure below illustrates this mode when configured to act on a rising edge.

**Figure 24-6. Input Capture Frequency Measurement**

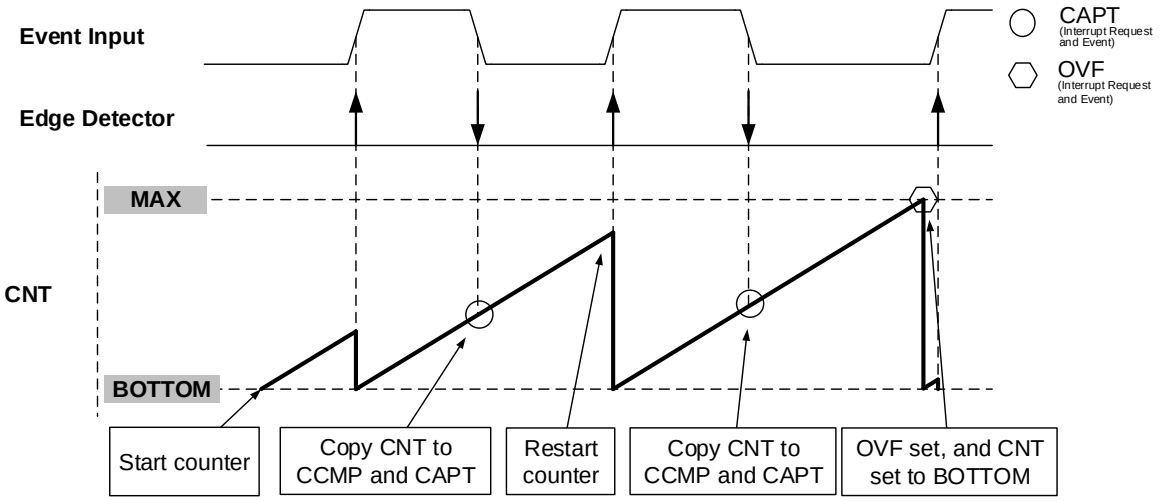



#### 24.3.3.1.5 Input Capture Pulse-Width Measurement Mode

In the Input Capture Pulse-Width Measurement mode, the input capture pulse-width measurement will restart the counter on a positive edge, and capture on the next falling edge before an interrupt request is generated. The CAPT Interrupt flag is automatically cleared after the low byte of the Compare/Capture (TCBn.CCMP) register has been read. An OVF interrupt and event is generated when the CNT is MAX. The timer will automatically switch between rising and falling edge detection, but a minimum edge separation of two clock cycles is required for correct behavior.

This mode requires TCB to be configured as an event user, and is explained in Events section.

**Figure 24-7. Input Capture Pulse-Width Measurement**



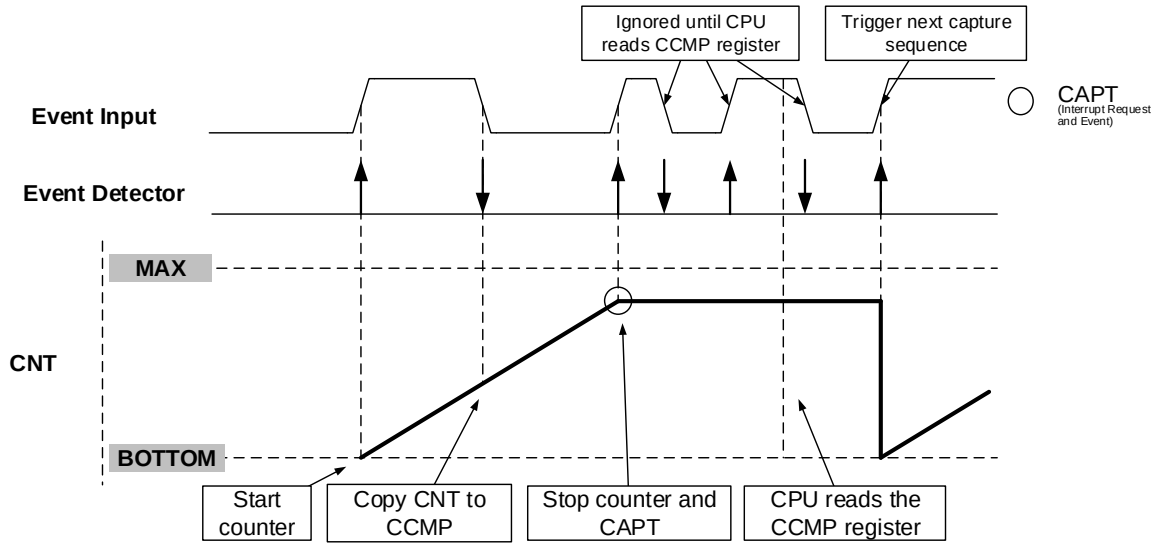

#### 24.3.3.1.6 Input Capture Frequency and Pulse-Width Measurement Mode

In the Input Capture Frequency and Pulse-Width Measurement mode, the timer will start counting when a positive edge is detected on the event input signal. The count value is captured on the following falling edge. The counter stops when the second rising edge of the event input signal is detected. This will set the CAPT interrupt flag.

This mode requires TCB to be configured as an event user, and is explained in Events section.

The CAPT Interrupt flag is automatically cleared after the low byte of the Compare/Capture (TCBn.CCMP) register has been read, and the timer/counter is ready for a new capture sequence. Therefore, the Count (TCBn.CNT) register must be read before the Compare/Capture (TCBn.CCMP) register, since it is reset to BOTTOM at the next positive edge of the event input signal. An OVF interrupt and event is generated when the CNT value is MAX.

**Figure 24-8. Input Capture Frequency and Pulse-Width Measurement**




#### 24.3.3.1.7 Single-Shot Mode

The Single-Shot mode can be used to generate a pulse with a duration defined by the Compare (TCBn.CCMP) register, every time a rising or falling edge is observed on a connected event channel.

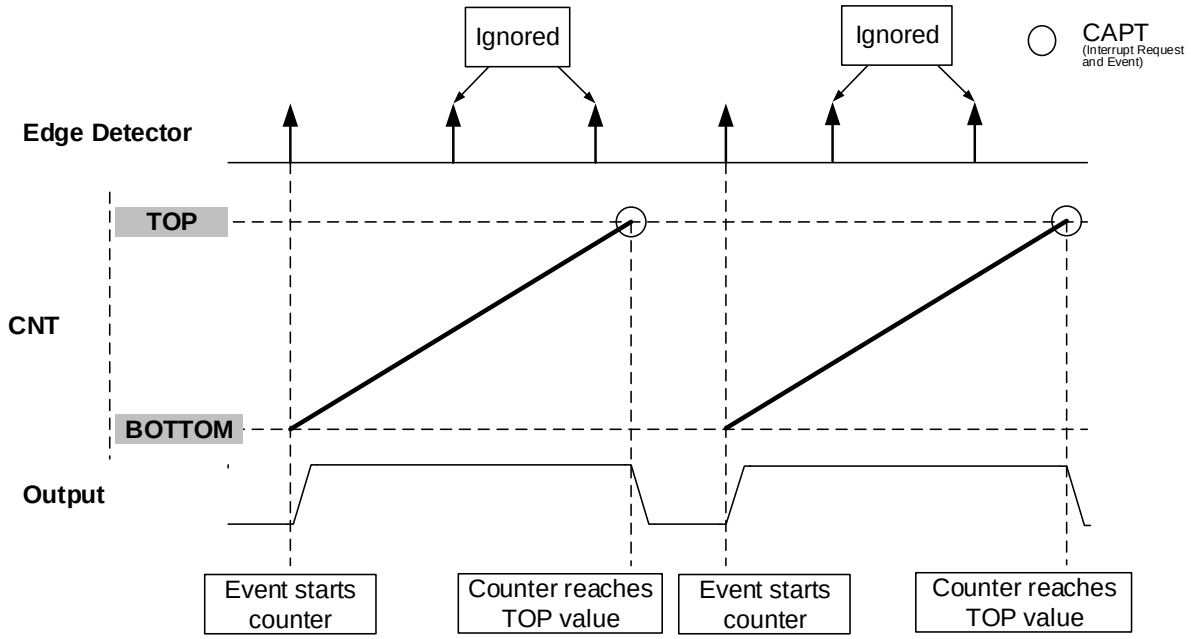
This mode requires TCB to be configured as an event user, and is explained in Events section.

When the counter is stopped, the output pin is set low. If an event is detected on the connected event channel, the timer will reset and start counting from BOTTOM to TOP while driving its output high. The RUN bit in the Status (TCBn.STATUS) register can be read to see if the counter is counting or not. When CNT reaches the CCMP register value, the counter will stop, and the output pin will go low for at least one counter clock cycle (TCB_CLK), and a new event arriving during this time will be ignored. After this, there is a delay of two peripheral clock cycles (PER_CLK) from when a new event is received until the output is set high.

The counter will start counting as soon as the peripheral is enabled, even without triggering by an event, or if the Event Edge (EDGE) bit in the Event Control (TCBn.EVCTRL) register is modified while the peripheral is enabled. This is prevented by writing TOP to the Counter register. Similar behavior is seen if the Event Edge (EDGE) bit in the Event Control (TCBn.EVCTRL) register is '`1`' while the module is enabled. Writing TOP to the Counter register prevents this as well.

If the Event Asynchronous (ASYNC) bit in the Control B (TCBn.CTRLB) register is written to '`1`', the timer will react asynchronously to an incoming event. An edge on the event will immediately cause the output signal to be set. The counter will still start counting two clock cycles after the event is received.

**Figure 24-9. Single-Shot Mode**

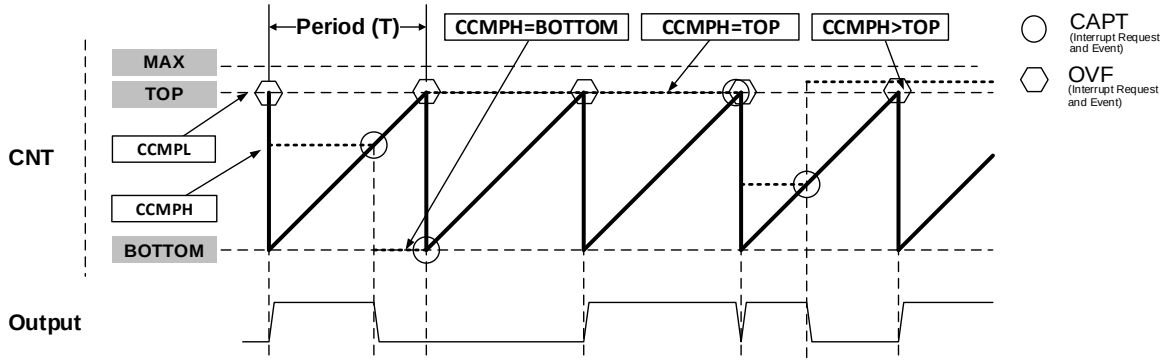



#### 24.3.3.1.8 8-Bit PWM Mode

The TCB can be configured to run in 8-bit PWM mode, where each of the register pairs in the 16-bit Compare/Capture (TCBn.CCMPH and TCBn.CCMPL) register are used as individual Compare registers. The period (T) is controlled by CCMPL, while CCMPH controls the duty cycle of the waveform. The counter will continuously count from BOTTOM to CCMPL, and the output will be set at BOTTOM and cleared when the counter reaches CCMPH.

CCMPH is the number of cycles for which the output will be driven high. CCMPL+1 is the period of the output pulse.

**Figure 24-10. 8-Bit PWM Mode**




### 24.3.3.2 Output

Timer synchronization and output logic level are dependent on the selected Timer Mode (CNTMODE) bit field in Control B (TCBn.CTRLB) register. In Single-Shot mode, the timer/counter can be configured so that the signal generation happens asynchronously to an incoming event (ASYNC = `1` in TCBn.CTRLB). The output signal is then set immediately at the incoming event instead of being synchronized to the TCB clock. Even though the output is set immediately, it will take two to three CLK_TCB cycles before the counter starts counting.

Writing the Compare/Capture Output Enable (CCMPEN) bit in TCBn.CTRLB to '1' enables the waveform output. This will make the waveform output available on the corresponding pin, overriding the value in the corresponding PORT output register. The corresponding pin direction must be configured as an output in the PORT peripheral.

The different configurations and their impact on the output are listed in the table below.

**Table 24-2. Output Configuration**

| CCMPEN | CNTMODE | ASYNC | Output |
|---|---|---|---|
| 1 | Single-Shot mode | 0 | The output is high when the counter starts, and the output is low when the counter stops |
| | | 1 | The output is high when the event arrives, and the output is low when the counter stops |
| | 8-bit PWM mode | Not applicable | 8-bit PWM mode |
| | Other modes | Not applicable | The Compare/Capture Pin Initial Value bit (CCMPINIT) in the Control B (TCBn.CTRLB) register selects the initial output level |
| 0 | Not applicable | Not applicable | No output |

It is not recommended to change modes while the peripheral is enabled, as this can produce an unpredictable output. There is a possibility that an interrupt flag is set during the timer configuration. It is recommended to clear the Timer/Counter Interrupt Flags (TCBn.INTFLAGS) register after configuring the peripheral.

#### 24.3.3.3 32-Bit Input Capture

Two 16-bit Timer/Counter Type B (TCBn) can be combined to work as a true 32-bit input capture:

One TCB is counting the two LSBs. Once this counter reaches MAX, an overflow (OVF) event is generated, and the counter wraps around. The second TCB is configured to count these OVF events and thus provides the two MSBs. The 32-bit counter value is concatenated from the two counter values.

To function as a 32-bit counter, the two TCBs and the system have to be set up as described in the following paragraphs.

**System Configuration**

- Configure a source (TCA, events, CLK_PER) for the count input for the LSB TCB, according to the application requirements
- Configure the event system to route the OVF events from the LSB TCB (event generator) to the TCB intended for counting the MSB (event user)
- Configure the event system to route the same capture event (CAPT) generator to both TCBs

**Configuration of the LSB Counter**

- Select the configured count input by writing the Clock Select (CLKSEL) bit field in the Control A (CTRLA) register
- Write the Timer Mode (CNTMODE) bit field in the Control B (CTRLB) register to select the Input Capture on Event mode
- Ensure that the Cascade Two Timer/Counters (CASCADE) bit in CTRLA is '0'

**Configuration of the MSB Counter**

- Enable the 32-bit mode by writing the Cascade Two Timer/Counters bit (CASCADE) in CTRLA to '1'
- Select events as clock input by writing to the Clock Select (CLKSEL) bit field in the Control A (CTRLA) register
- Write the Timer Mode (CNTMODE) bit field in the Control B (CTRLB) register to select the Input Capture on Event mode

**Capturing a 32-Bit Counter Value**

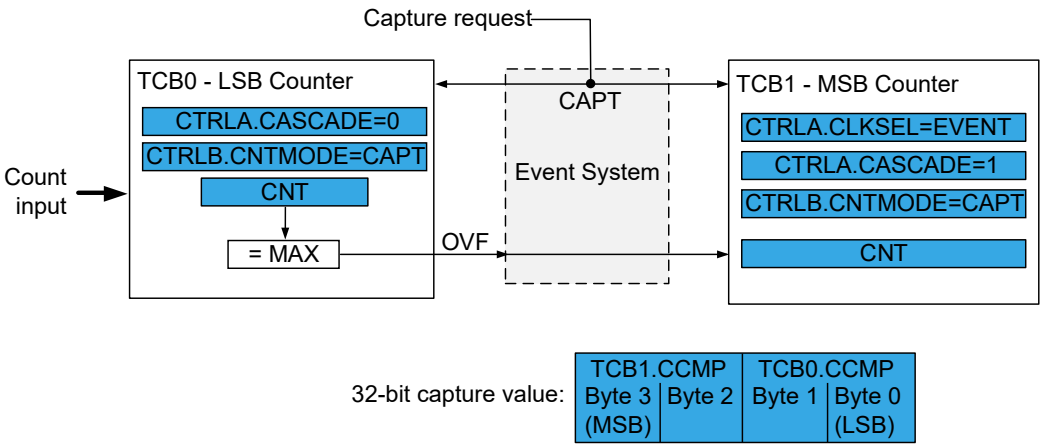
To acquire a 32-bit counter value, send a CAPT event to both TCBs. Both TCBs are running in Input Capture on Event mode, so each will capture the current counter value (CNT) in the respective Capture/Compare (CCMP) register. The 32-bit capture value is formed by concatenating the two CCMP registers.

**Example 24-1. Using TCB0 as LSB Counter and TCB1 as MSB Counter**

TCB0 is counting the count input, and TCB1 is counting the OVF signals from TCB0.

A CAPT event is generated and causes both TCB0 and TCB1 to copy their current CNT values to their respective CCMP registers. The two different CASCADE bit values allow correct timing of the CAPT event.

The captured 32-bit value is concatenated from TCB1.CCMP (MSB) and TCB0.CCMP (LSB).




### 24.3.3.4 Noise Canceler

The Noise Canceler improves the noise immunity by using a simple digital filter scheme. When the Noise Filter (FILTER) bit in the Event Control (TCBn.EVCTRL) register is enabled, the peripheral monitors the event channel and keeps a record of the last four observed samples. If four consecutive samples are equal, the input is considered to be stable, and the signal is fed to the edge detector.

When enabled, the Noise Canceler introduces an additional delay of four peripheral clock cycles between a change applied to the input and the update of the Input Compare register.

The Noise Canceler uses the peripheral clock and is, therefore, not affected by the prescaler.

### 24.3.3.5 Synchronized with Timer/Counter Type A

The TCB can be configured to use the clock (CLK_TCA) of a Timer/Counter type A (TCAn) by writing to the Clock Select bit field (CLKSEL) in the Control A register (TCBn.CTRLA). In this setting, the TCB will count on the same clock source as selected in TCAn.

When the Synchronize Update (SYNCUPD) bit in the Control A (TCBn.CTRLA) register is written to '1', the TCB counter will restart when the TCAn counter restarts.

## 24.3.4 Events

The TCB can generate the events described in the following table:

**Table 24-3. Event Generators in TCB**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| TCBn | CAPT | CAPT flag set | Pulse | CLK_PER | One CLK_PER period |
| | OVF | OVF flag set | | | |

The conditions for generating the CAPT and OVF events are identical to those that will raise the corresponding interrupt flags in the Timer/Counter Interrupt Flags (TCBn.INTFLAGS) register. Refer to the *Event System* section for more details regarding event users and Event System configuration.

The TCB can receive the events described in the following table:

**Table 24-4. Event Users and Available Event Actions in TCB**

| User Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| TCBn | CAPT | Time-Out Check Count mode | Edge | Sync |
| | | Input Capture on Event Count mode | | |
| | | Input Capture Frequency Measurement Count mode | | |
| | | Input Capture Pulse-Width Measurement Count mode | | |
| | | Input Capture Frequency and Pulse-Width Measurement Count mode | | |
| | | Single-Shot Count mode | | Both |
| | COUNT | Event as clock source in combination with a count mode | | Sync |

CAPT and COUNT are TCB event users that detect and act upon input events.

The COUNT event user is enabled on the peripheral by modifying the Clock Select (CLKSEL) bit field in the Control A (TCBn.CTRLA) register to EVENT and setting up the Event System accordingly.

If the Capture Event Input Enable (CAPTEI) bit in the Event Control (TCBn.EVCTRL) register is written to '`1`', incoming events will result in an event action as defined by the Event Edge (EDGE) bit in Event Control (TCBn.EVCTRL) register and the Timer Mode (CNTMODE) bit field in Control B (TCBn.CTRLB) register. The event needs to last for at least one CLK_PER cycle to be recognized.

If the Asynchronous mode is enabled for Single-Shot mode, the event is edge-triggered and will capture changes on the event input shorter than one peripheral clock cycle.

### 24.3.5 Interrupts

**Table 24-5. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| CAPT | TCB interrupt | Depending on the operating mode. See the description of the CAPT bit in the TCBn.INTFLAG register. |
| OVF | | The timer/counter overflows from MAX to BOTTOM. |

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

### 24.3.6 Sleep Mode Operation

TCBn is, by default, disabled in Standby sleep mode. It will be halted as soon as the sleep mode is entered.

The module can stay fully operational in the Standby sleep mode if the Run Standby (RUNSTDBY) bit in the TCBn.CTRLA register is written to '`1`'.

All operations are halted in Power-Down sleep mode.

## 24.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | RUNSTDBY | CASCADE | SYNCUPD | CLKSEL[2:0] | | | ENABLE |
| 0x01 | CTRLB | 7:0 | | ASYNC | CCMPINIT | CCMPEN | | CNTMODE[2:0] | | |
| 0x02<br>...<br>0x03 | Reserved | | | | | | | | | |
| 0x04 | EVCTRL | 7:0 | | FILTER | | EDGE | | | | CAPTEI |
| 0x05 | INTCTRL | 7:0 | | | | | | | OVF | CAPT |
| 0x06 | INTFLAGS | 7:0 | | | | | | | OVF | CAPT |
| 0x07 | STATUS | 7:0 | | | | | | | | RUN |
| 0x08 | DBGCTRL | 7:0 | | | | | | | | DBGRUN |
| 0x09 | TEMP | 7:0 | TEMP[7:0] | | | | | | | |
| 0x0A | CNT | 7:0 | CNT[7:0] | | | | | | | |
| | | 15:8 | CNT[15:8] | | | | | | | |
| 0x0C | CCMP | 7:0 | CCMP[7:0] | | | | | | | |
| | | 15:8 | CCMP[15:8] | | | | | | | |

## 24.5 Register Description

### 24.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | RUNSTDBY | CASCADE | SYNCUPD | CLKSEL[2:0] | | | ENABLE |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 6 – RUNSTDBY** Run Standby
Writing a '1' to this bit will enable the peripheral to run in Standby sleep mode.

**Bit 5 – CASCADE** Cascade Two Timer/Counters
Writing this bit to '1' enables cascading of two 16-bit Timer/Counters type B (TCBn) for 32-bit operation using the Event System. This bit must be '1' for the timer/counter used for the two Most Significant Bytes (MSB). When this bit is '1', the selected event source for capture (CAPT) is delayed by one peripheral clock cycle. This compensates the carry propagation delay when cascading two counters via the Event System.

**Bit 4 – SYNCUPD** Synchronize Update
When this bit is written to '1', the TCB will restart whenever TCAn is restarted or overflows. This can be used to synchronize capture with the PWM period. If TCAn is selected as the clock source, the TCB will restart when that TCAn is restarted. For other clock selections, it will restart together with TCA0.

**Bits 3:1 – CLKSEL[2:0]** Clock Select
Writing these bits selects the clock source for this peripheral.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV1 | CLK_PER |
| 0x1 | DIV2 | CLK_PER / 2 |
| 0x2 | TCA0 | CLK_TCA from TCA0 |
| 0x3 | TCA1 | CLK_TCA from TCA1 |
| 0x4-0x6 | - | Reserved |
| 0x07 | EVENT | Positive edge on event input |

**Bit 0 – ENABLE** Enable
Writing this bit to '1' enables the Timer/Counter type B peripheral.

### 24.5.2 Control B

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | ASYNC | CCMPINIT | CCMPEN | | CNTMODE[2:0] | | |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 6 – ASYNC** Asynchronous Enable
Writing this bit to '1' will allow asynchronous updates of the TCB output signal in Single-Shot mode.

| Value | Description |
|---|---|
| 0 | The output will go HIGH when the counter starts after synchronization |
| 1 | The output will go HIGH when an event arrives |

**Bit 5 – CCMPINIT** Compare/Capture Pin Initial Value
This bit is used to set the initial output value of the pin when a pin output is used. This bit has no effect in 8-bit PWM mode and Single-Shot mode.

| Value | Description |
|---|---|
| 0 | Initial pin state is LOW |
| 1 | Initial pin state is HIGH |

**Bit 4 – CCMPEN** Compare/Capture Output Enable
Writing this bit to '1' enables the waveform output. This will make the waveform output available on the corresponding pin, overriding the value in the corresponding PORT output register. The corresponding pin direction must be configured as an output in the PORT peripheral.

| Value | Description |
|---|---|
| 0 | Waveform output is not enabled on the corresponding pin. |
| 1 | Waveform output will override the output value of the corresponding pin. |

**Bits 2:0 – CNTMODE[2:0]** Timer Mode
Writing to this bit field selects the Timer mode.

| Value | Name | Description |
|---|---|---|
| 0x0 | INT | Periodic Interrupt mode |
| 0x1 | TIMEOUT | Time-out Check mode |
| 0x2 | CAPT | Input Capture on Event mode |
| 0x3 | FRQ | Input Capture Frequency Measurement mode |
| 0x4 | PW | Input Capture Pulse-Width Measurement mode |
| 0x5 | FRQPW | Input Capture Frequency and Pulse-Width Measurement mode |
| 0x6 | SINGLE | Single-Shot mode |
| 0x7 | PWM8 | 8-Bit PWM mode |

### 24.5.3 Event Control

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | FILTER | | EDGE | | | | CAPTEI |
| Access | | R/W | | R/W | | | | R/W |
| Reset | | 0 | | 0 | | | | 0 |

**Bit 6 – FILTER** Input Capture Noise Cancellation Filter
Writing this bit to '1' enables the Input Capture Noise Cancellation unit.

**Bit 4 – EDGE** Event Edge
This bit is used to select the event edge. The effect of this bit is dependent on the selected Count Mode (CNTMODE) bit field in TCBn.CTRLB. "—" means that an event or edge has no effect in this mode.

| Count Mode | EDGE | Positive Edge | Negative Edge |
|---|---|---|---|
| Periodic Interrupt mode | 0 | — | — |
| | 1 | — | — |
| Timeout Check mode | 0 | Start counter | Stop counter |
| | 1 | Stop counter | Start counter |
| Input Capture on Event mode | 0 | Input Capture, interrupt | — |
| | 1 | — | Input Capture, interrupt |
| Input Capture Frequency Measurement mode | 0 | Input Capture, clear and restart counter, interrupt | — |
| | 1 | — | Input Capture, clear and restart counter, interrupt |
| Input Capture Pulse-Width Measurement mode | 0 | Clear and restart counter | Input Capture, interrupt |
| | 1 | Input Capture, interrupt | Clear and restart counter |
| Input Capture Frequency and Pulse Width Measurement mode | 0 | • On the 1st Positive: Clear and restart counter<br>• On the following Negative: Input Capture<br>• On the 2nd Positive: Stop counter, interrupt | |
| | 1 | • On the 1st Negative: Clear and restart counter<br>• On the following Positive: Input Capture<br>• On the 2nd Negative: Stop counter, interrupt | |
| Single-Shot mode | 0 | Start counter | — |
| | 1 | — | Start counter |
| 8-Bit PWM mode | 0 | — | — |
| | 1 | — | — |

**Bit 0 – CAPTEI** Capture Event Input Enable
Writing this bit to '1' enables the input capture event.

### 24.5.4 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | OVF | CAPT |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – OVF** Overflow Interrupt Enable
Writing this bit to '1' enables interrupt on overflow.

**Bit 0 – CAPT** Capture Interrupt Enable
Writing this bit to '1' enables interrupt on capture.

### 24.5.5 Interrupt Flags

**Name:** INTFLAGS
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | OVF | CAPT |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – OVF** Overflow Interrupt Flag
This bit is set when an overflow interrupt occurs. The flag is set whenever the timer/counter wraps from MAX to BOTTOM.
The bit is cleared by writing a '1' to the bit position.

**Bit 0 – CAPT** Capture Interrupt Flag
This bit is set when a capture interrupt occurs. The interrupt conditions are dependent on the Counter Mode (CNTMODE) bit field in the Control B (TCBn.CTRLB) register.
This bit is cleared by writing a '1' to it or when the Capture register is read in Capture mode.

**Table 24-6. Interrupt Sources Set Conditions by Counter Mode**

| Counter Mode | Interrupt Set Condition | TOP Value | CAPT |
|---|---|---|---|
| Periodic Interrupt mode | Set when the counter reaches TOP | CCMP | CNT == TOP |
| Timeout Check mode | Set when the counter reaches TOP | | |
| Single-Shot mode | Set when the counter reaches TOP | | |
| Input Capture Frequency Measurement mode | Set on edge when the Capture register is loaded and the counter restarts; the flag clears when the capture is read | -- | On Event, copy CNT to CCMP, and restart counting (CNT == BOTTOM) |
| Input Capture on Event mode | Set when an event occurs and the Capture register is loaded; the flag clears when the capture is read | | On Event, copy CNT to CCMP, and continue counting |
| Input Capture Pulse-Width Measurement mode | Set on edge when the Capture register is loaded; the previous edge initialized the count; the flag clears when the capture is read | | |
| Input Capture Frequency and Pulse-Width Measurement mode | Set on the second edge (positive or negative) when the counter is stopped; the flag clears when the capture is read | | |
| 8-Bit PWM mode | Set when the counter reaches CCMH | CCML | CNT == CCMH |

### 24.5.6 Status

**Name:** STATUS
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

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

**Bit 0 – RUN** Run
When the counter is running, this bit is set to '1'. When the counter is stopped, this bit is cleared to '0'.
The bit is read-only and cannot be set by UPDI.

### 24.5.7 Debug Control

**Name:** DBGCTRL
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Debug Run

| Value | Description |
|---|---|
| 0 | The peripheral is halted in Break Debug mode and ignores events. |
| 1 | The peripheral will continue to run in Break Debug mode when the CPU is halted. |

### 24.5.8 Temporary Value

**Name:** TEMP
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

The Temporary register is used by the CPU for 16-bit single-cycle access to the 16-bit registers of this peripheral. The register is common for all the 16-bit registers of this peripheral and can be read and written by software. For more details on reading and writing 16-bit registers, refer to *Accessing 16-Bit Registers* in the *Memories* section.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TEMP[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 – TEMP[7:0]** Temporary Value

### 24.5.9 Count

**Name:** CNT
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

The TCBn.CNTL and TCBn.CNTH register pair represents the 16-bit value TCBn.CNT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

CPU and UPDI write access has priority over internal updates of the register.

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

**Bits 15:8 – CNT[15:8]** Count Value High
These bits hold the MSB of the 16-bit Counter register.

**Bits 7:0 – CNT[7:0]** Count Value Low
These bits hold the LSB of the 16-bit Counter register.

### 24.5.10 Capture/Compare

**Name:** CCMP
**Offset:** 0x0C
**Reset:** 0x00
**Property:** -

The TCBn.CCMPL and TCBn.CCMPH register pair represents the 16-bit value TCBn.CCMP. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

This register has different functions depending on the mode of operation:

- For Capture operation, these registers contain the captured value of the counter at the time the capture occurs
- In Periodic Interrupt/Time-Out and Single-Shot mode, this register acts as the TOP value
- In 8-bit PWM mode, TCBn.CCMPL and TCBn.CCMPH act as two independent registers: The period of the waveform is controlled by CCMPL, while CCMPH controls the duty cycle.

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

**Bits 15:8 – CCMP[15:8]** Capture/Compare Value High Byte
These bits hold the MSB of the 16-bit compare, capture, and top value.

**Bits 7:0 – CCMP[7:0]** Capture/Compare Value Low Byte
These bits hold the LSB of the 16-bit compare, capture, and top value.

# 25. TCD - 12-Bit Timer/Counter Type D

## 25.1 Features

- 12-bit Timer/Counter
- Programmable Prescaler
- Double-Buffered Compare Registers
- Waveform Generation:
  - One Ramp mode
  - Two Ramp mode
  - Four Ramp mode
  - Dual Slope mode
- Two Separate Input Channels
- Software and Input Based Capture
- Programmable Filter for Input Events
- Conditional Waveform Generation on External Events:
  - Fault handling
  - Input blanking
  - Overload protection
  - Fast emergency stop by hardware
- Half-Bridge and Full-Bridge Output Support

## 25.2 Overview

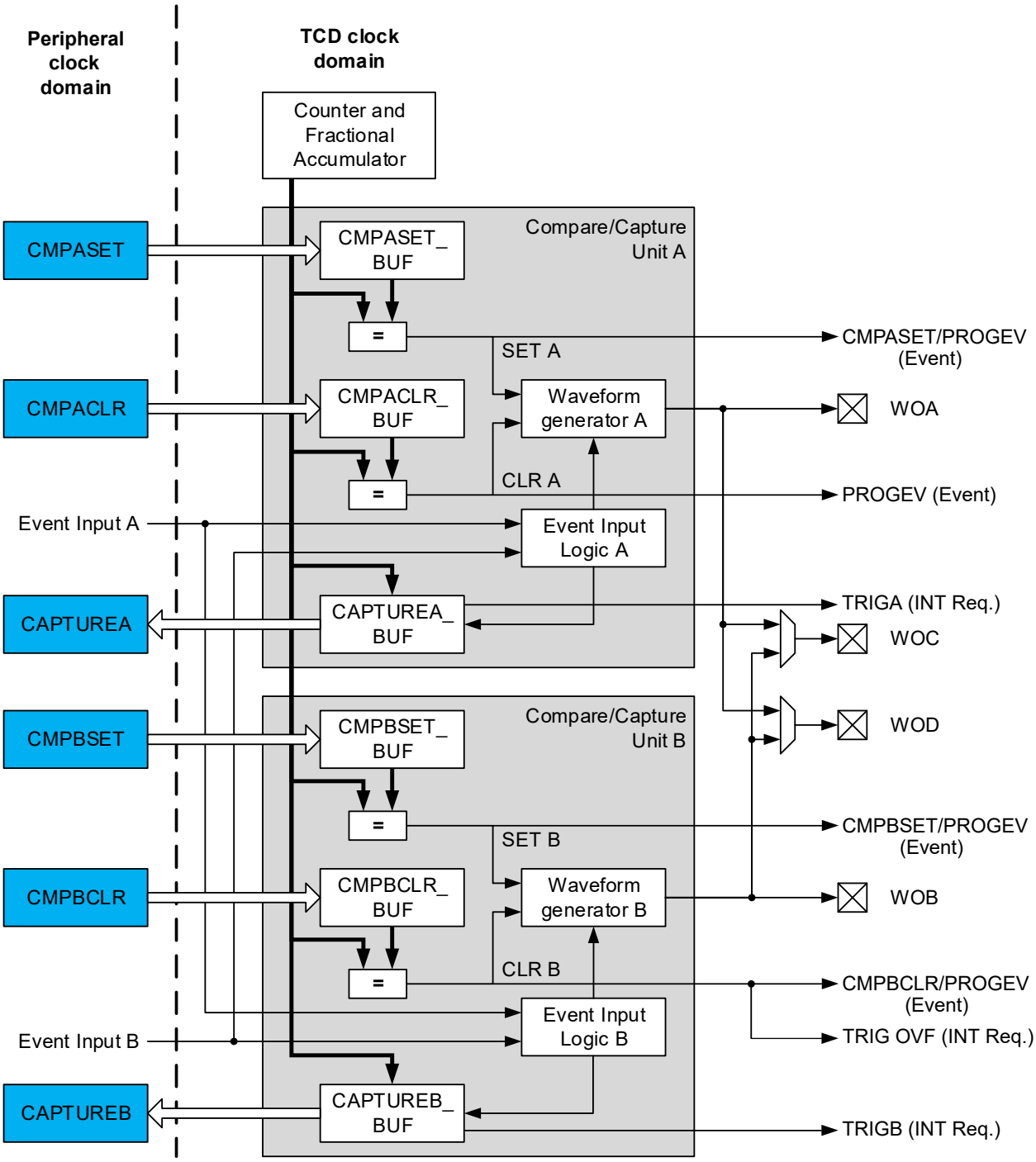
The Timer/Counter type D (TCD) is a high-performance waveform generator that consists of an asynchronous counter, a prescaler, and compare, capture and control logic.

The TCD contains a counter that can run on a clock which is asynchronous to the peripheral clock. It contains compare logic that generates two independent outputs with optional dead time. It is connected to the Event System for capture and deterministic Fault control. The timer/counter can generate interrupts and events on compare match and overflow.

This device provides one instance of the TCD peripheral, TCD0.

### 25.2.1 Block Diagram

**Figure 25-1. Timer/Counter Block Diagram**



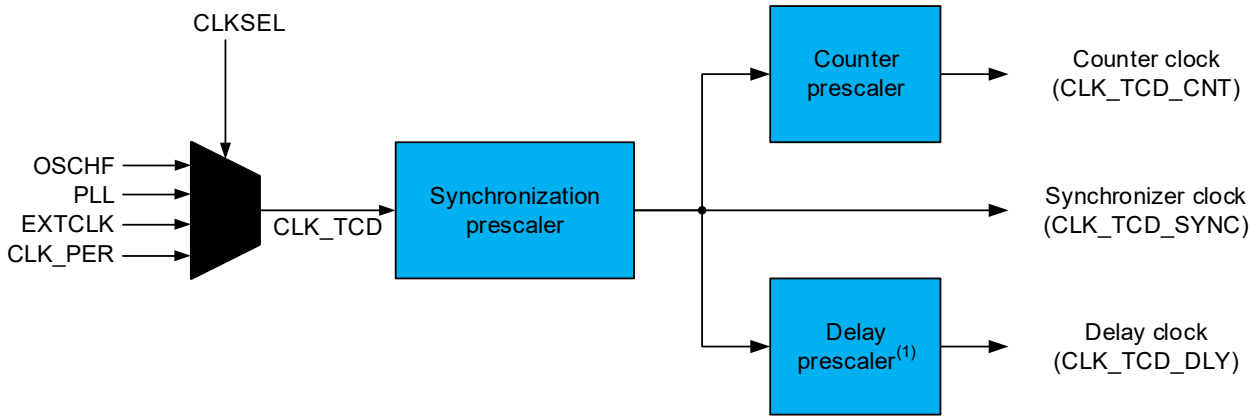

The TCD core is asynchronous to the peripheral clock. The timer/counter consists of two compare/capture units, each with a separate waveform output. There are also two extra waveform outputs which can be equal to the output from one of the units. For each compare/capture unit, there is a pair of compare registers which are stored in the respective peripheral registers (TCDn.CMPASET, TCDn.CMPACLR, TCDn.CMPBSET, TCDn.CMPBCLR).

During normal operation, the counter value is continuously compared to the compare registers. This is used to generate both interrupts and events.

The TCD can use the input events in ten different input modes, selected separately for the two input events. The input mode defines how the input events will affect the outputs, and where in the TCD cycle the counter must go when an event occurs.

The TCD can select between four different clock sources that can be prescaled. There are three different prescalers with separate controls, as shown below.

**Figure 25-2. Clock Selection and Prescalers Overview**




1. Used by input blanking/delay event out.

The TCD synchronizer clock is separate from the other module clocks, enabling faster synchronization between the TCD domain and the I/O domain.

The total prescaling for the counter is:

$$\text{SYNCPRESC_division_factor} \times \text{CNTPRESC_division_factor}$$

The delay prescaler is used to prescale the clock used for the input blanking/delayed event output functionality. The prescaler can be configured independently allowing separate range and accuracy settings from the counter functionality. The synchronization prescaler and counter prescaler can be configured from the Control A (TCDn.CTRLA) register, while the delay prescaler can be configured from the Delay Control (TCDn.DLYCTRL) register.

### 25.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| WOA | TCD waveform output A | Digital output |
| WOB | TCD waveform output B | Digital output |
| WOC | TCD waveform output C | Digital output |
| WOD | TCD waveform output D | Digital output |

## 25.3 Functional Description

### 25.3.1 Definitions

The following definitions are used throughout the documentation:

**Table 25-1. Timer/Counter Definitions**

| Name | Description |
|---|---|
| TCD cycle | The sequence of four states that the counter needs to go through before it has returned to the same position. |

| .....continued | |
|---|---|
| **Name** | **Description** |
| Input blanking | The functionality to ignore an event input for a programmable time in a selectable part of the TCD cycle. |
| Asynchronous output control | Allows the event to override the output instantly when an event occurs. It is used for handling non-recoverable Faults. |
| One ramp | The counter is reset to zero once during a TCD cycle. |
| Two ramp | The counter is reset to zero two times during a TCD cycle. |
| Four ramp | The counter is reset to zero four times during a TCD cycle. |
| Dual ramp | The counter counts both up and down between zero and a selected top value during a TCD cycle. |
| Input mode | A predefined setting that changes the output characteristics, based on the given input events. |

### 25.3.2 Initialization

To initialize the TCD:

1. Select the clock source and the prescaler from the Control A (TCDn.CTRLA) register.
2. Select the Waveform Generation Mode from the Control B (TCDn.CTRLB) register.
3. Optional: Configure the other static registers to the desired functionality.
4. Write the initial values in the Compare (TCDn.CMPxSET/CLR) registers.
5. Optional: Write the desired values to the other double-buffered registers.
6. Ensure that the Enable Ready (ENRDY) bit in the Status (TCDn.STATUS) register is set to '1'.
7. Enable the TCD by writing a '1' to the ENABLE bit in the Control A (TCDn.CTRLA) register.

### 25.3.3 Operation

#### 25.3.3.1 Register Synchronization Categories

Most of the I/O registers need to be synchronized to the TCD core clock domain. This is done differently for different register categories.

**Table 25-2. Categorization of Registers**

| Enable and Command Registers | Double-Buffered Registers | Static Registers | Read-Only Registers | Normal I/O Registers |
|---|---|---|---|---|
| TCDn.CTRLA (ENABLE bit) | TCDn.DLYCTRL | TCDn.CTRLA[(1)] (All bits except ENABLE bit) | TCDn.STATUS | TCDn.INTCTRL |
| TCDn.CTRLE | TCDn.DLYVAL | TCDn.CTRLB | TCDn.CAPTUREA | TCDn.INTFLAGS |
| | TCDn.DITCTRL | TCDn.CTRLC | TCDn.CAPTUREB | |
| | TCDn.DITVAL | TCDn.CTRLD | | |
| | TCDn.DBGCTRL | TCDn.EVCTRLA | | |
| | TCDn.CMPASET | TCDn.EVCTRLB | | |
| | TCDn.CMPACLR | TCDn.INPUTCTRLA | | |
| | TCDn.CMPBSET | TCDn.INPUTCTRLB | | |
| | TCDn.CMPBCLR | TCDn.FAULTCTRL[(2)] | | |

**Notes:**

1. The bits in the Control A (TCDn.CTRLA) register are enable-protected, except the ENABLE bit. They can only be written when ENABLE is written to '0' first.
2. This register is protected by the Configuration Change Protection Mechanism, requiring a timed write procedure for changing its value settings.

**Enable and Command Registers**

Because of the synchronization between the clock domains, it is only possible to change the ENABLE bit in the Control A (TCDn.CTRLA) register, while the Enable Ready (ENRDY) bit in the Status (TCDn.STATUS) register is '1'.

The Control E (TCDn.CTRLE) register is automatically synchronized to the TCD core domain when the TCD is enabled and as long as no synchronization is ongoing already. Check if the Command Ready (CCMDRDY) bit in TCDn.STATUS is '1' to ensure that it is possible to issue a new command. TCDn.CTRLE is a strobe register that will clear itself when the command is sent.

**Double-Buffered Registers**

The double-buffered registers can be updated in normal I/O writes, while TCD is enabled and no synchronization between the two clock domains is ongoing. Check that the CMDRDY bit in TCDn.STATUS is '1' to ensure that it is possible to update the double-buffered registers. The values will be synchronized to the TCD core domain when a synchronization command is sent or when TCD is enabled.

**Table 25-3. Issuing Synchronization Command**

| Synchronization Issuing Bit | Double Register Update |
|---|---|
| CTRLC.AUPDATE | Every time the CMPBCLRH register is written, the synchronization occurs at the end of the TCD cycle. |
| CTRLE.SYNC (1) | Occurs once, as soon as the SYNC bit is synchronized with the TDC domain. |
| CTRLE.SYNCEOC (1) | Occurs once at the end of the next TCD cycle. |

**Note:**

1. If synchronization is already ongoing, the action has no effect.

**Static Registers**

Static registers cannot be updated while TCD is enabled. Therefore, these registers must be configured before enabling TCD. To see if TCD is enabled, check if ENABLE in TCDn.CTRLA is read as '1'.

**Normal I/O and Read-Only Registers**

Normal I/O and read-only registers are not constrained by any synchronization between the domains. The read-only registers inform about synchronization status and values synchronized from the core domain.

### 25.3.3.2 Waveform Generation Modes

The TCD provides four different Waveform Generation modes controlled by the Waveform Generation Mode (WGMODE) bit field in the Control B (TCDn.CTRLB) register. The Waveform Generation modes are:

- One Ramp mode
- Two Ramp mode
- Four Ramp mode
- Dual Slope mode

The Waveform Generation modes determine how the counter is counting during a TCD cycle and how the compare values influence the waveform. A TCD cycle is split into these states:

- Dead time WOA (DTA)
- On time WOA (OTA)
- Dead time WOB (DTB)

- On time WOB (OTB)

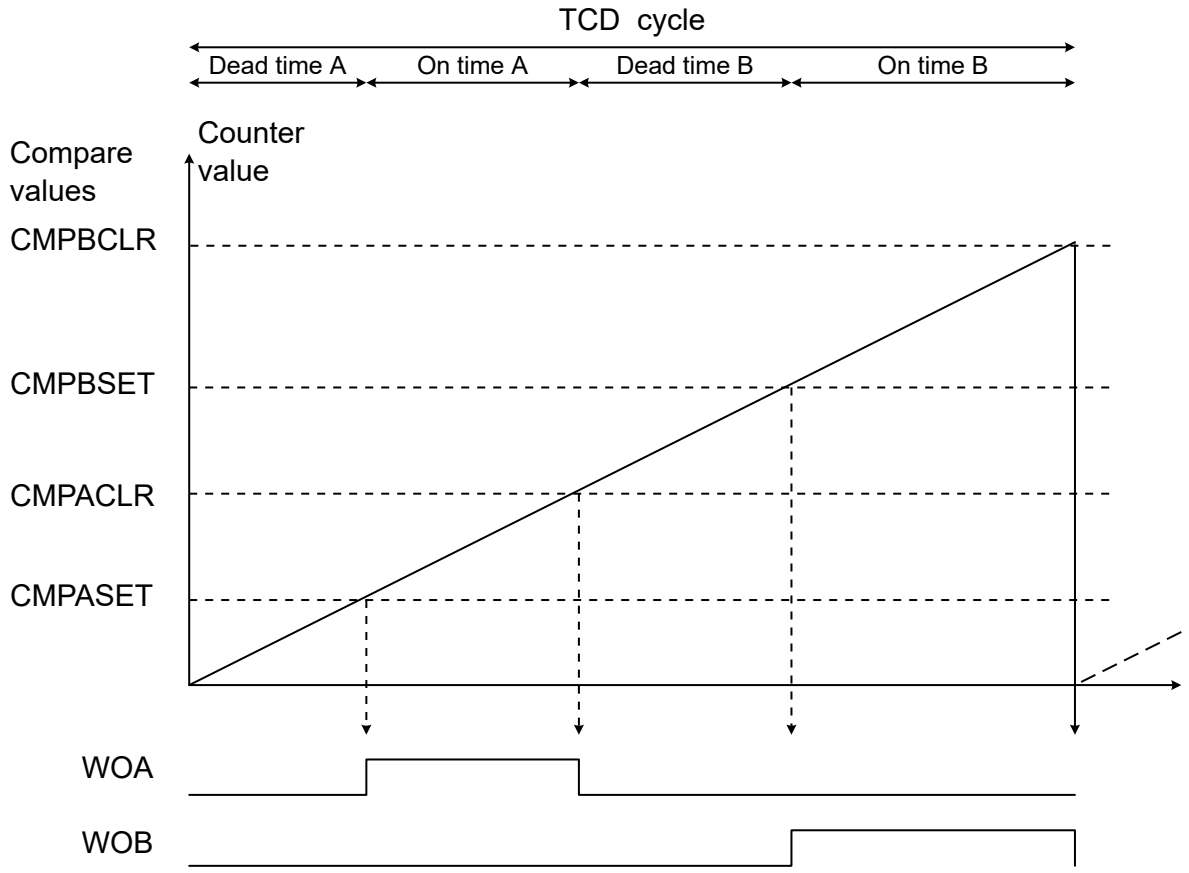
The Compare A Set (CMPASET), Compare A Clear (CMPACLR), Compare B Set (CMPBSET) and Compare B Clear (CMPBCLR) compare values define when each state ends and the next begins.

#### 25.3.3.2.1 One Ramp Mode

In One Ramp mode, the TCD counter counts up until it reaches the CMPBCLR value. Then, the TCD cycle is completed, and the counter restarts from `0x000`, beginning a new TCD cycle. The TCD cycle period is:

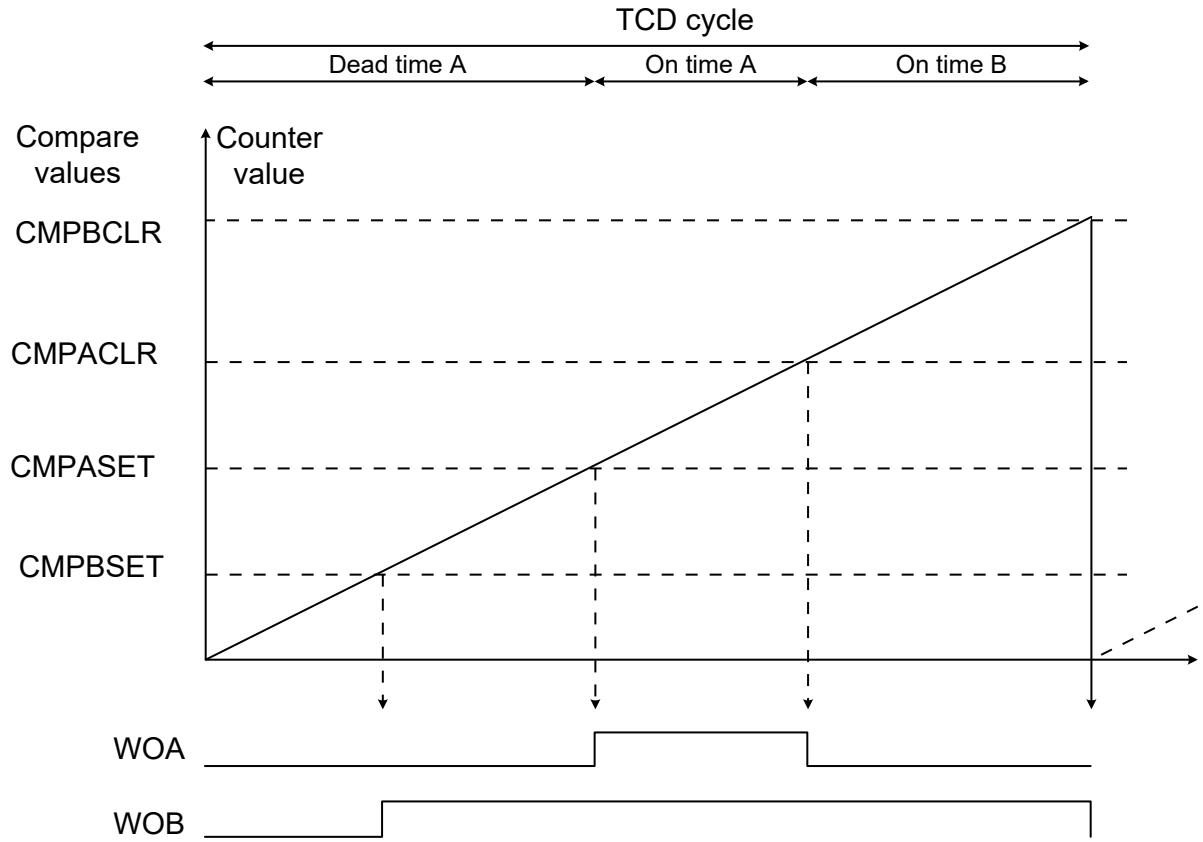
$$T_{\text{TCD_cycle}} = \frac{(\text{CMPBCLR} + 1)}{f_{\text{CLK_TCD_CNT}}}$$

**Figure 25-3. One Ramp Mode**




In the figure above, *CMPASET* < *CMPACLR* < *CMPBSET* < *CMPBCLR*. In One Ramp mode, this is required to avoid overlapping outputs during the on time. The figure below is an example where *CMPBSET* < *CMPASET* < *CMPACLR* < *CMPBCLR*, which has overlapping outputs during the on time.

**Figure 25-4. One Ramp Mode with *CMPBSET* < *CMPASET***



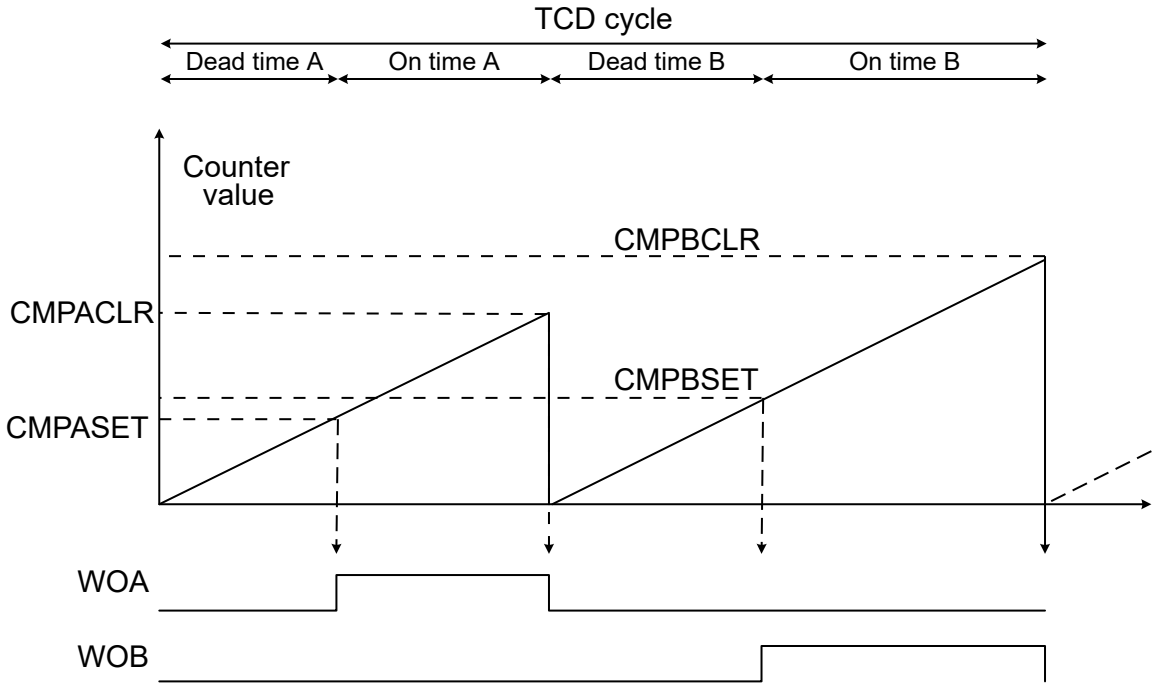

A match with CMPBCLR will always result in all outputs being cleared. If any of the other compare values are bigger than CMPBCLR, their associated effect will never occur. If the CMPACLR is smaller than the CMPASET value, the clear value will not have any effect.

#### 25.3.3.2.2 Two Ramp Mode

In Two Ramp mode, the TCD counter counts up until it reaches the CMPACLR value, then it resets and counts up until it reaches the CMPBCLR value. Then, the TCD cycle is completed, and the counter restarts from `0x000`, beginning a new TCD cycle. The TCD cycle period is given by:

$$T_{\text{TCD_cycle}} = \frac{(\text{CMPACLR} + 1 + \text{CMPBCLR} + 1)}{f_{\text{CLK_TCD_CNT}}}$$

**Figure 25-5. Two Ramp Mode**




In the figure above, *CMPASET* < *CMPACLR* and *CMPBSET* < *CMPBCLR*. This causes the outputs to go high. There are no restrictions on the CMPASET and CMPACLR compared to the CMPBSET and CMPBCLR values.

In Two Ramp mode, it is not possible to get overlapping outputs without using the override feature. Even if *CMPASET/CMPBSET* > *CMPACLR/CMPBCLR*, the counter resets at CMPACLR/CMPBCLR and will never reach CMPASET/CMPBSET.

#### 25.3.3.2.3 Four Ramp Mode

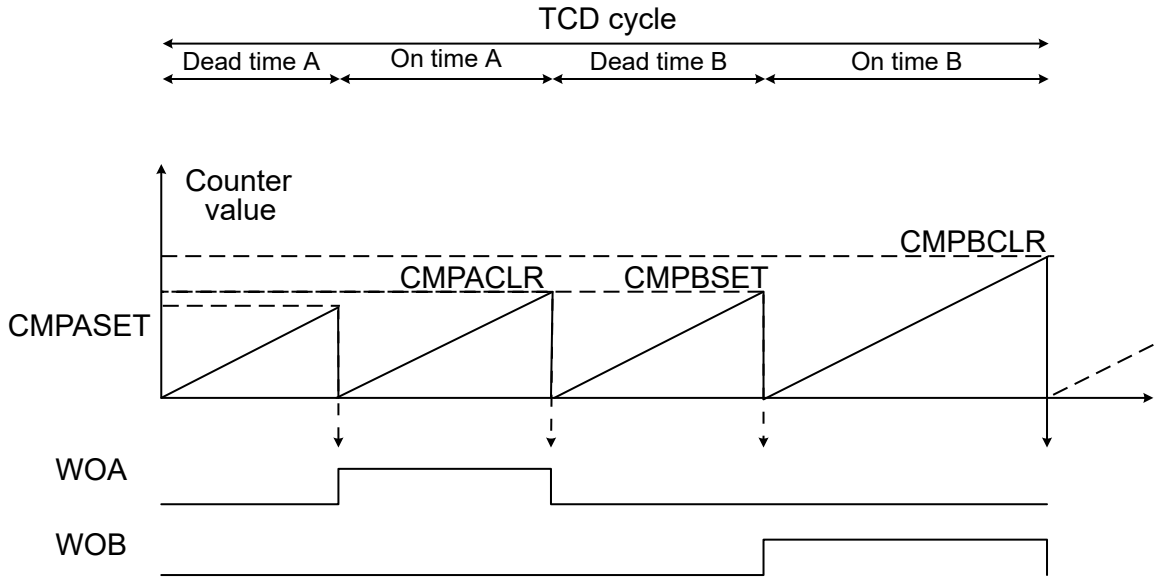
In Four Ramp mode, the TCD cycle follows this pattern:

1. A TCD cycle begins with the TCD counter counting up from zero until it reaches the CMPASET value, and resets to zero.
2. The counter counts up until it reaches the CMPACLR value, and resets to zero.
3. The counter counts up until it reaches the CMPBSET value, and resets to zero.
4. The counter counts up until it reaches the CMPBCLR value, and ends the TCD cycle by resetting to zero.

The TCD cycle period is given by:

$$T_{\mathrm{TCD_cycle}} = \frac{(\mathrm{CMPASET} + 1) + (\mathrm{CMPACLR} + 1) + (\mathrm{CMPBSET} + 1) + (\mathrm{CMPBCLR} + 1)}{f_{\mathrm{CLK_TCD_CNT}}}$$

**Figure 25-6. Four Ramp Mode**




There are no restrictions regarding the compare values, because there are no dependencies between them.

In Four Ramp mode, it is not possible to get overlapping outputs without using the override feature.

#### 25.3.3.2.4 Dual Slope Mode

In Dual Slope mode, a TCD cycle consists of the TCD counter counting down from CMPBCLR value to zero, and up again to the CMPBCLR value. This gives a TCD cycle period:

$$T_{\text{TCD_cycle}} = \frac{2 \times (\text{CMPBCLR} + 1)}{f_{\text{CLK_TCD_CNT}}}$$

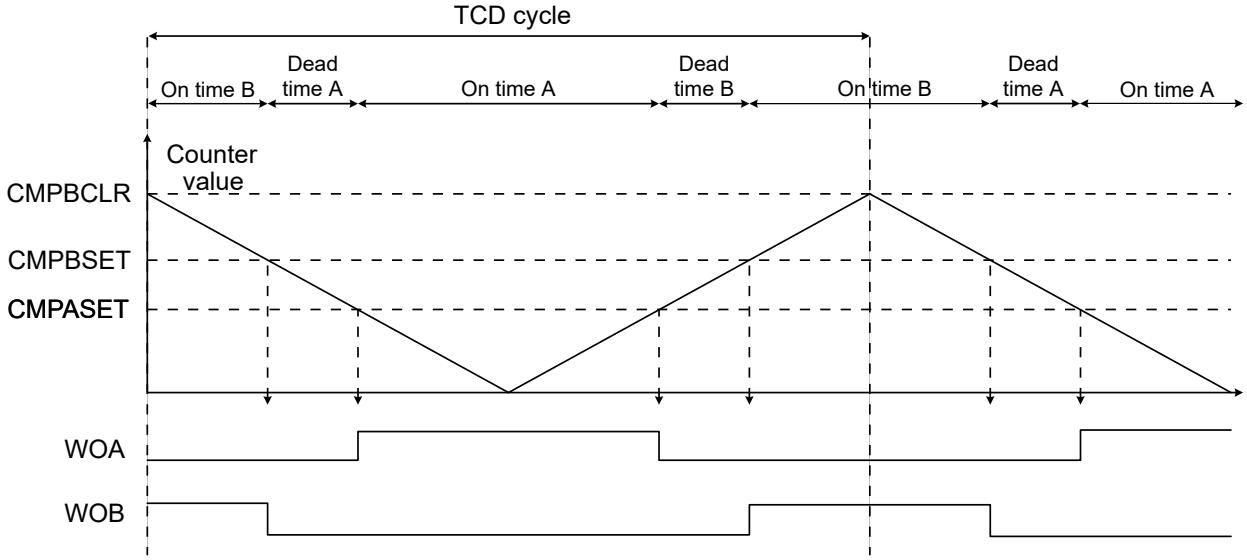
The WOA output is set when the TCD counter counts down and matches the CMPASET value. WOA is cleared when the TCD counter counts up and matches the CMPASET value.

The WOB output is set when the TCD counter counts up and matches the CMPBSET value. WOB is cleared when the TCD counter counts down and matches the CMPBSET value.

The outputs will overlap if *CMPASET* > *CMPBSET*.

CMPACLR is not used in Dual Slope mode. Writing a value to CMPACLR has no effect.

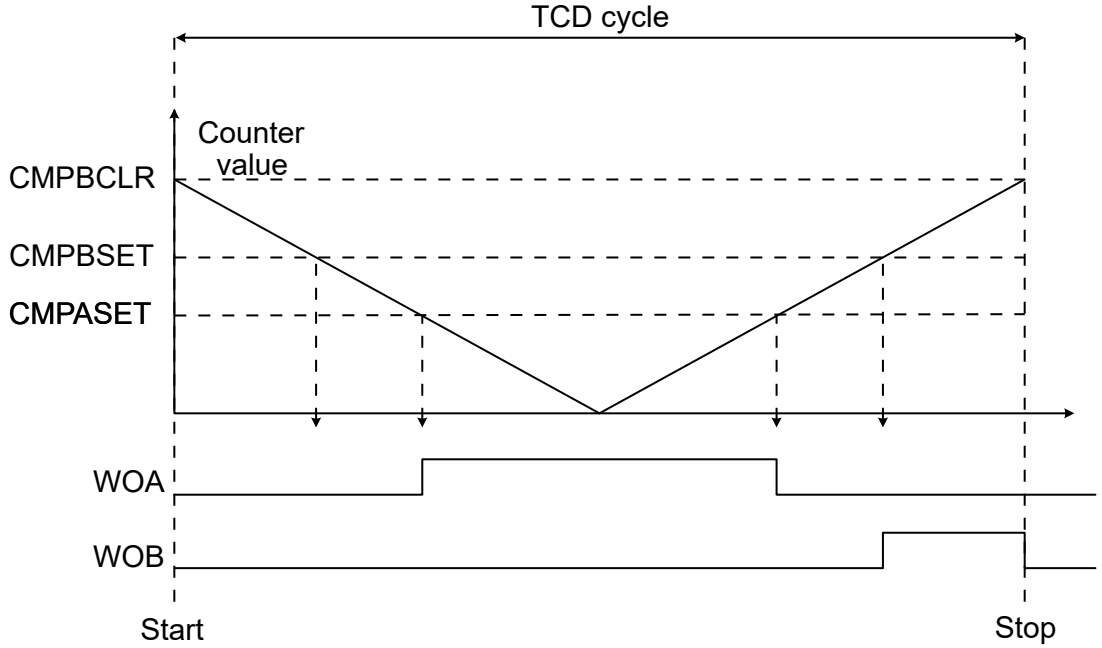
**Figure 25-7. Dual Slope Mode**



When starting the TCD in Dual Slope mode, the TCD counter starts at the CMPBCLR value and counts down. In the first cycle, the WOB will not be set until the TCD counter matches the CMPBSET value when counting up.

When the Disable at End of Cycle Strobe (DISEOC) bit in the Control E (TCDn.CTRLE) register is set, the TCD will automatically be disabled at the end of the TCD cycle.

**Figure 25-8. Dual Slope Mode Starting and Stopping**




#### 25.3.3.3 Disabling TCD

Disabling the TCD can be done in two different ways:

1. By writing a '0' to the ENABLE bit in the Control A (TCDn.CTRLA) register. This disables the TCD instantly when synchronized to the TCD core domain.
2. By writing a '1' to the Disable at End of Cycle Strobe (DISEOC) bit in the Control E (TCDn.CTRLE) register. This disables the TCD at the end of the TCD cycle.

#### 25.3.3.4 TCD Inputs

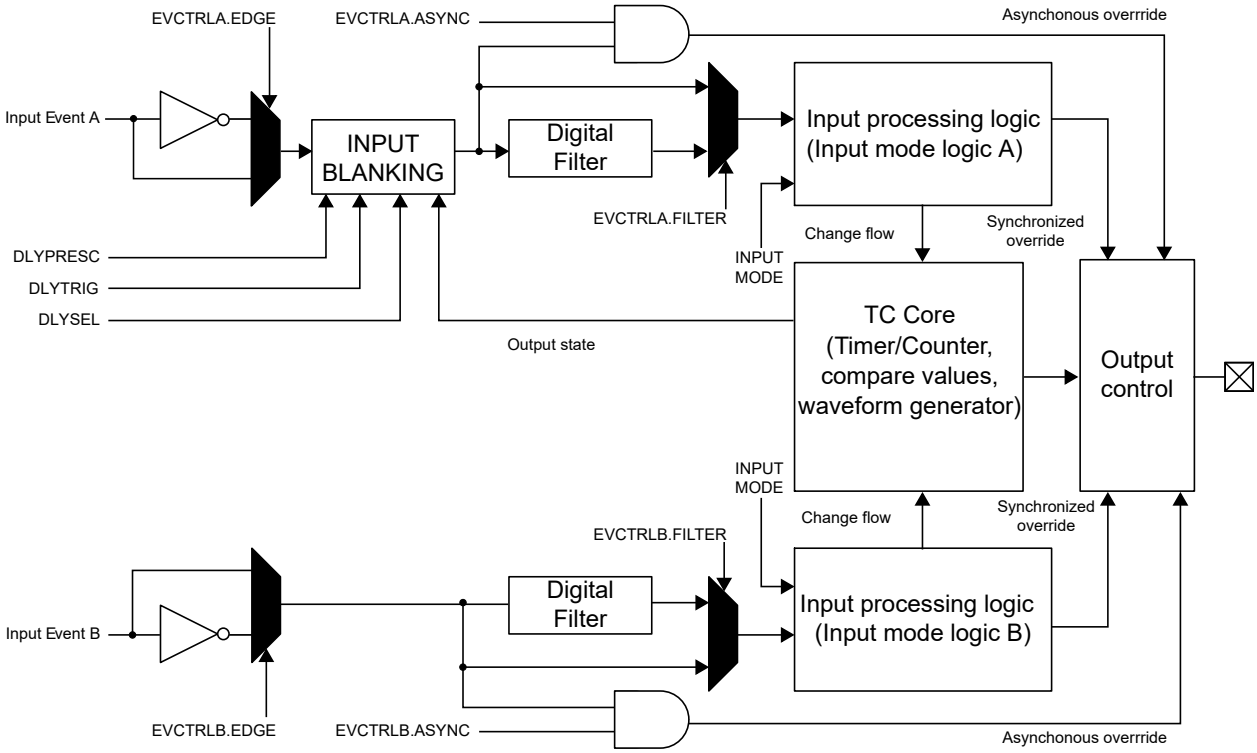
The TCD has two inputs connected to the Event System: input A and input B. Each input has a functionality connected to the corresponding output (WOA and WOB). This functionality is controlled by the Event Control (TCDn.EVCTRLA and TCDn.EVCTRLB) registers and the Input Control (TCDn.INPUTCTRLA and TCDn.INPUTCTRLB) registers.

To enable the input events, write a '1' to the Trigger Event Input Enable (TRIGEI) bit in the corresponding Event Control (TCDn.EVCTRLA or TCDn.EVCTRLB) register. The inputs will be used as a Fault detect by default, but they can also be used as a capture trigger. To enable a capture trigger, write a '1' to the ACTION bit in the corresponding Event Control (TCDn.EVCTRLA or TCDn.EVCTRLB) register. To disable Fault detect, the INPUTMODE bit field in the corresponding Input Control (TCDn.INPUTCTRLA or TCDn.INPUTCTRLB) register must be written to '0'.

There are ten different input modes for the Fault detection. The two inputs have the same functionality, except for input blanking which is only supported by input A. Input blanking is configured by the Delay Control (TCDn.DLYCTRL) register and the Delay Value (TCDn.DLYVAL) register.

The inputs are connected to the Event System. The connections between the event source and the TCD input must be configured in the Event System.

**Figure 25-9. TCD Input Overview**




There is a delay of two/three clock cycles on the TCD synchronizer clock between receiving the input event, processing it, and overriding the outputs. If using the asynchronous event detection, the outputs will override instantly outside the input processing.

#### 25.3.3.4.1 Input Blanking

Input blanking functionality masks out the input events for a programmable time in a selectable part of the TCD cycle. Input blanking can be used to mask out 'false' input events triggered right after changes on the outputs occur.

Input blanking can be enabled by configuring the Delay Select (DLYSEL) bit field in the Delay Control (TCDn.DLYCTRL) register. The trigger source is selected by the Delay Trigger (DLYTRIG) bit field in TCDn.DLYCTRL.

Input blanking uses the delay clock. After a trigger, a counter counts up until the Delay Value (DLYVAL) bit field in the Delay Value (TCDn.DLYVAL) register is reached. Afterward, input blanking is turned off. The TCD delay clock is a prescaled version of the synchronizer clock (CLK_TCD_SYNC). The division factor is set by the Delay Prescaler (DLYPRESC) bit field in the Delay Control (TCDn.DLYCTRL) register. The duration of the input blanking is given by:

$$t_{\mathrm{BLANK}} = \frac{\mathrm{DLYPRESC_division_factor} \times \mathrm{DLYVAL}}{f_{\mathrm{CLK_TCD_SYNC}}}$$

Input blanking uses the same logic as the programmable output event. For this reason, it is not possible to use both at the same time.

#### 25.3.3.4.2 Digital Filter

The digital filter for event input x is enabled by writing a '1' to the FILTER bit in the corresponding Event Control (TCDn.EVCTRLA or TCDn.EVCTRLB) register. When the digital filter is enabled, any pulse lasting less than four counter clock cycles will be filtered out. Any change on the incoming event will, therefore, take four counter clock cycles before it affects the input processing logic.

#### 25.3.3.4.3 Asynchronous Event Detection

To enable asynchronous event detection on an input event, the Event Configuration (CFG) bit field in the corresponding Event Control (TCDn.EVCTRLA or TCDn.EVCTRLB) register must be configured accordingly.

The asynchronous event detection makes it possible to asynchronously override the output when the input event occurs. What the input event will do depends on the input mode. The outputs have direct override while the counter flow will be changed when the event is synchronized to the synchronizer clock (CLK_TCD_SYNC).

It is not possible to use asynchronous event detection and digital filter at the same time.

#### 25.3.3.4.4 Software Commands

The following table displays the commands for the TCD module.

**Table 25-4. Software Commands**

| Trigger | Software Command |
|---|---|
| The SYNCEOC bit in the TCDn.CTRLE register | Update the double-buffered registers at the end of the TCD cycle |
| The SYNC bit in the TCDn.CTRLE register | Update the double-buffered registers |
| The RESTART bit in the TCDn.CTRLE register | Restart the TCD counter |
| The SCAPTUREA bit in the TCDn.CTRLE register | Capture to Capture A (TCDn.CAPTUREAL/H) register |
| The SCAPTUREB bit in the TCDn.CTRLE register | Capture to Capture B (TCDn.CAPTUREBL/H) register |

#### 25.3.3.4.5 Input Modes

The user can select between ten input modes. The selection is done by writing to the Input Mode (INPUTMODE) bit field in the Input Control (TCDn.INPUTCTRLA and TCDn.INPUTCTRLB) registers.

***Input Modes Validity***

Not all input modes work in all Waveform Generation modes. The table below shows the Waveform Generation modes in which the different input modes are valid.

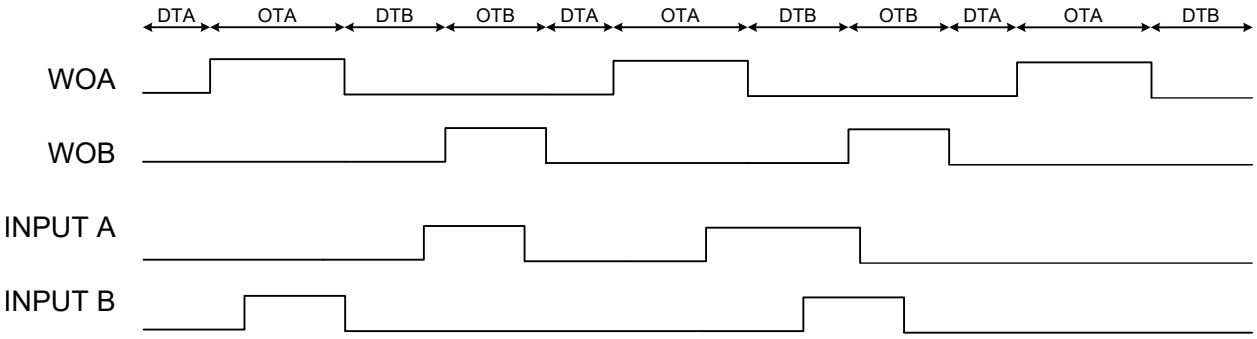
**Table 25-5. Input Modes Validity**

| INPUTMODE | One Ramp Mode | Two Ramp Mode | Four Ramp Mode | Dual Slope Mode |
|---|---|---|---|---|
| 0 | Valid | Valid | Valid | Valid |
| 1 | Valid | Valid | Valid | Do not use |
| 2 | Do not use | Valid | Valid | Do not use |
| 3 | Do not use | Valid | Valid | Do not use |
| 4 | Valid | Valid | Valid | Valid |
| 5 | Do not use | Valid | Valid | Do not use |
| 6 | Do not use | Valid | Valid | Do not use |
| 7 | Valid | Valid | Valid | Valid |
| 8 | Valid | Valid | Valid | Do not use |
| 9 | Valid | Valid | Valid | Do not use |
| 10 | Valid | Valid | Valid | Do not use |

***Input Mode 0: Input Has No Action***

In Input mode 0, the inputs do not affect the outputs, but they can still trigger captures and interrupts if enabled.

**Figure 25-10. Input Mode 0**

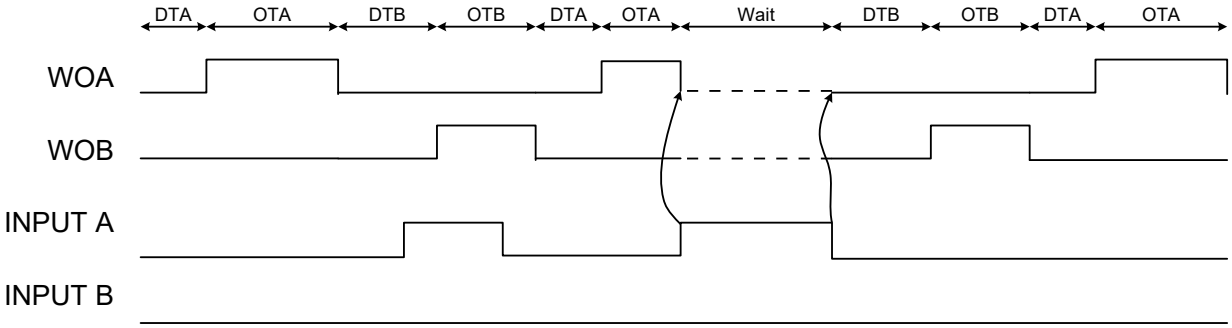



***Input Mode 1: Stop Output, Jump to Opposite Compare Cycle, and Wait***

An input event in Input mode 1 will stop the output signal, jump to the opposite dead time, and wait until the input event goes low before the TCD counter continues.

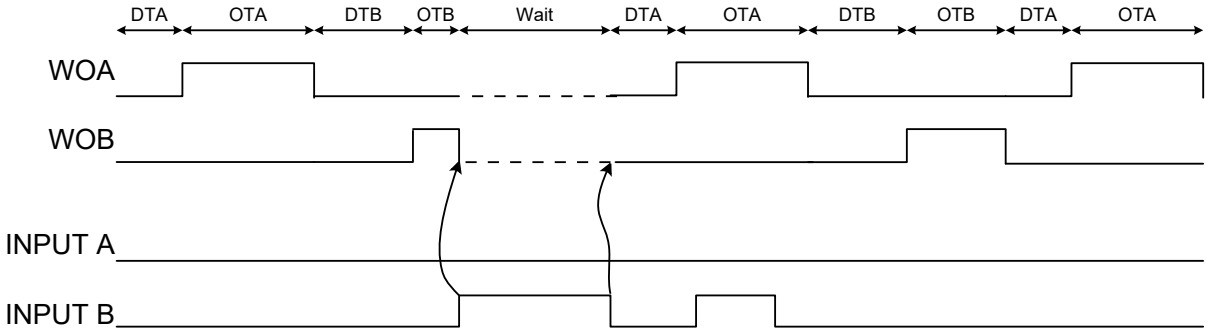
If Input mode 1 is used on input A, an event will only have an effect if the TCD is in dead time A or on time A, and it will only affect the WOA output. When the event is done, the TCD counter starts at dead time B.

**Figure 25-11. Input Mode 1 on Input A**




If Input mode 1 is used on input B, an event will only have an effect if the TCD is in dead time B or on time B, and it will only affect the WOB output. When the event is done, the TCD counter starts at dead time A.

**Figure 25-12. Input Mode 1 on Input B**

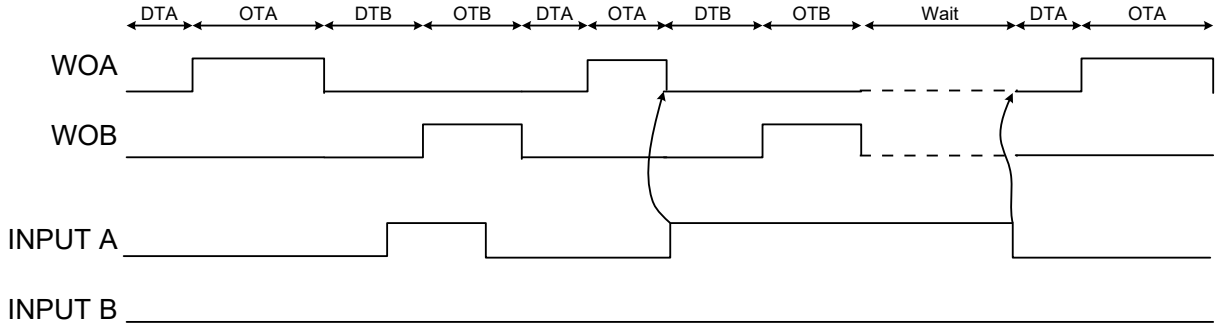



***Input Mode 2: Stop Output, Execute Opposite Compare Cycle, and Wait***

An input event in Input mode 2 will stop the output signal, execute to the opposite dead time and on time, and then wait until the input event goes low before the TCD counter continues. If the input is done before the opposite dead time and on time have finished, there will be no waiting, but the opposite dead time and on time will continue.

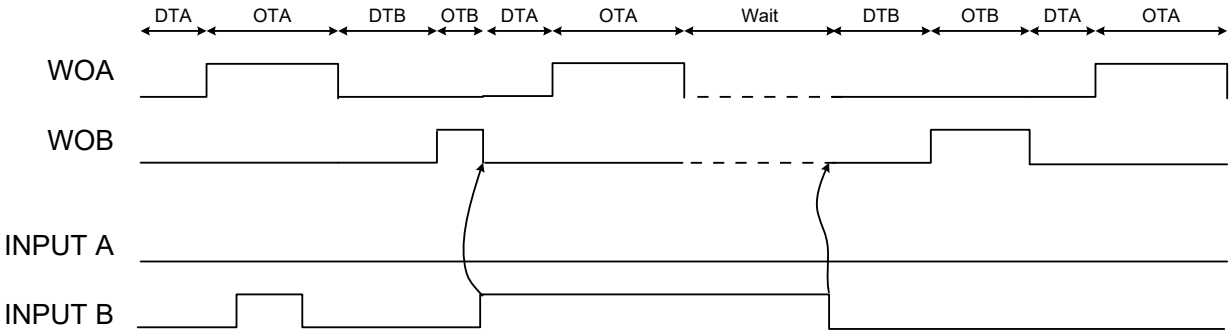
If Input mode 2 is used on input A, an event will only have an effect if the TCD is in dead time A or on time A, and will only affect the WOA output.

**Figure 25-13. Input Mode 2 on Input A**




If Input mode 2 is used on input B, an event will only have an effect if the TCD is in dead time B or on time B, and it will only affect the WOB output.

**Figure 25-14. Input Mode 2 on Input B**

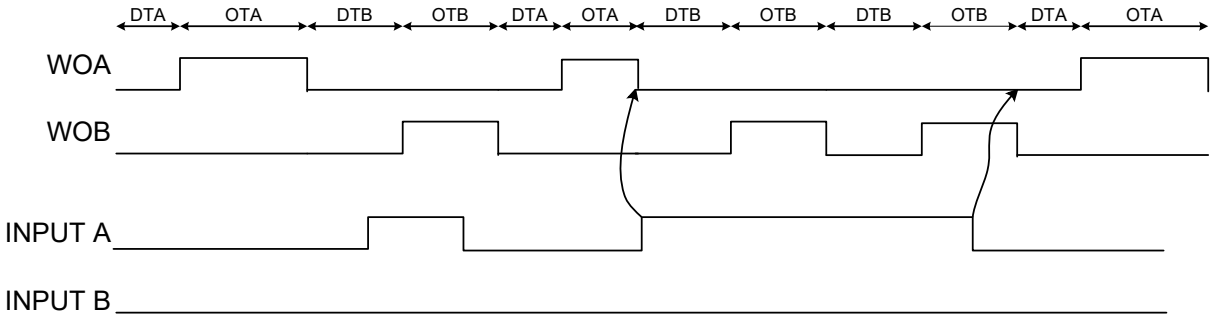



***Input Mode 3: Stop Output, Execute Opposite Compare Cycle while Fault Active***

An input event in Input mode 3 will stop the output signal and start executing the opposite dead time and on time repetitively, as long as the Fault/input is active. When the input is released, the ongoing dead time and/or on time will finish, and then the normal flow will start.

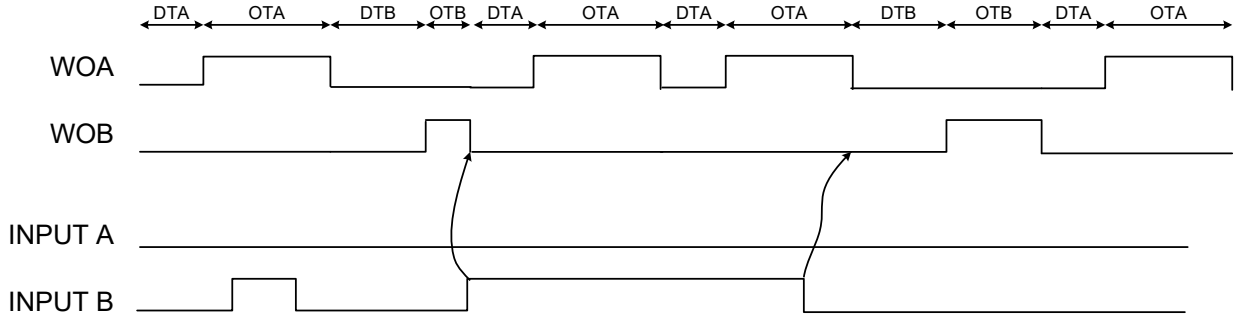
If Input mode 3 is used on input A, an event will only have an effect if the TCD is in dead time A or on time A.

**Figure 25-15. Input Mode 3 on Input A**




If Input mode 3 is used on input B, an event will only have an effect if the TCD is in dead time B or on time B.

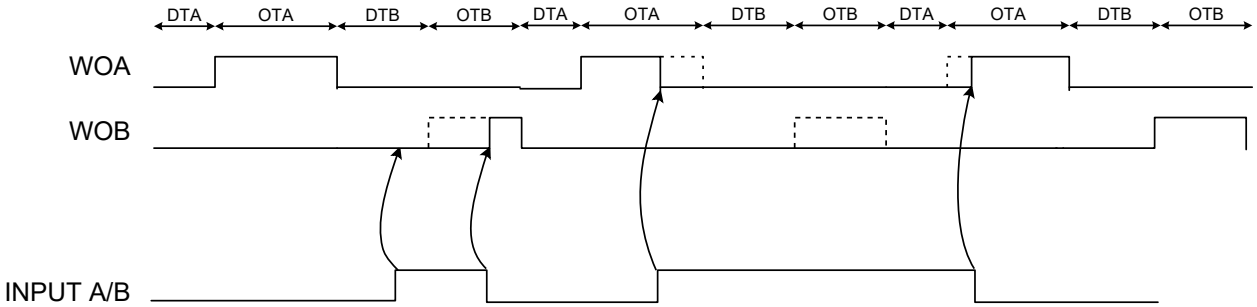
**Figure 25-16. Input Mode 3 on Input B**




***Input Mode 4: Stop all Outputs, Maintain Frequency***

When Input mode 4 is used, both input A and input B will give the same functionality. An input event will deactivate the outputs as long as the event is active. The TCD counter will not be affected by events in this input mode.

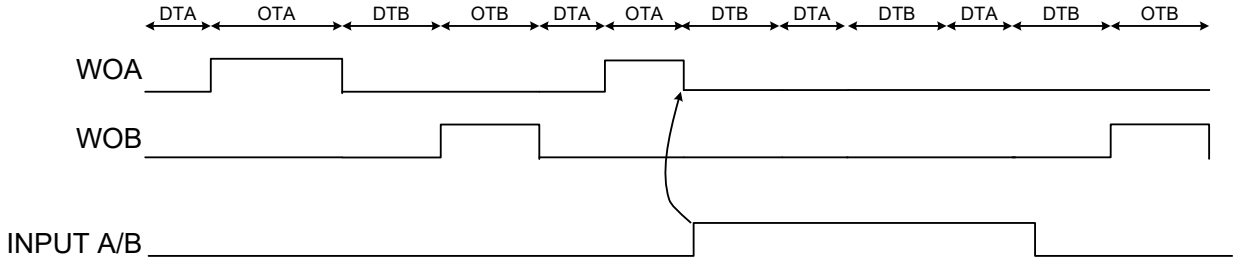
**Figure 25-17. Input Mode 4**




***Input Mode 5: Stop all Outputs, Execute Dead Time while Fault Active***

When Input mode 5 is used, both input A and input B give the same functionality. The input event stops the outputs and starts on the opposite dead time if it occurs during an on time. If the event occurs during dead time, the dead time will continue until the next on time is scheduled to start. Though, if the input is still active, the cycle will continue with the other dead time. As long as the input event is active, alternating dead times will occur. When the input event stops, the ongoing dead time will finish, and the next on time will continue in the normal flow.

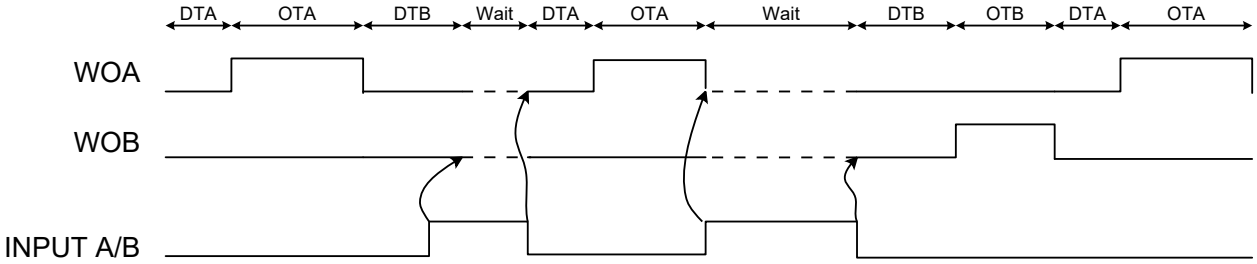
**Figure 25-18. Input Mode 5**




***Input Mode 6: Stop All Outputs, Jump to Next Compare Cycle, and Wait***

When Input mode 6 is used, both input A and input B will give the same functionality. The input event stops the outputs and jumps to the opposite dead time if it occurs during an on time. If the event occurs during dead time, the dead time will continue until the next on time is scheduled to start. As long as the input event is active, the TCD counter will wait. When the input event stops, the next dead time will start, and normal flow will continue.

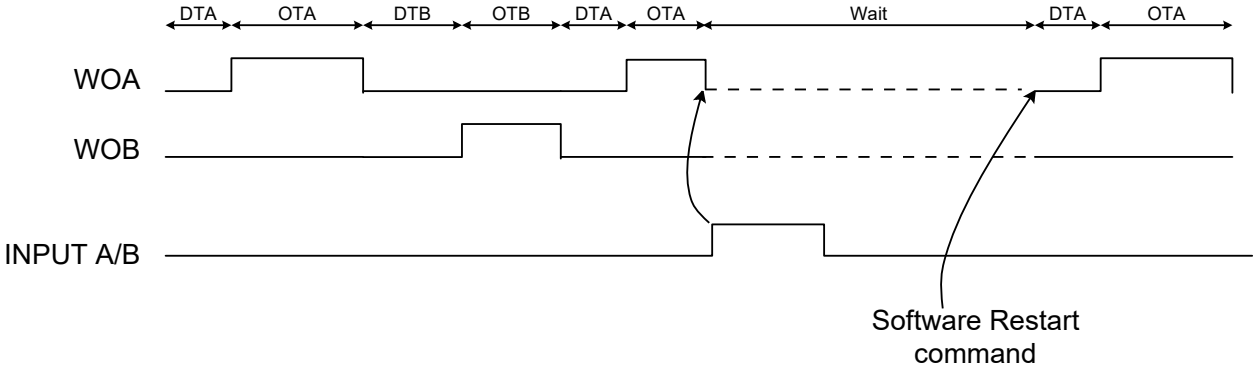
**Figure 25-19. Input Mode 6**




***Input Mode 7: Stop all Outputs, Wait for Software Action***

When Input mode 7 is used, both input A and input B will give the same functionality. The input events stop the outputs and the TCD counter. It will be stopped until a Restart command is given. If the input event is still high when the Restart command (RESTART bit in TCDn.CTRLE register) is given, it will stop again. When the TCD counter restarts, it will always start on dead time A.

**Figure 25-20. Input Mode 7**

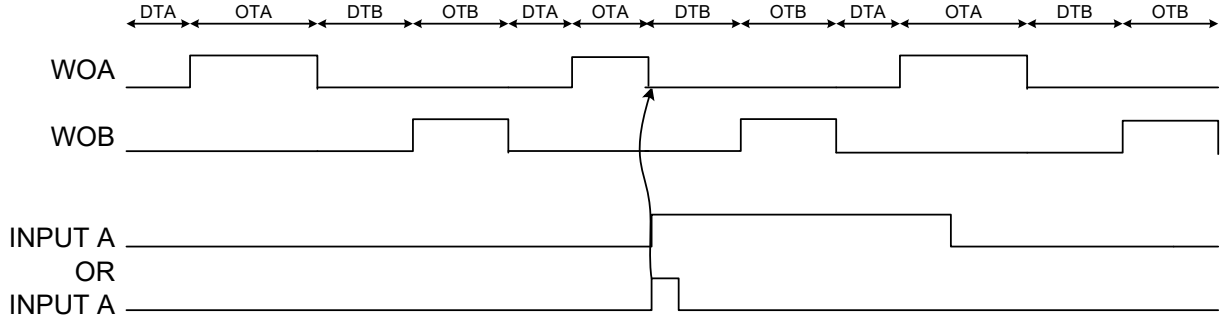



***Input Mode 8: Stop Output on Edge, Jump to Next Compare Cycle***

In Input mode 8, a positive edge on the input event while the corresponding output is ON will cause the output to stop and the TCD counter to jump to the opposite dead time.

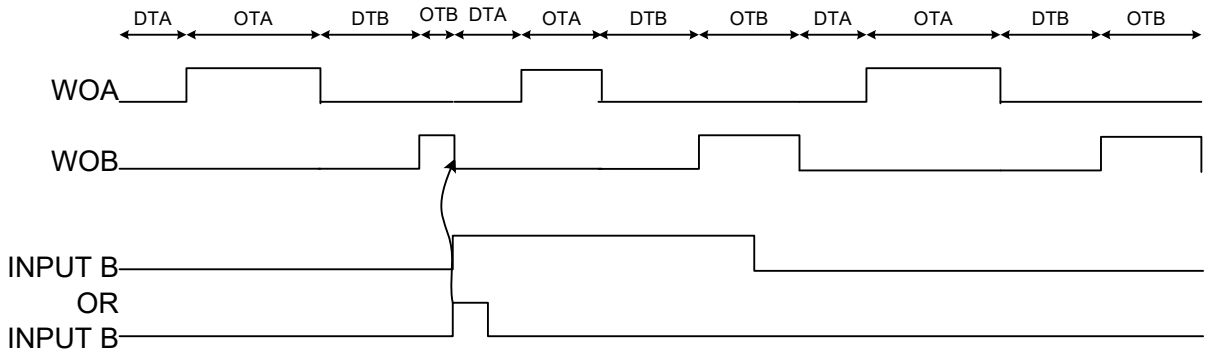
If Input mode 8 is used on input A and a positive edge on the input event occurs while in on time A, the TCD counter jumps to dead time B.

**Figure 25-21. Input Mode 8 on Input A**




If Input mode 8 is used on input B and a positive edge on the input event occurs while in on time B, the TCD counter jumps to dead time A.

**Figure 25-22. Input Mode 8 on Input B**

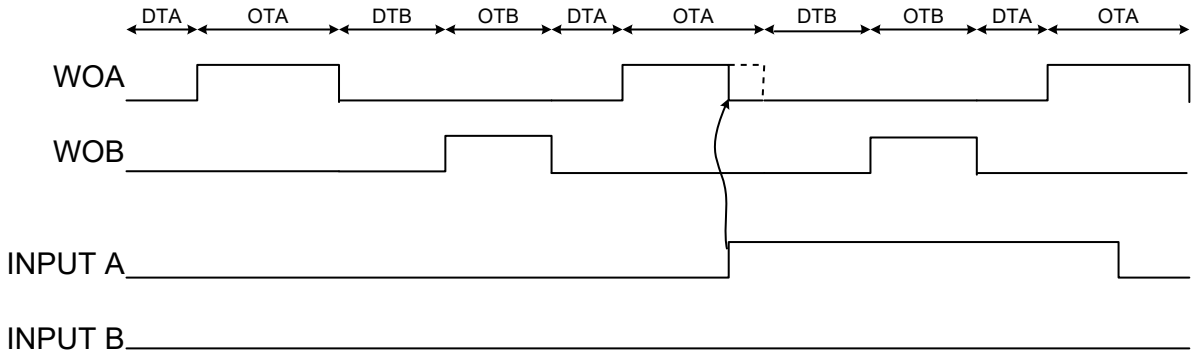



***Input Mode 9: Stop Output on Edge, Maintain Frequency***

In Input mode 9, a positive edge on the input event while the corresponding output is ON will cause the output to stop during the rest of the on time. The TCD counter will not be affected by the event, only the output.

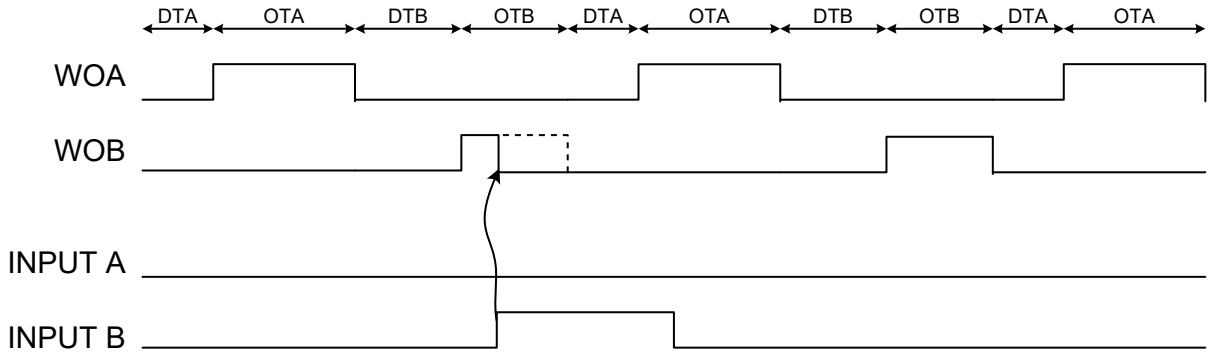
If Input mode 9 is used on input A and a positive edge on the input event occurs while in on time A, the output will be OFF for the rest of the on time.

**Figure 25-23. Input Mode 9 on Input A**




If Input mode 9 is used on input B and a positive edge on the input event occurs while in on time B, the output will be OFF for the rest of the on time.

**Figure 25-24. Input Mode 9 on Input B**

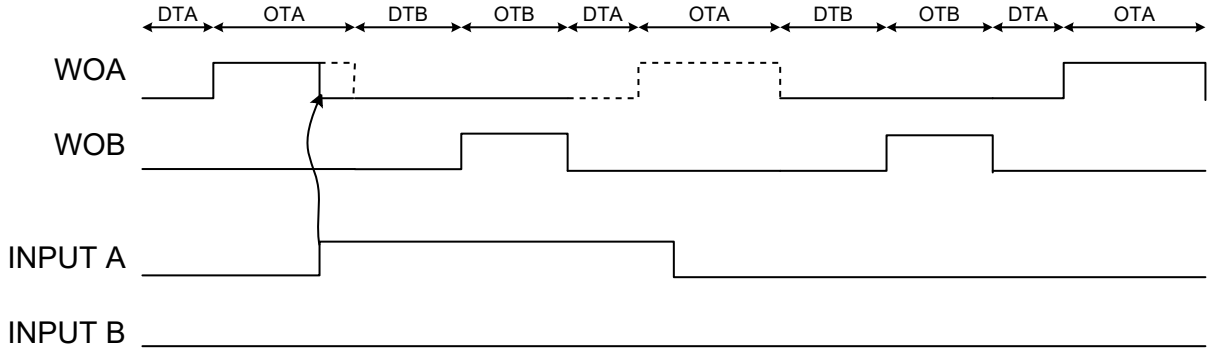



***Input Mode 10: Stop Output at Level, Maintain Frequency***

In Input mode 10, the input event will cause the corresponding output to stop, as long as the input is active. If the input goes low while there must have been an on time on the corresponding output, the output will be deactivated for the rest of the on time. The TCD counter is not affected by the event, only the output.

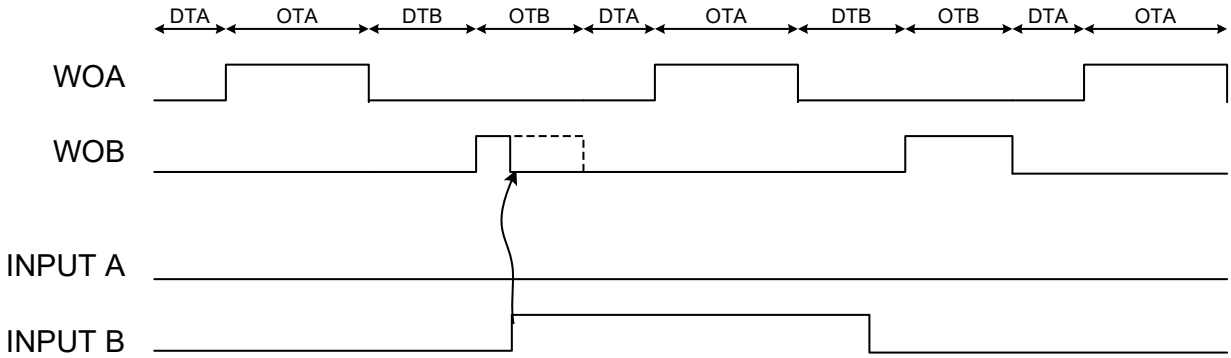
If Input mode 10 is used on input A and an input event occurs, the WOA will be OFF as long as the event lasts. If released during an on time, it will be OFF for the rest of the on time.

**Figure 25-25. Input Mode 10 on Input A**




If Input mode 10 is used on input B and an input event occurs, the WOB will be OFF as long as the event lasts. If released during an on time, it will be OFF for the rest of the on time.

**Figure 25-26. Input Mode 10 on Input B**




***Input Mode Summary***

Table 25-6 summarizes the conditions, as illustrated in the timing diagrams of the preceding sections.

**Table 25-6. Input Mode Summary**

| INPUTMODE | Trigger → Output Affected | Fault On/Active | Fault Release/Inactive |
|---|---|---|---|
| 0 | - | No action. | No action. |
| 1 | Input A→WOA | End the current on time and wait. | Start with dead time for the other compare. |
| | Input B→WOB | | |
| 2 | Input A→WOA | End the current on time, execute the other compare cycle and wait. | Start with dead time for the current compare. |
| | Input B→WOB | | |
| 3 | Input A→WOA | Execute the current on time, then execute the other compare cycle repetitively. | Re-enable the current compare cycle. |
| | Input B→WOB | | |
| 4 | Input A→{WOA, WOB} | Deactivate the outputs. | |
| | Input B→{WOA, WOB} | | |
| 5 | Input A→{WOA, WOB} | Execute dead time only. | |
| | Input B→{WOA, WOB} | | |
| 6 | Input A→{WOA, WOB} | End on time and wait. | Start with dead time for the other compare. |
| | Input B→{WOA, WOB} | | |

.....continued

| INPUTMODE | Trigger → Output Affected | Fault On/Active | Fault Release/Inactive |
|---|---|---|---|
| 7 | Input A→{WOA, WOB} | End on time and wait for software action. | Start with dead time for the current compare. |
| | Input B→{WOA, WOB} | | |
| 8 | Input A→WOA | End the current on time and continue with the other off time. | |
| | Input B→WOB | | |
| 9 | Input A→WOA | Block the current on time and continue the sequence. | |
| | Input B→WOB | | |
| 10 | Input A→WOA | Deactivate on time until the end of the sequence while the trigger is active. | |
| | Input B→WOB | | |
| other | - | - | - |

**Note:** When using different modes on each event input, take into consideration possible conflicts, keeping in mind that TCD has a single counter, to avoid unexpected results.

#### 25.3.3.5 Dithering

If it is not possible to achieve the desired frequency because of the prescaler/period selection limitations, dithering can be used to approximate the desired frequency and reduce the waveform drift.

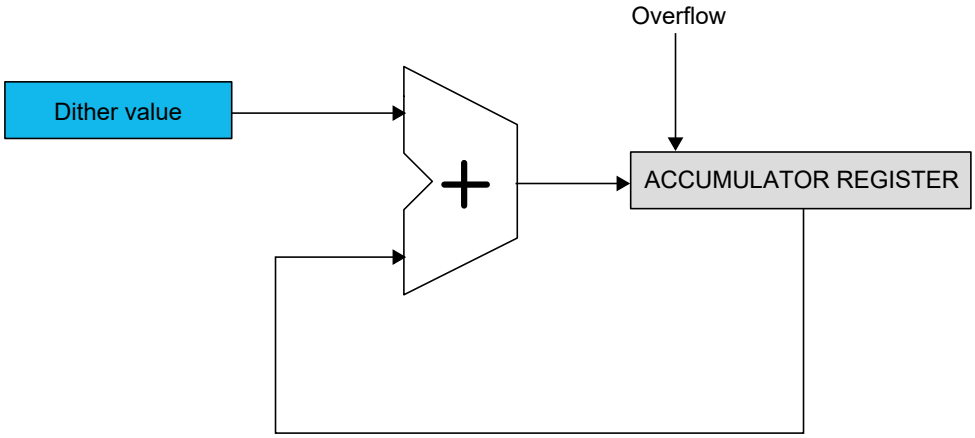
The dither accumulates the fractional error of the counter clock for each cycle. When the fractional error overflows, an additional clock cycle is added to the selected part of the TCD cycle.

**Example 25-1. Generate 75 kHz from a 10 MHz Clock**

If the timer clock frequency is 10 MHz, it will give the timer a resolution of 100 ns. The desired output frequency is 75 kHz, which means a period of 13333 ns. This period cannot be achieved with a 100 ns resolution as it would require 133.33 cycles. The output period can be set to either 133 cycles (75.188 kHz) or 134 cycles (74.626 kHz).

It is possible to change the period between the two frequencies manually in the firmware to get an average output frequency of 75 kHz (change every third period to 134 cycles). The dither can do this automatically by accumulating the error (0.33 cycles). The accumulator calculates when the accumulated error is larger than one clock cycle. When that happens, an additional cycle is added to the timer period.

**Figure 25-27. Dither Logic**




The user can select where in the TCD cycle the dither will be added by writing to the Dither Selection (DITHERSEL) bits in the Dither Control (TCDn.DITCTRL) register:

- On time B
- On time A and B
- Dead time B
- Dead time A and B

How much the dithering will affect the TCD cycle time depends on what Waveform Generation mode is used (see Table 25-7). Dithering is not supported in Dual Slope mode.

**Table 25-7. Mode-Dependent Dithering Additions to TCD Cycle**

| WAVEGEN | DITHERSEL in TCDn.DITCTRL | Additional TCD Clock Cycles to TCD Cycle |
|---|---|---|
| One Ramp mode | On time B | 1 |
| | On time A and B | 1 |
| | Dead time B | 0 |
| | Dead time A and B | 0 |
| Two Ramp mode | On time B | 1 |
| | On time A and B | 2 |
| | Dead time B | 0 |
| | Dead time A and B | 0 |
| Four Ramp mode | On time B | 1 |
| | On time A and B | 2 |
| | Dead time B | 1 |
| | Dead time A and B | 2 |
| Dual Slope mode | On time B | Not supported |
| | On time A and B | Not supported |
| | Dead time B | Not supported |
| | Dead time A and B | Not supported |

The differences in the number of TCD clock cycles added to the TCD cycle are caused by the different number of compare values used by the TCD cycle. For example, in One Ramp mode, only CMPBCLR affects the TCD cycle time.

For DITHERSEL configurations where no extra cycles are added to the TCD cycles, compensation is reached by shortening the following output state.

**Example 25-2. DITHERSEL in One Ramp Mode**

In One Ramp mode with DITHERSEL selecting dead time B, the dead time B will be increased by one cycle when dither overflow occurs, reducing on time B by one cycle.

#### 25.3.3.6 TCD Counter Capture

The TCD counter is asynchronous to the peripheral clock, so it is not possible to read out the counter value directly. It is possible to capture the TCD counter value, synchronized to the I/O clock domain, in two ways:

- Capture value on input events
- Software capture

The capture logic contains two separate capture blocks, CAPTUREA and CAPTUREB, that can capture and synchronize the TCD counter value to the I/O clock domain. CAPTUREA/B can be triggered by input event A/B or by software.

The capture values can be obtained by reading first TCDn.CAPTUREAL/TCDn.CAPTUREBL and then TCDn.CAPTUREAH/TCDn.CAPTUREBH registers.

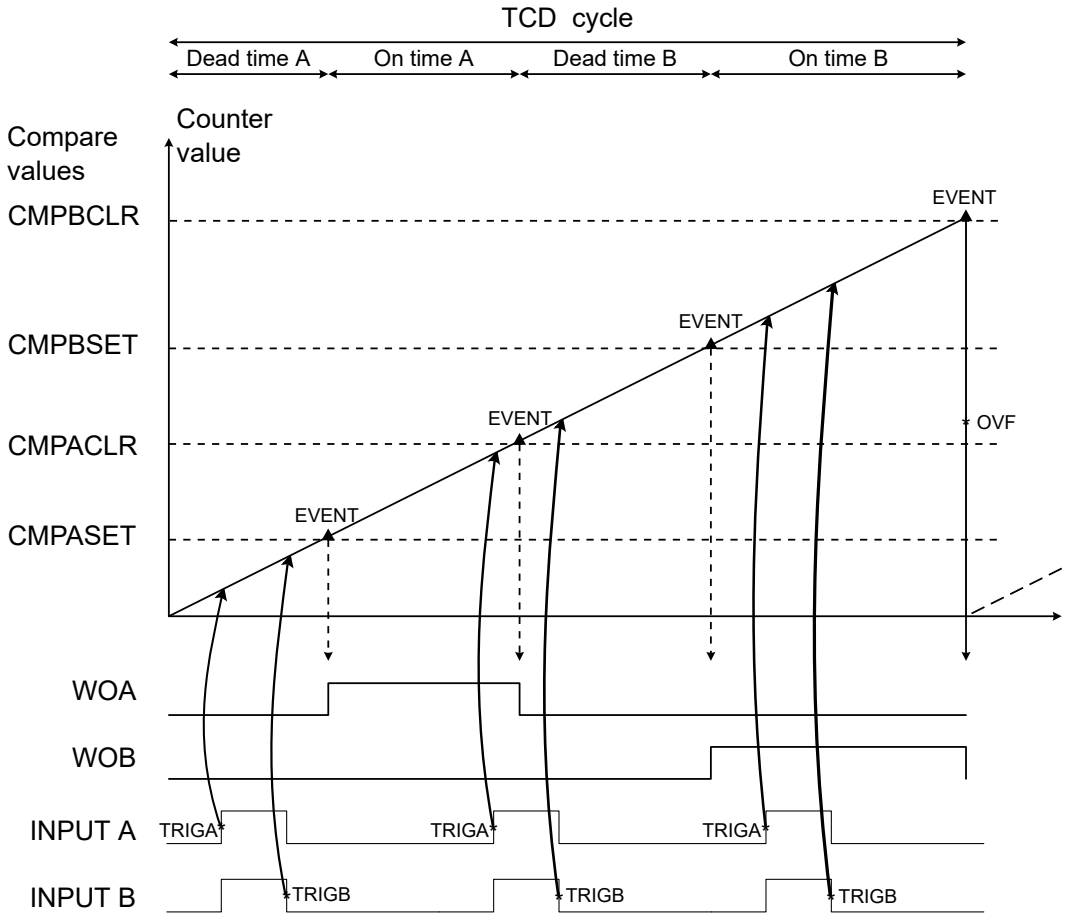
**Captures Triggered by Input Events**

To enable the capture on an input event, write a '1' to the ACTION bit in the respective Event Control (TCDn.EVCTRLA or TCDn.EVCTRLB) register when configuring an event input.

When a capture has occurred, the TRIGA/B flag is raised in the Interrupt Flags (TCDn.INTFLAGS) register. The corresponding TRIGA/B interrupt can be enabled by writing a '1' to the respective Trigger Interrupt Enable (TRIGA or TRIGB) bit in the Interrupt Control (TCDn.INTCTRL) register. By polling TRIGA or TRIGB in TCDn.INTFLAGS, the user knows that a CAPTURE value is available, and can read out the value by reading first the TCDn.CAPTUREAL or TCDn.CAPTUREBL register and then the TCDn.CAPTUREAH or TCDn.CAPTUREBH register.

**Example 25-3. PWM Capture**

To perform a PWM capture, connect both event A and event B to the same asynchronous event channel that contains the PWM signal. To get information on the PWM signal, configure one event input to capture the rising edge of the signal. Configure the other event input to capture the falling edge of the signal.




**Note:**

▲ Event trigger

* Interrupt trigger

**Capture Triggered by Software**

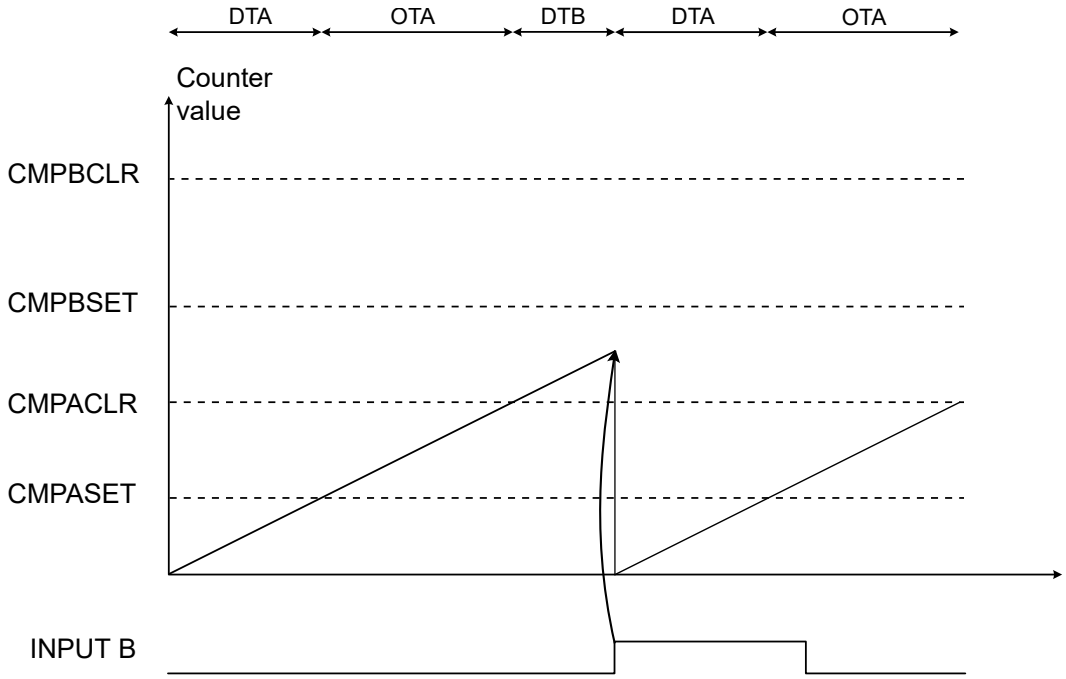
The software can capture the TCD value by writing a '1' to the respective Software Capture A/B Strobe (SCAPTUREx) bit in the Control E (TCDn.CTRLE) register. When this command is executed and the Command Ready (CMDRDY) bit in the Status (TCDn.STATUS) register reads '1' again, the CAPTUREA/B value is available. It can now be read by reading first the TCDn.CAPTUREAL or TCDn.CAPTUREBL register and then the TCDn.CAPTUREAH or TCDn.CAPTUREBH register.

**Using Capture Together with Input Modes**

The capture functionality can be used together with input modes. The same event will then both capture the counter value and trigger a change in the counter flow, depending on the input mode selected.

**Example 25-4. Reset One Ramp Mode by Input Event Capture**

In One Ramp mode, the counter can be reset by an input event capture. To achieve this, use input event B and write `0x08` to the INPUTMODE bit field in the Input Control B (TCDn.INPUTCTRLB) register.




### 25.3.3.7 Output Control

The outputs are configured by writing to the Fault Control (TCDn.FAULTCTRL) register.

The Compare x Enable (CMPxEN) bits in TCDn.FAULTCTRL enable the different outputs. The CMPx bits in TCDn.FAULTCTRL set the output values when a Fault is triggered.

The TCD itself generates two different outputs, WOA and WOB. The two additional outputs, WOC and WOD, can be configured by software to be connected to either WOA or WOB by writing the Compare C/D Output Select (CMPCSEL and CMPDSEL) bits in the Control C (TCDn.CTRLC) register.

The user can override the outputs based on the TCD counter state by writing a '1' to the Compare Output Value Override (CMPOVR) bit in the Control C (TCDn.CTRLC) register. The user can then select the output values in the different dead and on times by writing to the Compare Value (CMPAVAL and CMPBVAL) bit fields in the Control D (TCDn.CTRLD) register.

When used in One Ramp mode, WOA will only use the setup for dead time A (DTA) and on time A (OTA) to set the output. WOB will only use dead time B (DTB) and on time B (OTB) values to set the output.

When using the override feature together with Faults detection (input modes), the CMPA (and CMPC/D if WOC/D equals WOA) bit in TCDn.FAULTCTRL must be equal to CMPAVAL[0] and [2] in CTRL. If not, the first cycle after a

Fault is detected can have the wrong polarity on the outputs. The same applies to CMPB in the TCDn.FAULTCTRL (and CMPC/D if WOC/D equals WOB) bit, which must be equal to CMPBVAL[0] and [2] in TCDn.CTRLD.

Due to the asynchronous nature of the TCD and that input events can immediately affect the output signal, there is a risk of nanosecond spikes occurring on the output without any load on the pin. The case occurs in any input mode different from '0' and when an input event is triggering. The spike value will always be in the direction of the CMPx values given by the TCDn.FAULTCTRL register.

### 25.3.4 Events

The TCD can generate the events described in the following table:

**Table 25-8. Event Generators in TCD**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| TCDn | CMPBCLR | The counter matches CMPBCLR | Pulse | CLK_TCD | One CLK_TCD_CNT period |
| | CMPASET | The counter matches CMPASET | | | |
| | CMPBSET | The counter matches CMPBSET | | | |
| | PROGEV | Programmable event output(1) | | | One CLK_TCD_SYNC period |

**Note:**

1. The user can select the trigger and all the compare matches (including CMPACLR). Also, it is possible to delay the output event from 0 to 255 TCD delay cycles.

The three events based on the counter match directly generate event strobes that last for one clock cycle on the TCD counter clock. The programmable output event generates an event strobe that lasts for one clock cycle on the TCD synchronizer clock.

The TCD can receive the events described in the following table:

**Table 25-9. Event Users and Available Event Actions in TCD**

| User Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| TCDn | Input A/ Input B | Stop the output, jump to the opposite compare cycle and wait. | Level | Both |
| | | Stop the output, execute the opposite compare cycle and wait. | | |
| | | Stop the output, execute the opposite compare cycle while the Fault is active. | | |
| | | Stop all outputs, maintain the frequency. | | |
| | | Stop all outputs, execute dead time while the Fault is active. | | |
| | | Stop all outputs, jump to the next compare cycle and wait. | | |
| | | Stop all outputs, wait for software action. | | |
| | | Stop the output on the edge, jump to the next compare cycle. | Edge | |
| | | Stop the output on the edge, maintain the frequency. | | |
| | | Stop the output at level, maintain the frequency. | Level | |

Input A and Input B are TCD event users that detect and act upon the input events. Additional information about input events and how to configure them can be found in the 25.3.3.4 TCD Inputs section. Refer to the Event System (EVSYS) section for more details regarding event types and Event System configuration.

#### 25.3.4.1 Programmable Output Events

The Programmable Output Event (PROGEV) uses the same logic as the input blanking for trigger selection and delay. Therefore, it is not possible to configure the functionalities independently. If the input blanking functionality is used, the output event cannot be delayed, and the trigger used for input blanking will also be used for the output event.

PROGEV is configured in the TCDn.DLYCTRL and TCDn.DLYVAL registers. It is possible to delay the output event by 0 to 255 TCD delay clock cycles. The delayed output event functionality uses the TCD delay clock and counts until the DLYVAL value is reached before the trigger is sent out as an event. The TCD delay clock is a prescaled version of the TCD synchronizer clock (CLK_TCD_SYNC), and the division factor is set by the DLYPRESC bits in the TCDn.DLYCTRL register. The output event will be delayed by the TCD clock period x DLYPRESC division factor x DLYVAL.

### 25.3.5 Interrupts

**Table 25-10. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| OVF | Overflow interrupt | The TCD finishes one TCD cycle. |
| TRIG | Trigger interrupt | • TRIGA: On event input A<br>• TRIGB: On event input B |

When an interrupt condition occurs, the corresponding interrupt flag is set in the Interrupt Flags (TCDn.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the Interrupt Control (TCDn.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

When several interrupt request conditions are supported by an interrupt vector, the interrupt requests are ORed together into one combined interrupt request to the interrupt controller. The user must read the peripheral's INTFLAGS register to determine which of the interrupt conditions are present.

### 25.3.6 Sleep Mode Operation

The TCD operates in Idle sleep mode and is stopped when entering Standby and Power-Down sleep modes.

### 25.3.7 Debug Operation

Halting the CPU in Debugging mode will halt the normal operation of the peripheral. This peripheral can be forced to operate with the CPU halted by writing a '1' to the Debug Run (DBGRUN) bit in the Debug Control (TCDn.DBGCTRL) register.

When the Fault Detection (FAULTDET) bit in TCDn.DBGCTRL is written to '1', and the CPU is halted in Debug mode, an event/Fault is created on both input event channels. These events/Faults last as long as the break and can serve as a safeguard in Debug mode, for example, by forcing external components off.

If the peripheral is configured to require periodic service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

### 25.3.8 Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). To write to these registers, a certain key must first be written to the CPU.CCP register, followed by a write access to the protected bits within four CPU instructions.

Attempting to write to a protected register without following the appropriate CCP unlock sequence leaves the protected register unchanged.

The following registers are under CCP:

**Table 25-11. Registers under Configuration Change Protection in TCD**

| Register | Key |
|---|---|
| TCDn.FAULTCTRL | IOREG |

## 25.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | CLKSEL[1:0] | | CNTPRES[1:0] | | SYNCPRES[1:0] | | ENABLE |
| 0x01 | CTRLB | 7:0 | | | | | | | WGMODE[1:0] | |
| 0x02 | CTRLC | 7:0 | CMPDSEL | CMPCSEL | | | FIFTY | | AUPDATE | CMPOVR |
| 0x03 | CTRLD | 7:0 | CMPBVAL[3:0] | | | | CMPAVAL[3:0] | | | |
| 0x04 | CTRLE | 7:0 | DISEOC | | | SCAPTUREB | SCAPTUREA | RESTART | SYNC | SYNCEOC |
| 0x05 ... 0x07 | Reserved | | | | | | | | | |
| 0x08 | EVCTRLA | 7:0 | CFG[1:0] | | | EDGE | | ACTION | | TRIGEI |
| 0x09 | EVCTRLB | 7:0 | CFG[1:0] | | | EDGE | | ACTION | | TRIGEI |
| 0x0A ... 0x0B | Reserved | | | | | | | | | |
| 0x0C | INTCTRL | 7:0 | | | | | TRIGB | TRIGA | | OVF |
| 0x0D | INTFLAGS | 7:0 | | | | | TRIGB | TRIGA | | OVF |
| 0x0E | STATUS | 7:0 | PWMACTB | PWMACTA | | | | | CMDRDY | ENRDY |
| 0x0F | Reserved | | | | | | | | | |
| 0x10 | INPUTCTRLA | 7:0 | | | | | INPUTMODE[3:0] | | | |
| 0x11 | INPUTCTRLB | 7:0 | | | | | INPUTMODE[3:0] | | | |
| 0x12 | FAULTCTRL | 7:0 | CMPDxEN | CMPCxEN | CMPBxEN | CMPAxEN | CMPDx | CMPCx | CMPBx | CMPAx |
| 0x13 | Reserved | | | | | | | | | |
| 0x14 | DLYCTRL | 7:0 | | | DLYPRESC[1:0] | | DLYTRIG[1:0] | | DLYSEL[1:0] | |
| 0x15 | DLYVAL | 7:0 | DLYVAL[7:0] | | | | | | | |
| 0x16 ... 0x17 | Reserved | | | | | | | | | |
| 0x18 | DITCTRL | 7:0 | | | | | | | DITHERSEL[1:0] | |
| 0x19 | DITVAL | 7:0 | | | | | DITHER[3:0] | | | |
| 0x1A ... 0x1D | Reserved | | | | | | | | | |
| 0x1E | DBGCTRL | 7:0 | | | | | | FAULTDET | | DBGRUN |
| 0x1F ... 0x21 | Reserved | | | | | | | | | |
| 0x22 | CAPTUREA | 7:0 | CAPTUREA[7:0] | | | | | | | |
| | | 15:8 | | | | | CAPTUREA[11:8] | | | |
| 0x24 | CAPTUREB | 7:0 | CAPTUREB[7:0] | | | | | | | |
| | | 15:8 | | | | | CAPTUREB[11:8] | | | |
| 0x26 ... 0x27 | Reserved | | | | | | | | | |
| 0x28 | CMPASET | 7:0 | CMPASET[7:0] | | | | | | | |
| | | 15:8 | | | | | CMPASET[11:8] | | | |
| 0x2A | CMPACLR | 7:0 | CMPACLR[7:0] | | | | | | | |
| | | 15:8 | | | | | CMPACLR[11:8] | | | |
| 0x2C | CMPBSET | 7:0 | CMPBSET[7:0] | | | | | | | |
| | | 15:8 | | | | | CMPBSET[11:8] | | | |
| 0x2E | CMPBCLR | 7:0 | CMPBCLR[7:0] | | | | | | | |
| | | 15:8 | | | | | CMPBCLR[11:8] | | | |

## 25.5 Register Description

### 25.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** Enable-protected

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | CLKSEL[1:0] | | CNTPRES[1:0] | | SYNCPRES[1:0] | | ENABLE |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 6:5 – CLKSEL[1:0]** Clock Select
The Clock Select bits select the clock source of the TCD clock.

| Value | Name | Description |
|---|---|---|
| 0x0 | OSCHF | Internal High-Frequency Oscillator |
| 0x1 | PLL | PLL |
| 0x2 | EXTCLK | External Clock or external crystal oscillator |
| 0x3 | CLKPER | Main clock after prescaler (CLK_PER) |

**Bits 4:3 – CNTPRES[1:0]** Counter Prescaler
The Counter Prescaler bits select the division factor of the TCD counter clock.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV1 | Division factor 1 |
| 0x1 | DIV4 | Division factor 4 |
| 0x2 | DIV32 | Division factor 32 |
| 0x3 | - | Reserved |

**Bits 2:1 – SYNCPRES[1:0]** Synchronization Prescaler
The Synchronization Prescaler bits select the division factor of the TCD clock.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV1 | Division factor 1 |
| 0x1 | DIV2 | Division factor 2 |
| 0x2 | DIV4 | Division factor 4 |
| 0x3 | DIV8 | Division factor 8 |

**Bit 0 – ENABLE** Enable
When writing to this bit, it will automatically be synchronized to the TCD clock domain.
This bit can be changed as long as the synchronization of this bit is not ongoing. See the Enable Ready (ENRDY) bit in the Status (TCDn.STATUS) register.
This bit is not enable-protected.

| Value | Name | Description |
|---|---|---|
| 0 | NO | The TCD is disabled. |
| 1 | YES | The TCD is enabled and running. |

### 25.5.2 Control B

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

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

**Bits 1:0 – WGMODE[1:0]** Waveform Generation Mode
These bits select the waveform generation.

| Value | Name | Description |
|---|---|---|
| 0x0 | ONERAMP | One Ramp mode |
| 0x1 | TWORAMP | Two Ramp mode |
| 0x2 | FOURRAMP | Four Ramp mode |
| 0x3 | DS | Dual Slope mode |

### 25.5.3 Control C

**Name:** CTRLC
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMPDSEL | CMPCSEL | | | FIFTY | | AUPDATE | CMPOVR |
| Access | R/W | R/W | | | R/W | | R/W | R/W |
| Reset | 0 | 0 | | | 0 | | 0 | 0 |

**Bit 7 – CMPDSEL** Compare D Output Select
This bit selects which waveform will be connected to output D.

| Value | Name | Description |
|---|---|---|
| 0 | PWMA | Waveform A |
| 1 | PWMB | Waveform B |

**Bit 6 – CMPCSEL** Compare C Output Select
This bit selects which waveform will be connected to output C.

| Value | Name | Description |
|---|---|---|
| 0 | PWMA | Waveform A |
| 1 | PWMB | Waveform B |

**Bit 3 – FIFTY** Fifty Percent Waveform
If the two waveforms have identical characteristics, this bit can be written to '1'. This will cause any values written to the TCDn.CMPBSET/TCDn.CLR register to also be written to the TCDn.CMPASET/TCDn.CLR register.

**Bit 1 – AUPDATE** Automatically Update
If this bit is written to '1', synchronization at the end of the TCD cycle is automatically requested after the Compare B Clear High (TCDn.CMPBCLRH) register is written.
If the fifty percent waveform is enabled by setting the FIFTY bit in this register, writing the Compare A Clear High register will also request a synchronization at the end of the TCD cycle if the AUPDATE bit is set.

**Bit 0 – CMPOVR** Compare Output Value Override
When this bit is written to '1', default values of the Waveform Outputs A and B are overridden by the values written in the Compare x Value in Active state bit fields in the Control D register. See the 25.5.4 CTRLD register description for more details.

### 25.5.4 Control D

**Name:** CTRLD
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMPBVAL[3:0] | | | | CMPAVAL[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 0:3, 4:7 – CMPVAL** Compare x Value (in Active state)
These bits set the logical value of the PWMx signal for the corresponding states in the TCD cycle.
These settings are valid only if the Compare Output Value Override (CMPOVR) bit in the Control C (TCDn.CTRLC) register is written to '1'.

**Table 25-12. Two and Four Ramp Mode**

| CMPxVAL | DTA | OTA | DTB | OTB |
|---|---|---|---|---|
| PWMA | CMPAVAL[0] | CMPAVAL[1] | CMPAVAL[2] | CMPAVAL[3] |
| PWMB | CMPBVAL[0] | CMPBVAL[1] | CMPBVAL[2] | CMPBVAL[3] |

When used in One Ramp mode, WOA will only use the setup for dead time A (DTA) and on time A (OTA) to set the output. WOB will only use dead time B (DTB) and on time B (OTB) values to set the output.

**Table 25-13. One Ramp Mode**

| CMPxVAL | DTA | OTA | DTB | OTB |
|---|---|---|---|---|
| PWMA | CMPAVAL[1] | CMPAVAL[0] | - | - |
| PWMB | - | - | CMPBVAL[3] | CMPBVAL[2] |

### 25.5.5 Control E

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DISEOC | | | SCAPTUREB | SCAPTUREA | RESTART | SYNC | SYNCEOC |
| Access | R/W | | | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DISEOC** Disable at End of TCD Cycle Strobe
When this bit is written to '1', the TCD will automatically disable at the end of the TCD cycle.
Note that ENRDY in TCDn.STATUS will stay low until the TCD is disabled.
Writing to this bit has effect only if there is no ongoing synchronization of the ENABLE value in TCDn.CTRLA with the TCD domain. See also the ENRDY bit in TCDn.STATUS.

**Bit 4 – SCAPTUREB** Software Capture B Strobe
When this bit is written to '1', a software capture to the Capture B (TCDn.CAPTUREBL/H) register is triggered as soon as synchronization to the TCD clock domain occurs.
Writing to this bit has effect only if there is no ongoing synchronization of a command. See also the CMDRDY bit in TCDn.STATUS.

**Bit 3 – SCAPTUREA** Software Capture A Strobe
When this bit is written to '1', a software capture to the Capture A (TCDn.CAPTUREAL/H) register is triggered as soon as synchronization to the TCD clock domain occurs.
Writing to this bit has effect only if there is no ongoing synchronization of a command. See also the CMDRDY bit in TCDn.STATUS.

**Bit 2 – RESTART** Restart Strobe
When this bit is written to '1', a restart of the TCD counter is executed as soon as this bit is synchronized to the TCD domain.
Writing to this bit has effect only if there is no ongoing synchronization of a command. See also the CMDRDY bit in TCDn.STATUS.

**Bit 1 – SYNC** Synchronize Strobe
When this bit is written to '1', the double-buffered registers will be loaded to the TCD domain as soon as this bit is synchronized to the TCD domain.
Writing to this bit has effect only if there is no ongoing synchronization of a command. See also the CMDRDY bit in TCDn.STATUS.

**Bit 0 – SYNCEOC** Synchronize End of TCD Cycle Strobe
When this bit is written to '1', the double-buffered registers will be loaded to the TCD domain at the end of the next TCD cycle.
Writing to this bit has effect only if there is no ongoing synchronization of a command. See also the CMDRDY bit in TCDn.STATUS.

### 25.5.6 Event Control A

**Name:** EVCTRLA
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CFG[1:0] | | | EDGE | | ACTION | | TRIGEI |
| Access | R/W | R/W | | R/W | | R/W | | R/W |
| Reset | 0 | 0 | | 0 | | 0 | | 0 |

**Bits 7:6 – CFG[1:0]** Event Configuration
When the input capture noise canceler is activated (FILTERON), the event input is filtered. The filter function requires four successive equal valued samples of the trigger pin to change its output. The input capture is, therefore, delayed by four clock cycles when the noise canceler is enabled (FILTERON).
When the Asynchronous Event is enabled (ASYNCON), the event input will affect the output directly.

| Value | Name | Description |
|---|---|---|
| 0x0 | NEITHER | Neither filter nor asynchronous event is enabled. |
| 0x1 | FILTERON | Input capture noise cancellation filter enabled. |
| 0x2 | ASYNCON | Asynchronous event output qualification enabled. |
| other | - | Reserved. |

**Bit 4 – EDGE** Edge Selection
This bit is used to select the active edge or level for the event input.

| Value | Name | Description |
|---|---|---|
| 0 | FALL_LOW | The falling edge or low level of the event input triggers a Capture or Fault action. |
| 1 | RISE_HIGH | The rising edge or high level of the event input triggers a Capture or Fault action. |

**Bit 2 – ACTION** Event Action
This bit enables capturing on the event input. By default, the input will trigger a Fault, depending on the Input Control register's Input mode. It is also possible to trigger a capture on the event input.

| Value | Name | Description |
|---|---|---|
| 0 | FAULT | Event triggers a Fault. |
| 1 | CAPTURE | Event triggers a Fault and capture. |

**Bit 0 – TRIGEI** Trigger Event Input Enable
Writing this bit to '1' enables event as the trigger for input A.

### 25.5.7 Event Control B

**Name:** EVCTRLB
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CFG[1:0] | | | EDGE | | ACTION | | TRIGEI |
| Access | R/W | R/W | | R/W | | R/W | | R/W |
| Reset | 0 | 0 | | 0 | | 0 | | 0 |

**Bits 7:6 – CFG[1:0]** Event Configuration
When the input capture noise canceler is activated (FILTERON), the event input is filtered. The filter function requires four successive equal valued samples of the trigger pin to change its output. The input capture is, therefore, delayed by four clock cycles when the noise canceler is enabled (FILTERON).
When the Asynchronous Event is enabled (ASYNCON), the event input will affect the output directly.

| Value | Name | Description |
|---|---|---|
| 0x0 | NEITHER | Neither filter nor asynchronous event is enabled. |
| 0x1 | FILTERON | Input capture noise cancellation filter enabled. |
| 0x2 | ASYNCON | Asynchronous event output qualification enabled. |
| other | - | Reserved. |

**Bit 4 – EDGE** Edge Selection
This bit is used to select the active edge or level for the event input.

| Value | Name | Description |
|---|---|---|
| 0 | FALL_LOW | The falling edge or low level of the event input triggers a Capture or Fault action. |
| 1 | RISE_HIGH | The rising edge or high level of the event input triggers a Capture or Fault action. |

**Bit 2 – ACTION** Event Action
This bit enables capturing on the event input. By default, the input will trigger a Fault, depending on the Input Control register's Input mode. It is also possible to trigger a capture on the event input.

| Value | Name | Description |
|---|---|---|
| 0 | FAULT | Event triggers a Fault. |
| 1 | CAPTURE | Event triggers a Fault and capture. |

**Bit 0 – TRIGEI** Trigger Event Input Enable
Writing this bit to '1' enables event as a trigger for input B.

### 25.5.8 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x0C
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | TRIGB | TRIGA | | OVF |
| Access | | | | | R/W | R/W | | R/W |
| Reset | | | | | 0 | 0 | | 0 |

**Bit 3 – TRIGB** Trigger B Interrupt Enable
Writing this bit to '1' enables the interrupt when trigger input B is received.

**Bit 2 – TRIGA** Trigger A Interrupt Enable
Writing this bit to '1' enables the interrupt when trigger input A is received.

**Bit 0 – OVF** Counter Overflow
Writing this bit to '1' enables the restart-of-sequence interrupt or overflow interrupt.

### 25.5.9 Interrupt Flags

**Name:** INTFLAGS
**Offset:** 0x0D
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | TRIGB | TRIGA | | OVF |
| Access | | | | | R/W | R/W | | R/W |
| Reset | | | | | 0 | 0 | | 0 |

**Bit 3 – TRIGB** Trigger B Interrupt Flag
The Trigger B Interrupt (TRIGB) flag is set on a Trigger B or Capture B condition. The flag is cleared by writing a '1' to its bit location.

**Bit 2 – TRIGA** Trigger A Interrupt Flag
The Trigger A Interrupt (TRIGA) flag is set on a Trigger A or Capture A condition. The flag is cleared by writing a '1' to its bit location.

**Bit 0 – OVF** Overflow Interrupt Flag
The Overflow Flag (OVF) is set at the end of a TCD cycle. The flag is cleared by writing a '1' to its bit location.

### 25.5.10 Status

**Name:** STATUS
**Offset:** 0x0E
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWMACTB | PWMACTA | | | | | CMDRDY | ENRDY |
| Access | R/W | R/W | | | | | R | R |
| Reset | 0 | 0 | | | | | 0 | 0 |

**Bit 7 – PWMACTB** PWM Activity on B
This bit is set by hardware each time the WOB output toggles from '0' to '1' or from '1' to '0'.
This status bit must be cleared by software by writing a '1' to it before new PWM activity can be detected.

**Bit 6 – PWMACTA** PWM Activity on A
This bit is set by hardware each time the WOA output toggles from '0' to '1' or from '1' to '0'.
This status bit must be cleared by software by writing a '1' to it before new PWM activity can be detected.

**Bit 1 – CMDRDY** Command Ready
This status bit tells when a command is synced to the TCD domain and the system is ready to receive new commands.
The following actions clear the CMDRDY bit:

1. TCDn.CTRLE SYNCEOC strobe.
2. TCDn.CTRLE SYNC strobe.
3. TCDn.CTRLE RESTART strobe.
4. TCDn.CTRLE SCAPTUREA Capture A strobe.
5. TCDn.CTRLE SCAPTUREB Capture B strobe.
6. TCDn.CTRLC AUPDATE written to '1' and writing to the TCDn.CMPBCLRH register.

**Bit 0 – ENRDY** Enable Ready
This status bit tells when the ENABLE value in TCDn.CTRLA is synced to the TCD domain and is ready to be written to again.
The following actions clear the ENRDY bit:

1. Writing to the ENABLE bit in TCDn.CTRLA.
2. TCDn.CTRLE DISEOC strobe.
3. Going into BREAK in an On-Chip Debugging (OCD) session while the Debug Run (DBGCTRL) bit in TCDn.DBGCTRL is '0'.

### 25.5.11 Input Control A

**Name:** INPUTCTRLA
**Offset:** 0x10
**Reset:** 0x00
**Property:** -

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

**Bits 3:0 – INPUTMODE[3:0]** Input Mode

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | The input has no action. |
| 0x1 | JMPWAIT | Stop the output, jump to the opposite compare cycle, and wait. |
| 0x2 | EXECWAIT | Stop the output, execute the opposite compare cycle, and wait. |
| 0x3 | EXECFAULT | Stop the output, execute the opposite compare cycle while the Fault is active. |
| 0x4 | FREQ | Stop all outputs, maintain the frequency. |
| 0x5 | EXECDT | Stop all outputs, execute dead time while the Fault is active. |
| 0x6 | WAIT | Stop all outputs, jump to the next compare cycle, and wait. |
| 0x7 | WAITSW | Stop all outputs, wait for software action. |
| 0x8 | EDGETRIG | Stop the output on the edge, jump to the next compare cycle. |
| 0x9 | EDGETRIGFREQ | Stop the output on the edge, maintain the frequency. |
| 0xA | LVLTRIGFREQ | Stop the output at level, maintain the frequency. |

### 25.5.12 Input Control B

**Name:** INPUTCTRLB
**Offset:** 0x11
**Reset:** 0x00
**Property:** -

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

**Bits 3:0 – INPUTMODE[3:0]** Input Mode

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | The input has no action. |
| 0x1 | JMPWAIT | Stop the output, jump to the opposite compare cycle, and wait. |
| 0x2 | EXECWAIT | Stop the output, execute the opposite compare cycle, and wait. |
| 0x3 | EXECFAULT | Stop the output, execute the opposite compare cycle while the Fault is active. |
| 0x4 | FREQ | Stop all outputs, maintain the frequency. |
| 0x5 | EXECDT | Stop all outputs, execute dead time while the Fault is active. |
| 0x6 | WAIT | Stop all outputs, jump to the next compare cycle, and wait. |
| 0x7 | WAITSW | Stop all outputs, wait for software action. |
| 0x8 | EDGETRIG | Stop the output on the edge, jump to the next compare cycle. |
| 0x9 | EDGETRIGFREQ | Stop the output on the edge, maintain the frequency. |
| 0xA | LVLTRIGFREQ | Stop the output at level, maintain the frequency. |

### 25.5.13 Fault Control

**Name:** FAULTCTRL
**Offset:** 0x12
**Reset:** 0x00
**Property:** Configuration Change Protection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMPDxEN | CMPCxEN | CMPBxEN | CMPAxEN | CMPDx | CMPCx | CMPBx | CMPAx |
| Access | R/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 4, 5, 6, 7 – CMPxEN** Compare x Enable
These bits enable the waveform from compare as output on the pin.

**Bits 0, 1, 2, 3 – CMPx** Compare x Value
These bits set the default state of the compare waveform output.

### 25.5.14 Delay Control

**Name:** DLYCTRL
**Offset:** 0x14
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | DLYPRESC[1:0] | | DLYTRIG[1:0] | | DLYSEL[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 – DLYPRESC[1:0]** Delay Prescaler
These bits control the prescaler settings for the blanking or output event delay.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV1 | Prescaler division factor 1 |
| 0x1 | DIV2 | Prescaler division factor 2 |
| 0x2 | DIV4 | Prescaler division factor 4 |
| 0x3 | DIV8 | Prescaler division factor 8 |

**Bits 3:2 – DLYTRIG[1:0]** Delay Trigger
These bits control the trigger of the blanking or output event delay.

| Value | Name | Description |
|---|---|---|
| 0x0 | CMPASET | CMPASET triggers delay |
| 0x1 | CMPACLR | CMPACLR triggers delay |
| 0x2 | CMPBSET | CMPBSET triggers delay |
| 0x3 | CMPBCLR | CMPASET triggers delay (end of cycle) |

**Bits 1:0 – DLYSEL[1:0]** Delay Select
These bits control what function must be used by the delay trigger, the blanking or output event delay.

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | Delay functionality not used |
| 0x1 | INBLANK | Input blanking enabled |
| 0x2 | EVENT | Event delay enabled |
| 0x3 | - | Reserved |

### 25.5.15 Delay Value

**Name:** DLYVAL
**Offset:** 0x15
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DLYVAL[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 – DLYVAL[7:0]** Delay Value
These bits configure the blanking/output event delay time or event output synchronization delay in a number of prescaled TCD cycles.

### 25.5.16 Dither Control

**Name:** DITCTRL
**Offset:** 0x18
**Reset:** 0x00
**Property:** -

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

**Bits 1:0 – DITHERSEL[1:0]** Dither Select
This bit field selects which state of the TCD cycle will benefit from the dither function. See the 25.3.3.5 Dithering section.

| Value | Name | Description |
|---|---|---|
| 0x0 | ONTIMEB | On time ramp B |
| 0x1 | ONTIMEAB | On time ramp A and B |
| 0x2 | DEADTIMEB | Dead time ramp B |
| 0x3 | DEADTIMEAB | Dead time ramp A and B |

### 25.5.17 Dither Value

**Name:** DITVAL
**Offset:** 0x19
**Reset:** 0x00
**Property:** -

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

**Bits 3:0 – DITHER[3:0]** Dither Value
These bits configure the fractional adjustment of the on time or off time, according to the Dither Selection (DITHERSEL) bits in the Dither Control (TCDn.DITCTRL) register. The DITHER value is added to a 4-bit accumulator at the end of each TCD cycle. When the accumulator overflows, the frequency adjustment will occur.
The DITHER bits are double-buffered, so the new value is copied when an update condition occurs.

### 25.5.18 Debug Control

**Name:** DBGCTRL
**Offset:** 0x1E
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | FAULTDET | | DBGRUN |
| Access | | | | | | R/W | | R/W |
| Reset | | | | | | 0 | | 0 |

**Bit 2 – FAULTDET** Fault Detection
This bit defines how the peripheral behaves when stopped in Debug mode.

| Value | Name | Description |
|---|---|---|
| 0 | NONE | No Fault is generated if TCD is stopped in Debug mode. |
| 1 | FAULT | A Fault is generated, and both trigger flags are set, if TCD is halted in Debug mode. |

**Bit 0 – DBGRUN** Debug Run
When written to '1', the peripheral will continue operating in Debug mode when the CPU is halted.

| Value | Description |
|---|---|
| 0 | The peripheral is halted in Break Debug mode and ignores events. |
| 1 | The peripheral will continue to run in Break Debug mode when the CPU is halted. |

### 25.5.19 Capture A

**Name:** CAPTUREA
**Offset:** 0x22
**Reset:** 0x00
**Property:** -

The TCDn.CAPTUREAL and TCDn.CAPTUREAH register pair represents the 12-bit TCDn.CAPTUREA value.

For capture operation, these registers constitute the second buffer level and access point for the CPU. The TCDn.CAPTUREA registers are updated with the buffer value when an update condition occurs. The CAPTURE A register contains the TCD counter value when a trigger A or software capture A occurs.

The TCD counter value is synchronized to CAPTUREA by either software or an event.

The capture register is blocked for an update of new capture data until the higher byte of this register is read.

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CAPTUREA[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 11:0 – CAPTUREA[11:0]** Capture A Value

### 25.5.20 Capture B

**Name:** CAPTUREB
**Offset:** 0x24
**Reset:** 0x00
**Property:** -

The TCDn.CAPTUREBL and TCDn.CAPTUREBH register pair represents the 12-bit TCDn.CAPTUREB value.

For capture operation, these registers constitute the second buffer level and access point for the CPU. The TCDn.CAPTUREB registers are updated with the buffer value when an update condition occurs. The CAPTURE B register contains the TCD counter value when a trigger B or software capture B occurs.

The TCD counter value is synchronized to CAPTUREB by either software or an event.

The capture register is blocked for an update of new capture data until the higher byte of this register is read.

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CAPTUREB[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 11:0 – CAPTUREB[11:0]** Capture B Value

### 25.5.21 Compare Set A

**Name:** CMPASET
**Offset:** 0x28
**Reset:** 0x00
**Property:** -

The TCDn.CMPASETL and TCDn.CMPASETH register pair represents the 12-bit TCDn.CMPASET value. This register is continuously compared to the counter value. Then, the outputs from the comparators are used for generating waveforms.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMPASET[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 |
|---|---|---|---|---|---|---|---|---|
| | CMPASET[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 – CMPASET[11:0]** Compare A Set
These bits hold the value of the compare register.

### 25.5.22 Compare Set B

**Name:** CMPBSET
**Offset:** 0x2C
**Reset:** 0x00
**Property:** -

The TCDn.CMPBSETL and TCDn.CMPBSETH register pair represents the 12-bit TCDn.CMPBSET value. This register is continuously compared to the counter value. Then, the outputs from the comparators are used for generating waveforms.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMPBSET[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 |
|---|---|---|---|---|---|---|---|---|
| | CMPBSET[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 – CMPBSET[11:0]** Compare B Set
These bits hold the value of the compare register.

### 25.5.23 Compare Clear A

**Name:** CMPACLR
**Offset:** 0x2A
**Reset:** 0x00
**Property:** -

The TCDn.CMPACLRL and TCDn.CMPACLRH register pair represents the 12-bit TCDn.CMPACLR value. This register is continuously compared to the counter value. Then, the outputs from the comparators are used for generating waveforms.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMPACLR[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 |
|---|---|---|---|---|---|---|---|---|
| | CMPACLR[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 – CMPACLR[11:0]** Compare A Clear
These bits hold the value of the compare register.

### 25.5.24 Compare Clear B

**Name:** CMPBCLR
**Offset:** 0x2E
**Reset:** 0x00
**Property:** -

The TCDn.CMPBCLRL and TCDn.CMPBCLRH register pair represents the 12-bit TCDn.CMPBCLR value. This register is continuously compared to the counter value. Then, the outputs from the comparators are used for generating waveforms.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMPBCLR[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 |
|---|---|---|---|---|---|---|---|---|
| | CMPBCLR[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 – CMPBCLR[11:0]** Compare B Clear
These bits hold the value of the compare register.

# 26. RTC - Real-Time Counter

## 26.1 Features

- 16-bit Resolution
- Selectable Clock Sources
- Programmable 15-bit Clock Prescaling
- One Compare Register
- One Period Register
- Clear Timer on Period Overflow
- Optional Interrupt/Event on Overflow and Compare Match
- Periodic Interrupt and Event
- Crystal Error Correction

## 26.2 Overview

The RTC peripheral offers two timing functions: the Real-Time Counter (RTC) and a Periodic Interrupt Timer (PIT).

The PIT functionality can be enabled independently of the RTC functionality.

**RTC - Real-Time Counter**
The RTC counts (prescaled) clock cycles in a Counter register and compares the content of the Counter register to a Period register and a Compare register.

The RTC can generate both interrupts and events on compare match or overflow. It will generate a compare interrupt and/or event at the first count after the counter equals the Compare register value, and an overflow interrupt and/or event at the first count after the counter value equals the Period register value. The overflow will reset the counter value to zero.

The RTC peripheral typically runs continuously, including in Low-Power Sleep modes, to keep track of time. It can wake up the device from sleep modes and/or interrupt the device at regular intervals.

The reference clock is typically the 32.768 kHz output from an external crystal. The RTC can also be clocked from an external clock signal, the 32.768 kHz Internal Oscillator (OSC32K), or the OSC32K divided by 32.

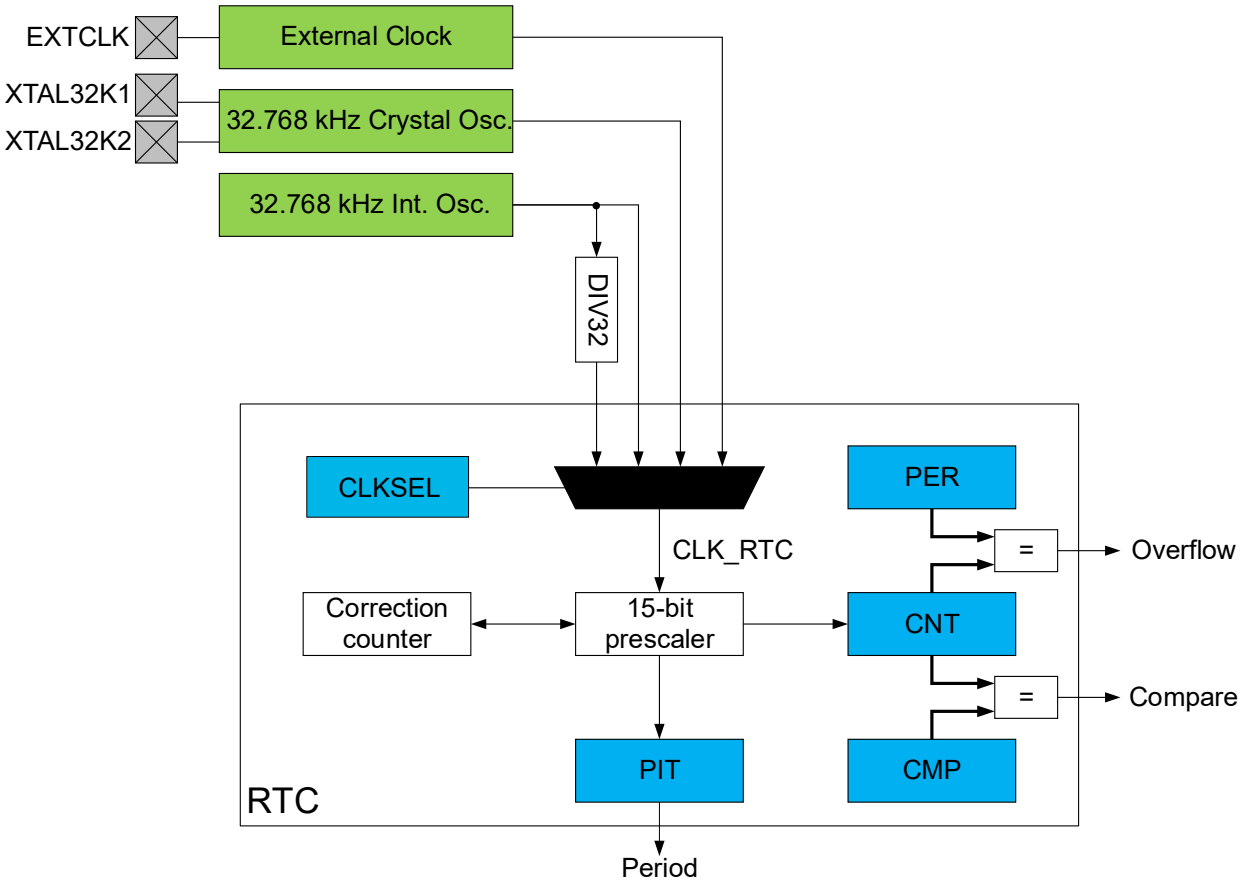
The RTC peripheral includes a 15-bit programmable prescaler that can scale down the reference clock before it reaches the counter. A wide range of resolutions and time-out periods can be configured for the RTC. With a 32.768 kHz clock source, the maximum resolution is 30.5 µs, and time-out periods can be up to two seconds. With a resolution of 1s, the maximum time-out period is more than 18 hours (65536 seconds).

The RTC also supports crystal error correction when operated using external crystal selection. An externally calibrated value will be used for correction. The RTC can be adjusted by software with an accuracy of ±1 PPM, and the maximum adjustment is ±127 PPM. The RTC correction operation will either speed up (by skipping count) or slow down (by adding extra count) the prescaler to account for the crystal error.

**PIT - Periodic Interrupt Timer**
The PIT uses the same clock source (CLK_RTC) as the RTC function and can generate an interrupt request or a level event on every $n^{th}$ clock period. The n can be selected from {4, 8, 16,... 32768} for interrupts and from {64, 128, 256,... 8192} for events.

### 26.2.1 Block Diagram

**RTC Block Diagram**




## 26.3 Clocks

The peripheral clock (CLK_PER) is required to be at least four times faster than the RTC clock (CLK_RTC) for reading the counter value, regardless of the prescaler setting.

A 32.768 kHz crystal can be connected to the XTAL32K1 or XTAL32K2 pins, along with any required load capacitors. Alternatively, an external digital clock can be connected to the XTAL32K1 pin.

## 26.4 RTC Functional Description

The RTC peripheral offers two timing functions: the Real-Time Counter (RTC) and a Periodic Interrupt Timer (PIT). This subsection describes the RTC.

### 26.4.1 Initialization

Before enabling the RTC peripheral and the desired actions (interrupt requests and output events), the source clock for the RTC counter must be configured to operate the RTC.

#### 26.4.1.1 Configure the Clock CLK_RTC

To configure the CLK_RTC, follow these steps:

1. Configure the desired oscillator to operate as required, in the Clock Controller (CLKCTRL) peripheral.
2. Write the Clock Select (CLKSEL) bit field in the Clock Selection (RTC.CLKSEL) register accordingly.

The CLK_RTC clock configuration is used by both RTC and PIT functionality.

#### 26.4.1.2 Configure RTC

To operate the RTC, follow these steps:

1. Set the compare value in the Compare (RTC.CMP) register, and/or the overflow value in the Period (RTC.PER) register.
2. Enable the desired interrupts by writing to the respective interrupt enable bits (CMP, OVF) in the Interrupt Control (RTC.INTCTRL) register.
3. Configure the RTC internal prescaler by writing the desired value to the Prescaler (PRESCALER) bit field in the Control A (RTC.CTRLA) register.
4. Enable the RTC by writing a '1' to the RTC Peripheral Enable (RTCEN) bit in the RTC.CTRLA register.

**Note:** The RTC peripheral is used internally during device start-up. Always check the Synchronization Busy bits in the Status (RTC.STATUS) and Periodic Interrupt Timer Status (RTC.PITSTATUS) registers, and on the initial configuration.

### 26.4.2 Operation - RTC

#### 26.4.2.1 Enabling and Disabling

The RTC is enabled by writing the RTC Peripheral Enable (RTCEN) bit in the Control A (RTC.CTRLA) register to '1'. The RTC is disabled by writing the RTC Peripheral Enable (RTCEN) bit in RTC.CTRLA to '0'.

## 26.5 PIT Functional Description

The RTC peripheral offers two timing functions: the Real-Time Counter (RTC) and a Periodic Interrupt Timer (PIT). This subsection describes the PIT.

### 26.5.1 Initialization

To operate the PIT, follow these steps:

1. Configure the RTC clock CLK_RTC as described in section 26.4.1.1 Configure the Clock CLK_RTC.
2. Enable the interrupt by writing a '1' to the Periodic Interrupt (PI) bit in the PIT Interrupt Control (RTC.PITINTCTRL) register.
3. Select the period for the interrupt by writing the desired value to the Period (PERIOD) bit field in the Periodic Interrupt Timer Control A (RTC.PITCTRLA) register.
4. Enable the PIT by writing a '1' to the Periodic Interrupt Timer Enable (PITEN) bit in the RTC.PITCTRLA register.

**Note:** The RTC peripheral is used internally during device start-up. Always check the Synchronization Busy bits in the RTC.STATUS and RTC.PITSTATUS registers, and on the initial configuration.

### 26.5.2 Operation - PIT

#### 26.5.2.1 Enabling and Disabling

The PIT is enabled by writing the Periodic Interrupt Timer Enable (PITEN) bit in the Periodic Interrupt Timer Control A (RTC.PITCTRLA) register to '1'. The PIT is disabled by writing the Periodic Interrupt Timer Enable (PITEN) bit in RTC.PITCTRLA to '0'.

#### 26.5.2.2 PIT Interrupt Timing

**Timing of the First Interrupt**

The PIT function and the RTC function are running from the same counter inside the prescaler and can be configured as described below:

- The RTC interrupt period is configured by writing the Period (RTC.PER) register
- The PIT interrupt period is configured by writing the Period (PERIOD) bit field in Periodic Interrupt Timer Control A (RTC.PITCTRLA) register

The prescaler is OFF when both functions are OFF (RTC Peripheral Enable (RTCEN) bit in RTC.CTRLA and the Periodic Interrupt Timer Enable (PITEN) bit in RTC.PITCTRLA are '0'), but it is running (that is, its internal counter is

counting) when either function is enabled. For this reason, the timing of the first PIT interrupt and the first RTC count tick will be unknown (anytime between enabling and a full period).

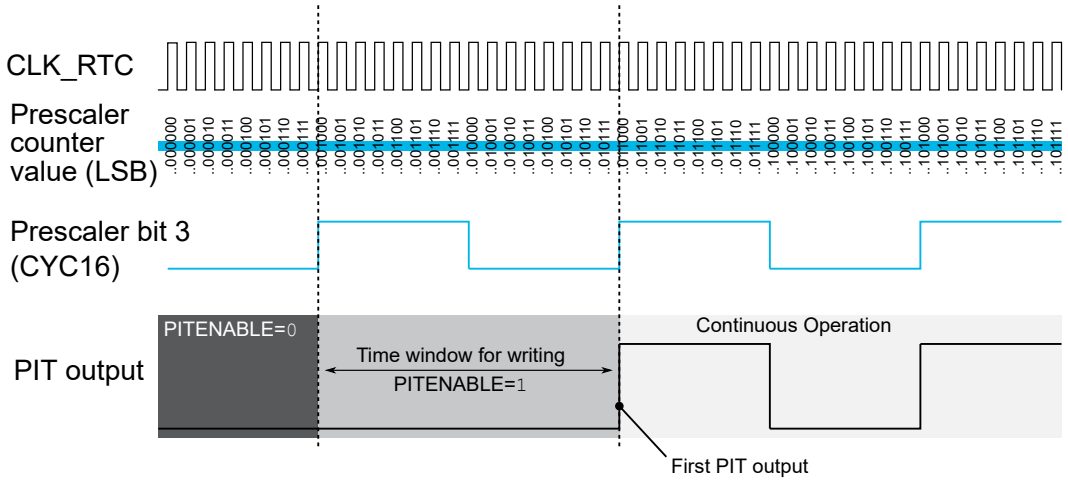
**Continuous Operation**
After the first interrupt, the PIT will continue toggling every ½ PIT period resulting in a full PIT period signal.

**Example 26-1. PIT Timing Diagram for PERIOD=CYC16**

For PERIOD=CYC16 in RTC.PITCTRLA, the PIT output effectively follows the state of the prescaler counter bit 3, so the resulting interrupt output has a period of 16 CLK_RTC cycles.

The time between writing PITEN to '1' and the first PIT interrupt can vary between virtually zero and a full PIT period of 16 CLK_RTC cycles. The precise delay between enabling the PIT and its first output depends on the prescaler's counting phase: the first interrupt shown below is produced by writing PITEN to '1' at any time inside the leading time window.

**Figure 26-1. Timing Between PIT Enable and First Interrupt**




## 26.6 Crystal Error Correction

The prescaler for the RTC and PIT can do internal frequency correction of the crystal clock by using the PPM error value from the Crystal Frequency Calibration (CALIB) register when the Frequency Correction Enable (CORREN) bit in the RTC.CTRLA register is '1'.

The CALIB register must be written by the user, based on the information about the frequency error. The correction operation is performed by adding or removing a number of cycles equal to the value given in the Error Correction Value (ERROR) bit field in the CALIB register spread throughout a million-cycle interval.

The correction of the clock will be reflected in the RTC count value available through the Count (RTC.CNT) registers or in the PIT intervals.

If disabling the correction feature, an ongoing correction cycle will be completed before the function is disabled.

**Note:** If using this feature with a negative correction, the minimum prescaler configuration is DIV2.

## 26.7 Events

The RTC can generate the events described in the following table:

**Table 26-1. Event Generators in RTC**

| Generator Name | | Description | Event Type | Clock Domain | Length of the Event |
|---|---|---|---|---|---|
| Module | Event | | | | |
| RTC | OVF | Overflow | Pulse | CLK_RTC | One CLK_RTC period |
| | CMP | Compare Match | | | One CLK_RTC period |
| | PIT_DIV8192 | Prescaled RTC clock divided by 8192 | Level | | Given by prescaled RTC clock divided by 8192 |
| | PIT_DIV4096 | Prescaled RTC clock divided by 4096 | | | Given by prescaled RTC clock divided by 4096 |
| | PIT_DIV2048 | Prescaled RTC clock divided by 2048 | | | Given by prescaled RTC clock divided by 2048 |
| | PIT_DIV1024 | Prescaled RTC clock divided by 1024 | | | Given by prescaled RTC clock divided by 1024 |
| | PIT_DIV512 | Prescaled RTC clock divided by 512 | | | Given by prescaled RTC clock divided by 512 |
| | PIT_DIV256 | Prescaled RTC clock divided by 256 | | | Given by prescaled RTC clock divided by 256 |
| | PIT_DIV128 | Prescaled RTC clock divided by 128 | | | Given by prescaled RTC clock divided by 128 |
| | PIT_DIV64 | Prescaled RTC clock divided by 64 | | | Given by prescaled RTC clock divided by 64 |

The conditions for generating the OVF and CMP events are identical to those that will raise the corresponding interrupt flags in the RTC.INTFLAGS register.

Refer to the *(EVSYS) Event System* section for more details regarding event users and Event System configuration.

## 26.8 Interrupts

**Table 26-2. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| RTC | Real-Time Counter overflow and compare match interrupt | • Overflow (OVF): The counter has reached the value from the RTC.PER register and wrapped to zero.<br>• Compare (CMP): Match between the value from the Counter (RTC.CNT) register and the value from the Compare (RTC.CMP) register. |
| PIT | Periodic Interrupt Timer interrupt | A time period has passed, as configured by the PERIOD bit field in RTC.PITCTRLA. |

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

Note that:

- The RTC has two INTFLAGS registers: RTC.INTFLAGS and RTC.PITINTFLAGS.
- The RTC has two INTCTRL registers: RTC.INTCTRL and RTC.PITINTCTRL.

## 26.9 Sleep Mode Operation

The RTC will continue to operate in Idle Sleep mode. It will run in Standby Sleep mode if the Run in Standby (RUNSTDBY) bit in RTC.CTRLA is set.

The PIT will continue to operate in any sleep mode.

## 26.10 Synchronization

Both the RTC and the PIT are asynchronous, operating from a different clock source (CLK_RTC) independently of the peripheral clock (CLK_PER). For Control and Count register updates, it will take some RTC and/or peripheral clock cycles before an updated register value is available in a register or until a configuration change affects the RTC or PIT, respectively. This synchronization time is described for each register in the *Register Description* section.

For some RTC registers, a Synchronization Busy flag is available (CMPBUSY, PERBUSY, CNTBUSY, CTRLABUSY) in the Status (RTC.STATUS) register.

For the RTC.PITCTRLA register, a Synchronization Busy flag is available (CTRLBUSY) in the Periodic Interrupt Timer Status (RTC.PITSTATUS) register.

Check these flags before writing to the mentioned registers.

## 26.11 Debug Operation

If the Debug Run (DBGRUN) bit in the Debug Control (RTC.DBGCTRL) register is '1', the RTC will continue normal operation. If DBGRUN is '0' and the CPU is halted, the RTC will halt the operation and ignore any incoming events.

If the Debug Run (DBGRUN) bit in the Periodic Interrupt Timer Debug Control (RTC.PITDBGCTRL) register is '1', the PIT will continue normal operation. If DBGRUN is '0' in the Debug mode and the CPU is halted, the PIT output will be low. When the PIT output is high at the time, a new positive edge occurs to set the interrupt flag when restarting from a break. The result is an additional PIT interrupt that would not happen during normal operation. If the PIT output is low at the break, the PIT will resume low without additional interrupt.

## 26.12 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | RUNSTDBY | PRESCALER[3:0] | | | | CORREN | | RTCEN |
| 0x01 | STATUS | 7:0 | | | | | CMPBUSY | PERBUSY | CNTBUSY | CTRLABUSY |
| 0x02 | INTCTRL | 7:0 | | | | | | | CMP | OVF |
| 0x03 | INTFLAGS | 7:0 | | | | | | | CMP | OVF |
| 0x04 | TEMP | 7:0 | TEMP[7:0] | | | | | | | |
| 0x05 | DBGCTRL | 7:0 | | | | | | | | DBGRUN |
| 0x06 | CALIB | 7:0 | SIGN | ERROR[6:0] | | | | | | |
| 0x07 | CLKSEL | 7:0 | | | | | | | CLKSEL[1:0] | |
| 0x08 | CNT | 7:0 | CNT[7:0] | | | | | | | |
| | | 15:8 | CNT[15:8] | | | | | | | |
| 0x0A | PER | 7:0 | PER[7:0] | | | | | | | |
| | | 15:8 | PER[15:8] | | | | | | | |
| 0x0C | CMP | 7:0 | CMP[7:0] | | | | | | | |
| | | 15:8 | CMP[15:8] | | | | | | | |
| 0x0E ... 0x0F | Reserved | | | | | | | | | |
| 0x10 | PITCTRLA | 7:0 | | PERIOD[3:0] | | | | | | PITEN |
| 0x11 | PITSTATUS | 7:0 | | | | | | | | CTRLBUSY |
| 0x12 | PITINTCTRL | 7:0 | | | | | | | | PI |
| 0x13 | PITINTFLAGS | 7:0 | | | | | | | | PI |
| 0x14 | Reserved | | | | | | | | | |
| 0x15 | PITDBGCTRL | 7:0 | | | | | | | | DBGRUN |

## 26.13 Register Description

### 26.13.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | PRESCALER[3:0] | | | | CORREN | | RTCEN |
| 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 – RUNSTDBY** Run in Standby

| Value | Description |
|---|---|
| 0 | RTC disabled in Standby Sleep mode |
| 1 | RTC enabled in Standby Sleep mode |

**Bits 6:3 – PRESCALER[3:0]** Prescaler
These bits define the prescaling of the CLK_RTC clock signal. Due to synchronization between the RTC clock and the peripheral clock, there is a latency of two RTC clock cycles from updating the register until this has an effect. Application software needs to check that the CTRLABUSY flag in RTC.STATUS register is cleared before writing to this register.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV1 | RTC clock/1 (no prescaling) |
| 0x1 | DIV2 | RTC clock/2 |
| 0x2 | DIV4 | RTC clock/4 |
| 0x3 | DIV8 | RTC clock/8 |
| 0x4 | DIV16 | RTC clock/16 |
| 0x5 | DIV32 | RTC clock/32 |
| 0x6 | DIV64 | RTC clock/64 |
| 0x7 | DIV128 | RTC clock/128 |
| 0x8 | DIV256 | RTC clock/256 |
| 0x9 | DIV512 | RTC clock/512 |
| 0xA | DIV1024 | RTC clock/1024 |
| 0xB | DIV2048 | RTC clock/2048 |
| 0xC | DIV4096 | RTC clock/4096 |
| 0xD | DIV8192 | RTC clock/8192 |
| 0xE | DIV16384 | RTC clock/16384 |
| 0xF | DIV32768 | RTC clock/32768 |

**Bit 2 – CORREN** Frequency Correction Enable

| Value | Description |
|---|---|
| 0 | Frequency correction is disabled |
| 1 | Frequency correction is enabled |

**Bit 0 – RTCEN** RTC Peripheral Enable

| Value | Description |
|---|---|
| 0 | RTC peripheral is disabled |
| 1 | RTC peripheral is enabled |

### 26.13.2 Status

**Name:** STATUS
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CMPBUSY | PERBUSY | CNTBUSY | CTRLABUSY |
| Access | | | | | R | R | R | R |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bit 3 – CMPBUSY** Compare Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Compare (RTC.CMP) register in the RTC clock domain.

**Bit 2 – PERBUSY** Period Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Period (RTC.PER) register in the RTC clock domain.

**Bit 1 – CNTBUSY** Counter Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Count (RTC.CNT) register in the RTC clock domain.

**Bit 0 – CTRLABUSY** Control A Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Control A (RTC.CTRLA) register in the RTC clock domain.

### 26.13.3 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | CMP | OVF |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – CMP** Compare Match Interrupt Enable
Enable interrupt-on-compare match (that is, when the value from the Count (RTC.CNT) register matches the value from the Compare (RTC.CMP) register).

| Value | Description |
|---|---|
| 0 | The compare match interrupt is disabled |
| 1 | The compare match interrupt is enabled |

**Bit 0 – OVF** Overflow Interrupt Enable
Enable interrupt-on-counter overflow (that is, when the value from the Count (RTC.CNT) register matched the value from the Period (RTC.PER) register and wraps around to zero).

| Value | Description |
|---|---|
| 0 | The overflow interrupt is disabled |
| 1 | The overflow interrupt is enabled |

### 26.13.4 Interrupt Flag

**Name:** INTFLAGS
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | CMP | OVF |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – CMP** Compare Match Interrupt Flag
This flag is set when the value from the Count (RTC.CNT) register matches the value from the Compare (RTC.CMP) register.
Writing a '1' to this bit clears the flag.

**Bit 0 – OVF** Overflow Interrupt Flag
This flag is set when the value from the Count (RTC.CNT) register has reached the value from the Period (RTC.PER) register and wrapped to zero.
Writing a '1' to this bit clears the flag.

### 26.13.5 Temporary

**Name:** TEMP
**Offset:** 0x4
**Reset:** 0x00
**Property:** -

The Temporary register is used by the CPU for 16-bit single-cycle access to the 16-bit registers of this peripheral. The register is common for all the 16-bit registers of this peripheral and can be read and written by software. For more details on reading and writing 16-bit registers, refer to *Accessing 16-Bit Registers* in the *Memories* section.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TEMP[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 – TEMP[7:0]** Temporary
Temporary register for read/write operations in 16-bit registers.

### 26.13.6 Debug Control

**Name:** DBGCTRL
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Debug Run

| Value | Description |
|---|---|
| 0 | The peripheral is halted in Break Debug mode and ignores events. |
| 1 | The peripheral will continue to run in Break Debug mode when the CPU is halted. |

### 26.13.7 Crystal Frequency Calibration

**Name:** CALIB
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

This register stores the error value and the type of correction to be done. This register is written by software with any error value based on external calibration and/or temperature correction/s.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SIGN | ERROR[6:0] | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – SIGN** Error Correction Sign Bit
This bit shows the direction of the correction.

| Value | Description |
|---|---|
| 0x0 | Positive correction causing the prescaler to count slower. |
| 0x1 | Negative correction causing the prescaler to count faster. This requires that the minimum prescaler configuration is DIV2. |

**Bits 6:0 – ERROR[6:0]** Error Correction Value
The number of correction clocks for each million RTC clock cycles interval (PPM).

### 26.13.8 Clock Selection

**Name:** CLKSEL
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

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

**Bits 1:0 – CLKSEL[1:0]** Clock Select
Writing these bits select the source for the RTC clock (CLK_RTC).

| Value | Name | Description |
|---|---|---|
| 0x0 | OSC32K | 32.768 kHz from OSC32K |
| 0x1 | OSC1K | 1.024 kHz from OSC32K |
| 0x2 | XOSC32K | 32.768 kHz from XOSC32K |
| 0x3 | EXTCLK | External clock from the EXTCLK/ XTALHF1 pin |

### 26.13.9 Count

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

The RTC.CNTL and RTC.CNTH register pair represents the 16-bit value, RTC.CNT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

Due to the synchronization between the RTC clock and main clock domains, there is a latency of two RTC clock cycles from updating the register until this has an effect. The application software needs to check that the CNTBUSY flag in RTC.STATUS is cleared before writing to this register.

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

**Bits 15:8 – CNT[15:8]** Counter High Byte
These bits hold the MSB of the 16-bit Counter register.

**Bits 7:0 – CNT[7:0]** Counter Low Byte
These bits hold the LSB of the 16-bit Counter register.

### 26.13.10 Period

**Name:** PER
**Offset:** 0x0A
**Reset:** 0xFFFF
**Property:** -

The RTC.PERL and RTC.PERH register pair represents the 16-bit value, RTC.PER. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

Due to the synchronization between the RTC clock and main clock domains, there is a latency of two RTC clock cycles from updating the register until this has an effect. The application software needs to check that the PERBUSY flag in RTC.STATUS is cleared before writing to this register.

| 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

**Bits 15:8 – PER[15:8]** Period High Byte
These bits hold the MSB of the 16-bit Period register.

**Bits 7:0 – PER[7:0]** Period Low Byte
These bits hold the LSB of the 16-bit Period register.

### 26.13.11 Compare

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

The RTC.CMPL and RTC.CMPH register pair represents the 16-bit value, RTC.CMP. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

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

**Bits 15:8 – CMP[15:8]** Compare High Byte
These bits hold the MSB of the 16-bit Compare register.

**Bits 7:0 – CMP[7:0]** Compare Low Byte
These bits hold the LSB of the 16-bit Compare register.

### 26.13.12 Periodic Interrupt Timer Control A

**Name:** PITCTRLA
**Offset:** 0x10
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | PERIOD[3:0] | | | | | | PITEN |
| Access | | R/W | R/W | R/W | R/W | | | R/W |
| Reset | | 0 | 0 | 0 | 0 | | | 0 |

**Bits 6:3 – PERIOD[3:0]** Period
Writing this bit field selects the number of RTC clock cycles between each interrupt.

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | No interrupt |
| 0x1 | CYC4 | 4 cycles |
| 0x2 | CYC8 | 8 cycles |
| 0x3 | CYC16 | 16 cycles |
| 0x4 | CYC32 | 32 cycles |
| 0x5 | CYC64 | 64 cycles |
| 0x6 | CYC128 | 128 cycles |
| 0x7 | CYC256 | 256 cycles |
| 0x8 | CYC512 | 512 cycles |
| 0x9 | CYC1024 | 1024 cycles |
| 0xA | CYC2048 | 2048 cycles |
| 0xB | CYC4096 | 4096 cycles |
| 0xC | CYC8192 | 8192 cycles |
| 0xD | CYC16384 | 16384 cycles |
| 0xE | CYC32768 | 32768 cycles |
| 0xF | - | Reserved |

**Bit 0 – PITEN** Periodic Interrupt Timer Enable

| Value | Description |
|---|---|
| 0 | Periodic Interrupt Timer disabled |
| 1 | Periodic Interrupt Timer enabled |

### 26.13.13 Periodic Interrupt Timer Status

**Name:** PITSTATUS
**Offset:** 0x11
**Reset:** 0x00
**Property:** -

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

**Bit 0 – CTRLBUSY** PITCTRLA Synchronization Busy
This bit is '1' when the RTC is busy synchronizing the Periodic Interrupt Timer Control A (RTC.PITCTRLA) register in the RTC clock domain.

### 26.13.14 PIT Interrupt Control

**Name:** PITINTCTRL
**Offset:** 0x12
**Reset:** 0x00
**Property:** -

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

**Bit 0 – PI** Periodic Interrupt

| Value | Description |
|---|---|
| 0 | The periodic interrupt is disabled |
| 1 | The periodic interrupt is enabled |

### 26.13.15 PIT Interrupt Flag

**Name:** PITINTFLAGS
**Offset:** 0x13
**Reset:** 0x00
**Property:** -

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

**Bit 0 – PI** Periodic Interrupt Flag
This flag is set when a periodic interrupt is issued.
Writing a '`1`' clears the flag.

### 26.13.16 Periodic Interrupt Timer Debug Control

**Name:** PITDBGCTRL
**Offset:** 0x15
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Debug Run

| Value | Description |
|---|---|
| 0 | The peripheral is halted in Break Debug mode and ignores events. |
| 1 | The peripheral will continue to run in Break Debug mode when the CPU is halted. |

# 27. USART - Universal Synchronous and Asynchronous Receiver and Transmitter

## 27.1 Features

- Full-Duplex Operation
- Half-Duplex Operation:
  - One-Wire mode
  - RS-485 mode
- Asynchronous or Synchronous Operation
- Supports Serial Frames with Five, Six, Seven, Eight or Nine Data Bits and One or Two Stop Bits
- Fractional Baud Rate Generator:
  - Can generate the desired baud rate from any peripheral clock frequency
  - No need for an external oscillator
- Built-In Error Detection and Correction Schemes:
  - Odd or even parity generation and parity check
  - Buffer overflow and frame error detection
  - Noise filtering including false Start bit detection and digital low-pass filter
- Separate Interrupts for:
  - Transmit complete
  - Transmit Data register empty
  - Receive complete
- Master SPI Mode
- Multiprocessor Communication Mode
- Start-of-Frame Detection
- IRCOM Module for IrDA® Compliant Pulse Modulation/Demodulation
- LIN Slave Support

## 27.2 Overview

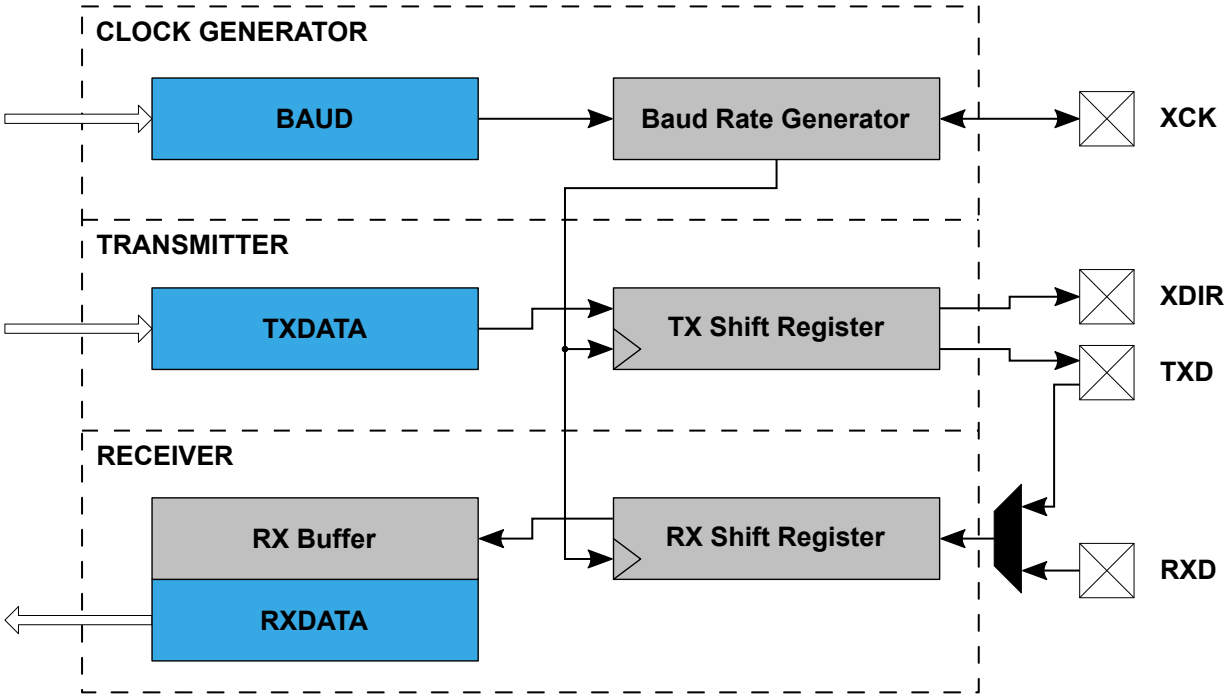
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a fast and flexible serial communication peripheral. The USART supports a number of different modes of operation that can accommodate multiple types of applications and communication devices. For example, the One-Wire Half-Duplex mode is useful when low pin count applications are desired. The communication is frame-based, and the frame format can be customized to support a wide range of standards.

The USART is buffered in both directions, enabling continued data transmission without any delay between frames. Separate interrupts for receive and transmit completion allow fully interrupt-driven communication.

The transmitter consists of a single-write buffer, a Shift register, and control logic for different frame formats. The receiver consists of a two-level receive buffer and a Shift register. The 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.

### 27.2.1 Block Diagram

**Figure 27-1. USART Block Diagram**




### 27.2.2 Signal Description

| Signal | Type | Description |
|---|---|---|
| XCK | Output/input | Clock for synchronous operation |
| XDIR | Output | Transmit enable for RS-485 |
| TxD | Output/input | Transmitting line (and receiving line in One-Wire mode) |
| RxD | Input | Receiving line |

## 27.3 Functional Description

### 27.3.1 Initialization

**Full Duplex Mode:**

1. Set the baud rate (USARTn.BAUD).
2. Set the frame format and mode of operation (USARTn.CTRLC).
3. Configure the TXD pin as an output.
4. Enable the transmitter and the receiver (USARTn.CTRLB).

**Notes:**

- For interrupt-driven USART operation, global interrupts must be disabled during the initialization
- Before doing a reinitialization with a changed baud rate or frame format, be sure that there are no ongoing transmissions while the registers are changed

**One-Wire Half Duplex Mode:**

1. Internally connect the TXD to the USART receiver (the LBME bit in the USARTn.CTRLA register).
2. Enable internal pull-up for the RX/TX pin (the PULLUPEN bit in the PORTx.PINnCTRL register).

3. Enable Open-Drain mode (the ODME bit in the USARTn.CTRLB register).
4. Set the baud rate (USARTn.BAUD).
5. Set the frame format and mode of operation (USARTn.CTRLC).
6. Enable the transmitter and the receiver (USARTn.CTRLB).

**Notes:**

- When Open-Drain mode is enabled, the TXD pin is automatically set to output by hardware
- For interrupt-driven USART operation, global interrupts must be disabled during the initialization
- Before doing a reinitialization with a changed baud rate or frame format, be sure that there are no ongoing transmissions while the registers are changed

## 27.3.2 Operation

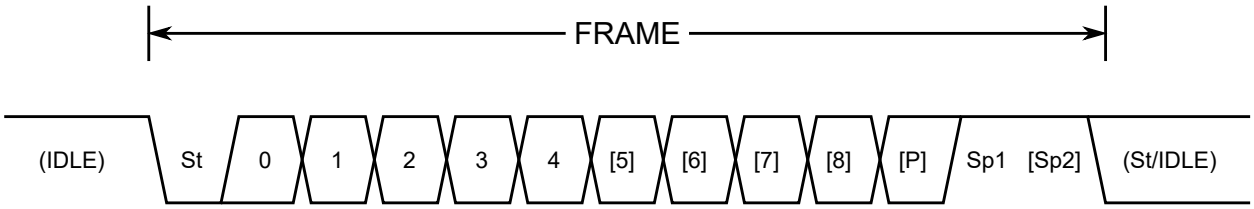
### 27.3.2.1 Frame Formats

The USART data transfer is frame-based. A frame starts with a 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 USART accepts all combinations of the following as valid frame formats:

- 1 Start bit
- 5, 6, 7, 8, or 9 data bits
- No, even, or odd Parity bit
- 1 or 2 Stop bits

The figure below illustrates the possible combinations of frame formats. Bits inside brackets are optional.

**Figure 27-2. Frame Formats**




**St** Start bit, always low

**(n)** Data bits (0 to 8)

**P** Parity bit, may be odd or even

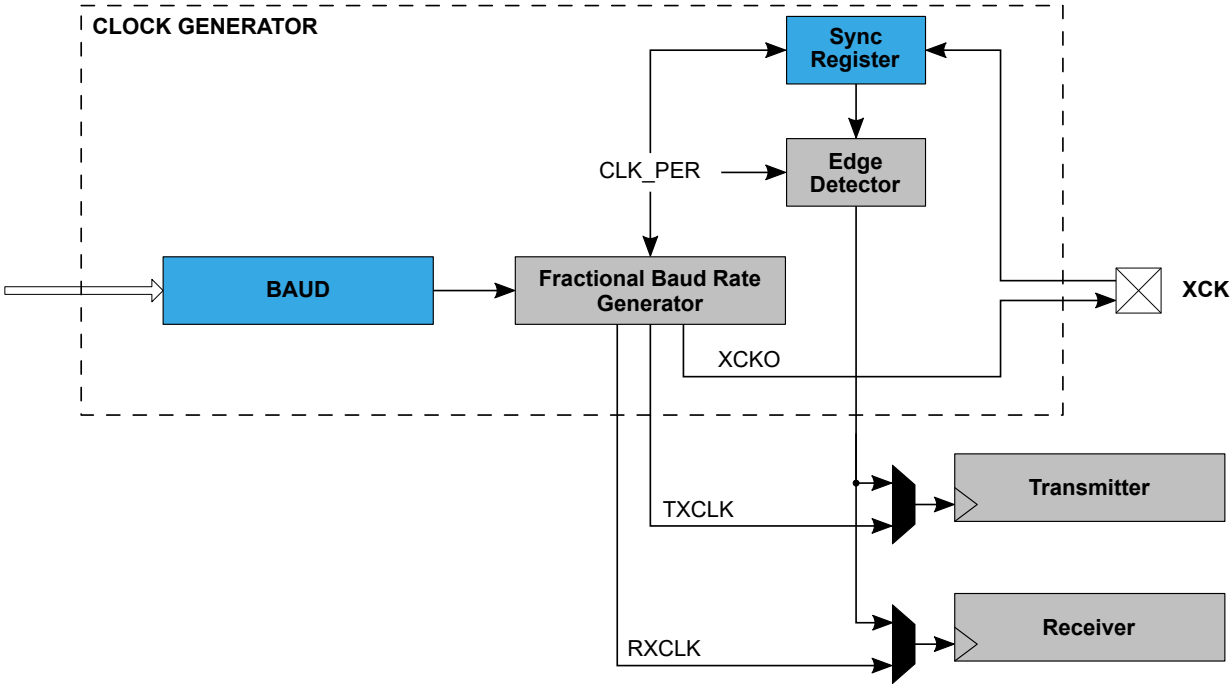
**Sp** Stop bit, always high

**IDLE** No transfer on the communication line (RxD or TxD). The Idle state is always high.

### 27.3.2.2 Clock Generation

The clock used for shifting and sampling data bits is generated internally by the fractional baud rate generator or externally from the Transfer Clock (XCK) pin.

**Figure 27-3. Clock Generation Logic Block Diagram**




#### 27.3.2.2.1 The Fractional Baud Rate Generator

In modes where the USART is not using the XCK input as a clock source, the fractional Baud Rate Generator is used to generate the clock. Baud rate is given in terms of bits per second (bps) and is configured by writing the USARTn.BAUD register. The baud rate ($f_{BAUD}$) is generated by dividing the peripheral clock ($f_{CLK_PER}$) by a division factor decided by the BAUD register.

The fractional Baud Rate Generator features hardware that accommodates cases where $f_{CLK_PER}$ is not divisible by $f_{BAUD}$. Usually, this situation would lead to a rounding error. The fractional Baud Rate Generator expects the BAUD register to contain the desired division factor left shifted by six bits, as implemented by the equations in Table 27-1. The six LSbs will then hold the fractional part of the desired divisor. The fractional part of the BAUD register is used to dynamically adjust $f_{BAUD}$ to achieve a closer approximation to the desired baud rate.

Since the baud rate cannot be higher than $f_{CLK_PER}$, the integer part of the BAUD register needs to be at least 1. Since the result is left shifted by six bits, the corresponding minimum value of the BAUD register is 64. The valid range is, therefore, 64 to 65535.

In Synchronous mode, only the 10-bit integer part of the BAUD register (BAUD[15:6]) determines the baud rate, and the fractional part (BAUD[5:0]) must, therefore, be written to zero.

The table below lists equations for translating baud rates into input values for the BAUD register. The equations take fractional interpretation into consideration, so the BAUD values calculated with these equations can be written directly to USARTn.BAUD without any additional scaling.

**Table 27-1. Equations for Calculating Baud Rate Register Setting**

| Operating Mode | Conditions | Baud Rate (Bits Per Seconds) | USART.BAUD Register Value Calculation |
|---|---|---|---|
| Asynchronous | $f_{BAUD} \leq \frac{f_{CLK_PER}}{S}$<br>$USART.BAUD \geq 64$ | $f_{BAUD} = \frac{64 \times f_{CLK_PER}}{S \times BAUD}$ | $BAUD = \frac{64 \times f_{CLK_PER}}{S \times f_{BAUD}}$ |
| Synchronous Master | $f_{BAUD} \leq \frac{f_{CLK_PER}}{S}$<br>$USART.BAUD \geq 64$ | $f_{BAUD} = \frac{f_{CLK_PER}}{S \times BAUD[15:6]}$ | $BAUD[15:6] = \frac{f_{CLK_PER}}{S \times f_{BAUD}}$ |

S is the number of samples per bit

- Asynchronous Normal mode: S = 16
- Asynchronous Double-Speed mode: S = 8
- Synchronous mode: S = 2

#### 27.3.2.3 Data Transmission

The USART transmitter sends data by periodically driving the transmission line low. The data transmission is initiated by loading the transmit buffer (USARTn.TXDATA) with the data to be sent. The data in the transmit buffer is moved to the Shift register once it 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 frame in the Shift register has been shifted out, and there are no new data present in the transmit buffer, the Transmit Complete Interrupt Flag (the TXCIF bit in the USARTn.STATUS register) is set, and the interrupt is generated if it is enabled.

TXDATA can only be written when the Data Register Empty Interrupt Flag (the DREIF bit in the USARTn.STATUS register) is set, indicating that the register is empty and ready for new data.

When using frames with fewer than eight bits, the Most Significant bits (MSb) written to TXDATA are ignored. If 9-bit characters are used, the DATA[8] bit in the USARTn.TXDATAH register has to be written before the DATA[7:0] bits in the USARTn.TXDATAL register.

##### 27.3.2.3.1 Disabling the Transmitter

When disabling the transmitter, the operation will not become effective until ongoing and pending transmissions are completed (that is, when the Transmit Shift register and Transmit Buffer register do not contain data to be transmitted). When the transmitter is disabled, it will no longer override the TXD pin, and the PORT module regains control of the pin. The pin is automatically configured as an input by hardware regardless of its previous setting. The pin can now be used as a normal I/O pin with no port override from the USART.

#### 27.3.2.4 Data Reception

The USART receiver samples the reception line to detect and interpret the received data. The direction of the pin must, therefore, be configured as an input by writing a '0' to the corresponding bit in the Direction register (PORTx.DIRn).

The receiver accepts data when a valid Start bit is detected. Each bit that follows the Start bit will be sampled at the baud rate or XCK clock and shifted into the Receive Shift register until the first Stop bit of a frame is received. A 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 receive buffer. The Receive Complete Interrupt Flag (the RXCIF bit in the USARTn.STATUS register) is set, and the interrupt is generated if enabled.

The RXDATA registers are the part of the double-buffered RX buffer that can be read by the application software when RXCIF is set. If only one frame has been received, the data and status bits for that frame are pushed to the RXDATA registers directly. If two frames are present in the RX buffer, the RXDATA registers contain the data for the oldest frame.

The buffer shifts out the data either when RXDATAL or RXDATAL is read, depending on the configuration. The register which does not lead to data being shifted should be read first to be able to read both bytes before shifting. When the Character Size (CHSIZE) bits in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), a read of RXDATAH shifts the receive buffer. Otherwise, RXDATAL shifts the buffer.

##### 27.3.2.4.1 Receiver Error Flags

The USART receiver features error detection mechanisms that uncover corruption of the transmission. These mechanisms include the following:

- Frame Error detection - controls whether the received frame is valid
- Buffer Overflow detection - indicates data loss due to the receiver buffer being full and overwritten by the new data
- Parity Error detection - checks the validity of the incoming frame by calculating its parity and comparing it to the Parity bit

Each error detection mechanism controls one error flag that can be read in the RXDATAH register:

- Frame Error (FERR)

- Buffer Overflow (BUFOVF)
- Parity Error (PERR)

The error flags are located in the RX buffer together with their corresponding frame. The RXDATAH register that contains the error flags must be read before the RXDATAL register, since reading the RXDATAL register will trigger the RX buffer to shift out the RXDATA bytes.

**Note:** If the Character Size bit field (the CHSIZE bits in the USARTn.CTRLC register) is set to nine bits, low byte first (9BITL), the RXDATAH register will, instead of the RXDATAL register, trigger the RX buffer to shift out the RXDATA bytes. The RXDATAL register must, in that case, be read before the RXDATAH register.

#### 27.3.2.4.2 Disabling the Receiver

When disabling the receiver, the operation is immediate. The receiver buffer will be flushed, and data from ongoing receptions will be lost.

#### 27.3.2.4.3 Flushing the Receive Buffer

If the RX buffer has to be flushed during normal operation, repeatedly read the DATA location (USARTn.RXDATAH and USARTn.RXDATAL registers) until the Receive Complete Interrupt Flag (the RXCIF bit in the USARTn.RXDATAH register) is cleared.

### 27.3.3 Communication Modes

The USART is a flexible peripheral that supports multiple different communication protocols. The available modes of operation can be split into two groups: Synchronous and asynchronous communication.

The synchronous communication relies on one device on the bus to be the master, providing the rest of the devices with a clock signal through the XCK pin. All the devices use this common clock signal for both transmission and reception, requiring no additional synchronization mechanism.

The device can be configured to run either as a master or a slave on the synchronous bus.

The asynchronous communication does not use a common clock signal. Instead, it relies on the communicating devices to be configured with the same baud rate. When receiving a transmission, the hardware synchronization mechanisms are used to align the incoming transmission with the receiving device peripheral clock.

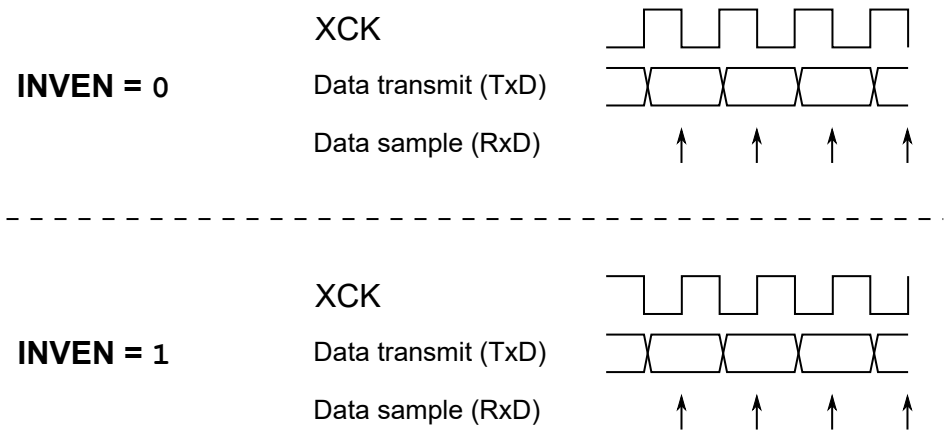
Four different modes of reception are available when communicating asynchronously. One of these modes can receive transmissions at twice the normal speed, sampling only eight times per bit instead of the normal 16. The other three operating modes use variations of synchronization logic, all receiving at normal speed.

#### 27.3.3.1 Synchronous Operation

#### 27.3.3.1.1 Clock Operation

The XCK pin direction controls whether the transmission clock is an input (Slave mode) or an output (Master mode). The corresponding port pin direction must be set to output for Master mode or to input for Slave mode (PORTx.DIRn). The data input (on RXD) is sampled at the XCK clock edge which is opposite the edge where data are transmitted (on TXD) as shown in the figure below.

**Figure 27-4. Synchronous Mode XCK Timing**



The I/O pin can be inverted by writing a '1' to the Inverted I/O Enable (INVEN) bit in the Pin n Control register of the port peripheral (PORTx.PINnCTRL). Using the inverted I/O setting for the corresponding XCK port pin, the XCK clock edges used for sampling RxD and transmitting on TxD can be selected. If the inverted I/O is disabled (INVEN = 0), the rising XCK clock edge represents the start of a new data bit, and the received data will be sampled at the falling XCK clock edge. If inverted I/O is enabled (INVEN = 1), the falling XCK clock edge represents the start of a new data bit, and the received data will be sampled at the rising XCK clock edge.

#### 27.3.3.1.2 External Clock Limitations

When the USART is configured in Synchronous Slave mode, the XCK signal must be provided externally by the master device. Since the clock is provided externally, configuring the BAUD register will have no impact on the transfer speed. Successful clock recovery requires the clock signal to be sampled at least twice for each rising and falling edge. The maximum XCK speed in Synchronous Operation mode, $f_{Slave_XCK}$, is therefore limited by:

$$f_{\text{Slave_XCK}} < \frac{f_{CLK_PER}}{4}$$

If the XCK clock has jitter, or if the high/low period duty cycle is not 50/50, the maximum XCK clock speed must be reduced accordingly to ensure that XCK is sampled a minimum of two times for each edge.

#### 27.3.3.1.3 USART in Master SPI Mode

The USART may be configured to function with multiple different communication interfaces, and one of these is the Serial Peripheral Interface (SPI) where it can function as the master device. The SPI is a four-wire interface that enables a master device to communicate with one or multiple slaves.

***Frame Formats***

The serial frame for the USART in Master SPI mode always contains eight Data bits. The Data bits can be configured to be transmitted with either the LSb or MSb first, by writing to the Data Order bit (UDORD) in the Control C register (USARTn.CTRLC).

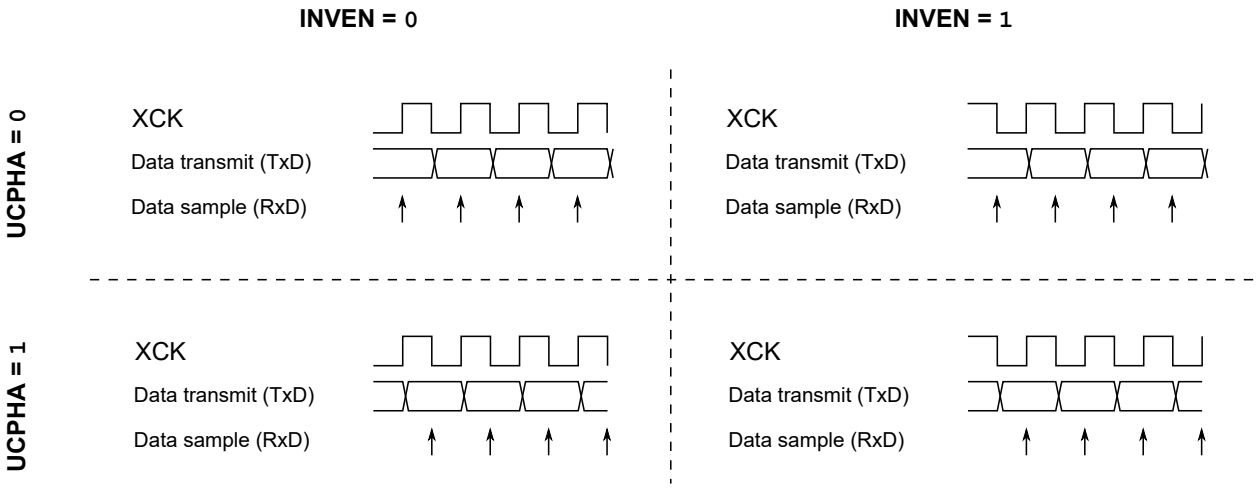
SPI does not use Start, Stop, or Parity bits, so the transmission frame can only consist of the Data bits.

***Clock Generation***

Being a master device in a synchronous communication interface, the USART in Master SPI mode must generate the interface clock to be shared with the slave devices. The interface clock is generated using the fractional Baud Rate Generator, which is described in 27.3.2.2.1 The Fractional Baud Rate Generator.

Each Data bit is transmitted by pulling the data line high or low for one full clock period. The receiver will sample bits in the middle of the transmitter hold period as shown in the figure below. It also shows how the timing scheme can be configured using the Inverted I/O Enable (INVEN) bit in the PORTx.PINnCTRL register and the USART Clock Phase (UCPHA) bit in the USARTn.CTRLC register.

**Figure 27-5. Data Transfer Timing Diagrams**




The table below further explains the figure above.

**Table 27-2. Functionality of INVEN and UCPHA Bits**

| INVEN | UCPHA | Leading Edge (1) | Trailing Edge (1) |
|---|---|---|---|
| 0 | 0 | Rising, sample | Falling, transmit |
| 0 | 1 | Rising, transmit | Falling, sample |
| 1 | 0 | Falling, sample | Rising, transmit |
| 1 | 1 | Falling, transmit | Rising, sample |

**Note:**

1. The leading edge is the first clock edge of a clock cycle. The trailing edge is the last clock edge of a clock cycle.

***Data Transmission***

Data transmission in Master SPI mode is functionally identical to general USART operation as described in the *Operation* section. The transmitter interrupt flags and corresponding USART interrupts are also identical. See 27.3.2.3 Data Transmission for further description.

***Data Reception***

Data reception in Master SPI mode is identical in function to general USART operation as described in the *Operation* section. The receiver interrupt flags and the corresponding USART interrupts are also identical, aside from the receiver error flags that are not in use and always read as '0'. See 27.3.2.4 Data Reception for further description.

***USART in Master SPI Mode vs. SPI***

The USART in Master SPI mode is fully compatible with a stand-alone SPI peripheral. Their data frame and timing configurations are identical. Some SPI specific special features are, however, not supported with the USART in Master SPI mode:

- Write Collision Flag Protection
- Double-Speed mode
- Multi-Master support

A comparison of the pins used with USART in Master SPI mode and with SPI is shown in the table below.

**Table 27-3. Comparison of USART in Master SPI Mode and SPI Pins**

| USART | SPI | Comment |
|---|---|---|
| TXD | MOSI | Master out |
| RXD | MISO | Master in |
| XCK | SCK | Functionally identical |
| - | $\overline{SS}$ | Not supported by USART in Master SPI mode(1) |

**Note:**

1. For the stand-alone SPI peripheral, this pin is used with the Multi-Master function or as a dedicated Slave Select pin. The Multi-Master function is not available with the USART in Master SPI mode, and no dedicated Slave Select pin is available.

### 27.3.3.2 Asynchronous Operation

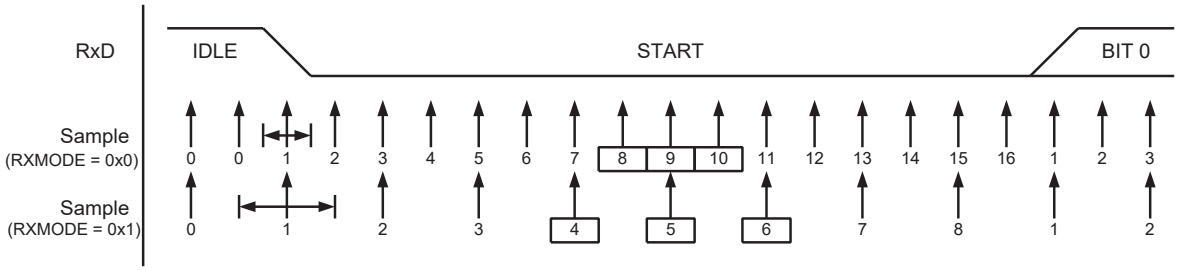
#### 27.3.3.2.1 Clock Recovery

Since there is no common clock signal when using Asynchronous mode, each communicating device generates separate clock signals. These clock signals must be configured to run at the same baud rate for the communication to take place. The devices, therefore, run at the same speed, but their timing is skewed in relation to each other. To accommodate this, the USART features a hardware clock recovery unit which synchronizes the incoming asynchronous serial frames with the internally generated baud rate clock.

The figure below illustrates the sampling process for the Start bit of an incoming frame. It shows the timing scheme for both Normal and Double-Speed mode (the RXMODE bits in the USARTn.CTRLB register configured respectively

to 0x00 and 0x01). The sample rate for Normal mode is 16 times the baud rate, while the sample rate for Double-Speed mode is eight times the baud rate (see 27.3.3.2.4 Double-Speed Operation for more details). The horizontal arrows show the maximum synchronization error. Note that the maximum synchronization error is larger in Double-Speed mode.

**Figure 27-6. Start Bit Sampling**

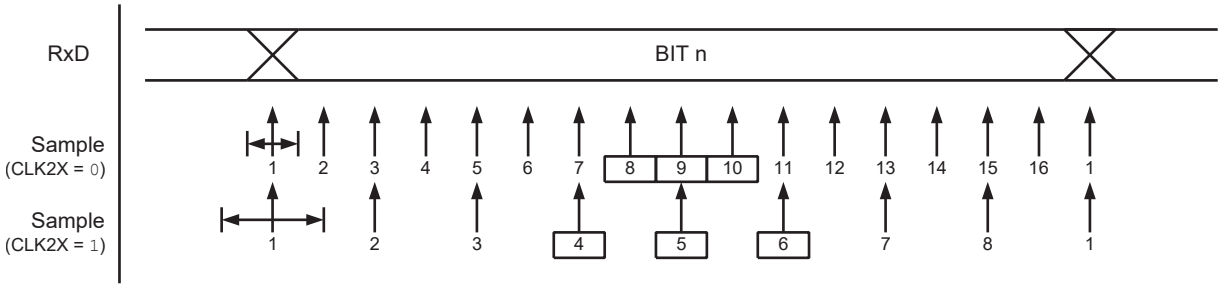



When the clock recovery logic detects a falling edge from Idle (high) state to the Start bit (low), the Start bit detection sequence is initiated. In the figure above, sample 1 denotes the first sample reading '0'. The clock recovery logic then uses three subsequent samples (samples 8, 9, and 10 in Normal mode, and samples 4, 5, 6 in Double-Speed mode) to decide if a valid Start bit is received. If two or three samples read '0', the Start bit is accepted. The clock recovery unit is synchronized, and the data recovery can begin. If less than two samples read '0', the Start bit is rejected. This process is repeated for each Start bit.

#### 27.3.3.2.2 Data Recovery

As with clock recovery, the data recovery unit samples at a rate 8 or 16 times faster than the baud rate depending on whether it is running in Double-Speed or Normal mode, respectively. The figure below shows the sampling process for reading a bit in a received frame.

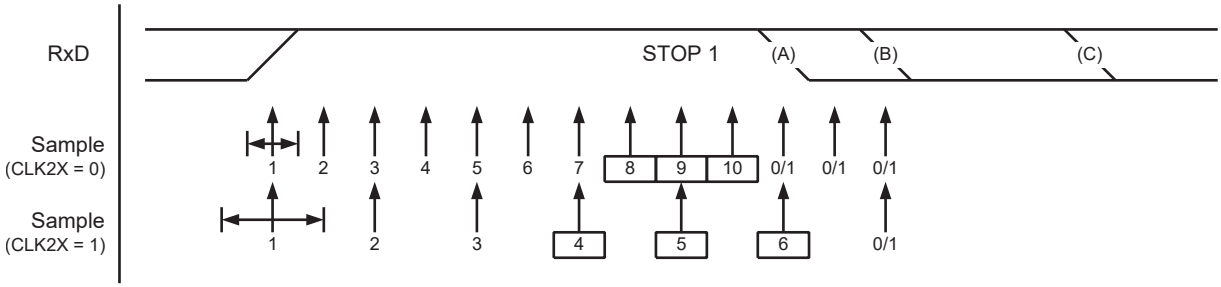
**Figure 27-7. Sampling of Data and Parity Bits**




A majority voting technique is, like with clock recovery, used on the three center samples for deciding the logic level of the received bit. The process is repeated for each bit until a complete frame is received.

The data recovery unit will only receive the first Stop bit while ignoring the rest if there are more. If the sampled Stop bit is read '0', the Frame Error flag will be set. The figure below shows the sampling of a Stop bit. It also shows the earliest possible beginning of the next frame's Start bit.

**Figure 27-8. Stop Bit and Next Start Bit Sampling**




A new high-to-low transition indicating the Start bit of a new frame can come right after the last of the bits used for majority voting. For Normal-Speed mode, the first low-level sample can be at the point marked (A) in the figure above. For Double-Speed mode the first low level must be delayed to point (B), being the first sample after the majority vote samples. Point (C) marks a Stop bit of full length at the nominal baud rate.

#### 27.3.3.2.3 Error Tolerance

The speed of the internally generated baud rate and the externally received data rate should ideally be identical, but due to natural clock source error, this is normally not the case. The USART is tolerant of such error, and the limits of this tolerance make up what is sometimes known as the Operational Range.

The following tables list the operational range of the USART, being the maximum receiver baud rate error that can be tolerated. Note that Normal-Speed mode has higher toleration of baud rate variations than Double-Speed mode.

**Table 27-4. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode**

| D | $R_{slow}$ [%] | $R_{fast}$ [%] | Maximum Total Error [%] | Recommended Max. Receiver Error [%] |
|---|---|---|---|---|
| 5 | 93.20 | 106.67 | -6.80/+6.67 | ±3.0 |
| 6 | 94.12 | 105.79 | -5.88/+5.79 | ±2.5 |
| 7 | 94.81 | 105.11 | -5.19/+5.11 | ±2.0 |
| 8 | 95.36 | 104.58 | -4.54/+4.58 | ±2.0 |
| 9 | 95.81 | 104.14 | -4.19/+4.14 | ±1.5 |
| 10 | 96.17 | 103.78 | -3.83/+3.78 | ±1.5 |

**Notes:**

- D: The sum of character size and parity size (D = 5 to 10 bits)
- $R_{SLOW}$: The ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- $R_{FAST}$: The ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate

**Table 27-5. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode**

| D | $R_{slow}$ [%] | $R_{fast}$ [%] | Maximum Total Error [%] | Recommended Max. Receiver Error [%] |
|---|---|---|---|---|
| 5 | 94.12 | 105.66 | -5.88/+5.66 | ±2.5 |
| 6 | 94.92 | 104.92 | -5.08/+4.92 | ±2.0 |
| 7 | 95.52 | 104.35 | -4.48/+4.35 | ±1.5 |
| 8 | 96.00 | 103.90 | -4.00/+3.90 | ±1.5 |
| 9 | 96.39 | 103.53 | -3.61/+3.53 | ±1.5 |
| 10 | 96.70 | 103.23 | -3.30/+3.23 | ±1.0 |

**Notes:**

- D: The sum of character size and parity size (D = 5 to 10 bits)
- $R_{SLOW}$: The ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- $R_{FAST}$: The ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate

The recommendations of the maximum receiver baud rate error were made under the assumption that the receiver and transmitter equally divide the maximum total error.

The following equations are used to calculate the maximum ratio of the incoming data rate and the internal receiver baud rate.

| | |
|---|---|
| $R_{SLOW} = \frac{S(D+1)}{S(D+1)+S_F-1}$ | $R_{FAST} = \frac{S(D+2)}{S(D+1)+S_M}$ |

- D: The sum of character size and parity size (D = 5 to 10 bits)
- S: Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double-Speed mode.
- $S_F$: First sample number used for majority voting. SF = 8 for Normal-Speed mode and SF = 4 for Double-Speed mode.

- $S_M$: Middle sample number used for majority voting. SM = 9 for Normal-Speed mode and SM = 5 for Double-Speed mode.
- $R_{SLOW}$: The ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- $R_{FAST}$: The ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate

#### 27.3.3.2.4 Double-Speed Operation

Double-speed operation allows for higher baud rates under asynchronous operation with lower peripheral clock frequencies. This operation mode is enabled by writing the RXMODE bits in the Control B (USARTn.CTRLB) register to 0x01.

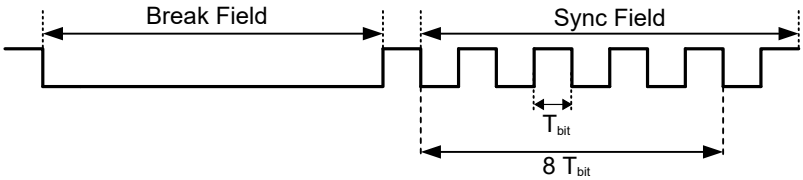
When enabled, the baud rate for a given asynchronous baud rate setting will be doubled. This is shown in the equations in 27.3.2.2.1 The Fractional Baud Rate Generator. In this mode, the receiver will use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery. This requires a more accurate baud rate setting and peripheral clock. See 27.3.3.2.3 Error Tolerance for more details.

#### 27.3.3.2.5 Auto-Baud

The auto-baud feature lets the USART configure its BAUD register based on input from a communication device. This allows the device to communicate autonomously with multiple devices communicating with different baud rates. The USART peripheral features two auto-baud modes: Generic Auto-Baud mode and LIN Constrained Auto-Baud mode.

Both auto-baud modes must receive an auto-baud frame as seen in the figure below.

**Figure 27-9. Auto-Baud Timing**




The break field is detected when 12 or more consecutive low cycles are sampled and notifies the USART that it is about to receive the synchronization field. After the break field, when the Start bit of the synchronization field is detected, a counter running at the peripheral clock speed is started. The counter is then incremented for the next eight $T_{bit}$ of the synchronization field. When all eight bits are sampled, the counter is stopped. The resulting counter value is in effect the new BAUD register value.

When the USART Receive mode is set to GENAUTO (the RXMODE bits in the USARTn.CTRLB register), the Generic Auto-Baud mode is enabled. In this mode, one can set the Wait For Break (WFB) bit in the USARTn.STATUS register to enable detection of a break field of any length (that is, also shorter than 12 cycles). This makes it possible to set an arbitrary new baud rate without knowing the current baud rate. If the measured sync field results in a valid BAUD value (0x0064 - 0xFFFF), the BAUD register is updated.

When USART Receive mode is set to LINAUTO mode (the RXMODE bits in the USARTn.CTRLB register), it follows the LIN format. The WFB functionality of the Generic Auto-Baud mode is not compatible with the LIN Constrained Auto-Baud mode. This means that the received signal must be low for 12 peripheral clock cycles or more for a break field to be valid. When a break field has been detected, the USART expects the following synchronization field character to be 0x55. The tolerance for the difference in baud rates between the two synchronizing devices can be configured using the Auto-baud Window Size (ABW) bits in the Control D (USARTn.CTRLD) register. If any of these conditions are not met, the Inconsistent Sync Field Error Flag (the ISFIF bit in the USARTn.STATUS register) is set, and the baud rate is unchanged.

#### 27.3.3.2.6 Half Duplex Operation

Half duplex is a type of communication where two or more devices may communicate with each other, but only one at a time. The USART can be configured to operate in the following half duplex modes:

- One-Wire mode
- RS-485 mode

***One-Wire Mode***

One-Wire mode is enabled by setting the Loop-Back Mode Enable (LBME) bit in the USARTn.CTRLA register. This will enable an internal connection between the TXD pin and the USART receiver, making the TXD pin a combined

TxD/RxD line. The RXD pin will be disconnected from the USART receiver and may be controlled by a different peripheral.

In One-Wire mode, multiple devices are able to manipulate the TxD/RxD line at the same time. In the case where one device drives the pin to a logical high level ($V_{CC}$), and another device pulls the line low (GND), a short will occur. To accommodate this, the USART features an Open-Drain mode (the ODME bit in the USARTn.CTRLB register) which prevents the transmitter from driving a pin to a logical high level, thereby constraining it to only be able to pull it low. Combining this function with the internal pull-up feature (the PULLUPEN bit in the PORTx.PINnCTRL register) will let the line be held high through a pull-up resistor, allowing any device to pull it low. When the line is pulled low the current from $V_{CC}$ to GND will be limited by the pull-up resistor. The TXD pin is automatically set to output by hardware when the Open-Drain mode is enabled.

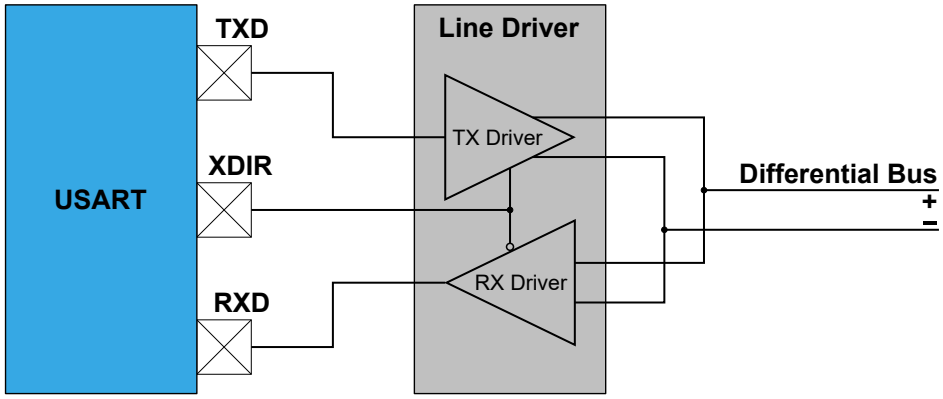
When the USART is transmitting to the TxD/RxD line, it will also receive its own transmission. This can be used to check for overlapping transmissions by checking if the received data are the same as the transmitted data as it should be.

***RS-485 Mode***

RS-485 is a communication standard supported by the USART peripheral. It is a physical interface that defines the setup of a communication circuit. Data are transmitted using differential signaling, making communication robust against noise. RS-485 is enabled by writing the RS485 bit (USARTn.CTRLA) to '`1`'.

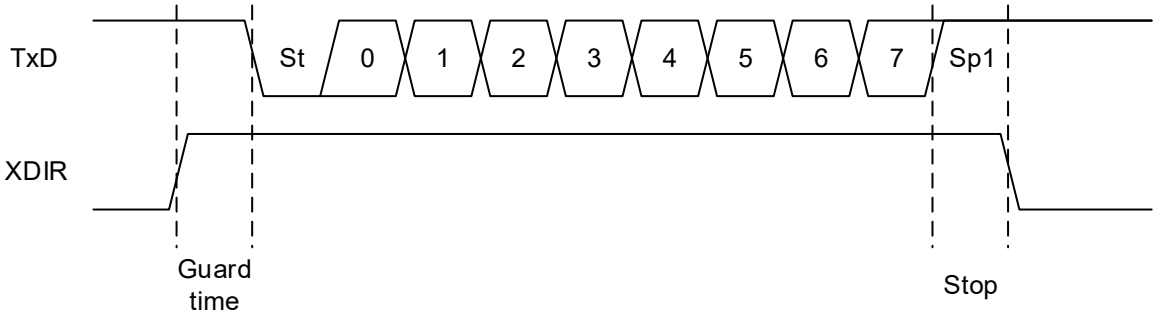
The RS-485 mode supports external line driver devices that convert a single USART transmission into corresponding differential pair signals. It implements automatic control of the XDIR pin that can be used to enable transmission or reception for the line driver device. The USART automatically drives the XDIR pin high while the USART is transmitting and pulls it low when the transmission is complete. An example of such a circuit is shown in the figure below.

**Figure 27-10. RS-485 Bus Connection**



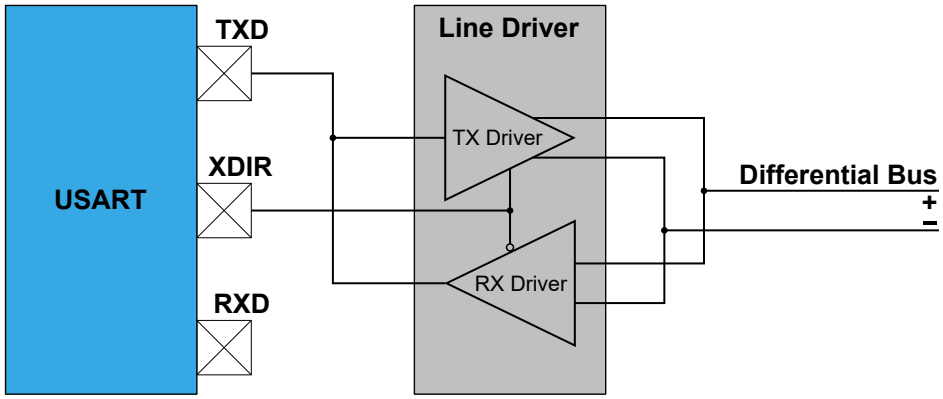

The XDIR pin goes high one baud clock cycle in advance of data being shifted out to allow some guard time to enable the external line driver. The XDIR pin will remain high for the complete frame including Stop bit(s).

**Figure 27-11. XDIR Drive Timing**




RS-485 mode is compatible with One-Wire mode. One-Wire mode enables an internal connection between the TXD pin and the USART receiver, making the TXD pin a combined TxD/RxD line. The RXD pin will be disconnected from the USART receiver and may be controlled by a different peripheral. An example of such a circuit is shown in the figure below.

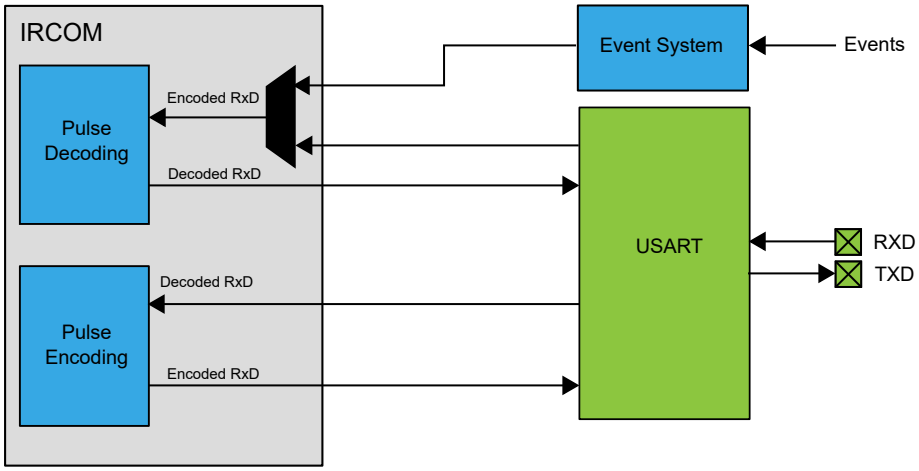
**Figure 27-12. RS-485 with Loop-Back Mode Connection**




#### 27.3.3.2.7 IRCOM Mode of Operation

The USART peripheral can be configured in Infrared Communication mode (IRCOM) which is IrDA® 1.4 compatible with baud rates up to 115.2 kbps. When enabled, the IRCOM mode enables infrared pulse encoding/decoding for the USART.

**Figure 27-13. Block Diagram**




The USART is set in IRCOM mode by writing 0x02 to the CMODE bits in the USARTn.CTRLC register. The data on the TXD/RXD pins are the inverted values of the transmitted/received infrared pulse. It is also possible to select an event channel from the Event System as an input for the IRCOM receiver. This enables the IRCOM to receive input from the I/O pins or sources other than the corresponding RXD pin. This will disable the RxD input from the USART pin.

For transmission, three pulse modulation schemes are available:

- 3/16 of the baud rate period
- Fixed programmable pulse time based on the peripheral clock frequency
- Pulse modulation disabled

For the reception, a fixed programmable minimum high-level pulse-width for the pulse to be decoded as a logical '0' is used. Shorter pulses will then be discarded, and the bit will be decoded to logical '1' as if no pulse was received.

When IRCOM mode is enabled, Double-Speed mode cannot be used for the USART.

### 27.3.4 Additional Features

#### 27.3.4.1 Parity

Parity bits can be used by the USART to check the validity of a data frame. The Parity bit is set by the transmitter based on the number of bits with the value of '1' in a transmission and controlled by the receiver upon reception. If

the Parity bit is inconsistent with the transmission frame, the receiver may assume that the data frame has been corrupted.

Even or odd parity can be selected for error checking by writing the Parity Mode (PMODE) bits in the USARTn.CTRLC register. If even parity is selected, the Parity bit is set to '1' if the number of Data bits with value '1' is odd (making the total number of bits with value '1' even). If odd parity is selected, the Parity bit is set to '1' if the number of data bits with value '1' is even (making the total number of bits with value '1' odd).

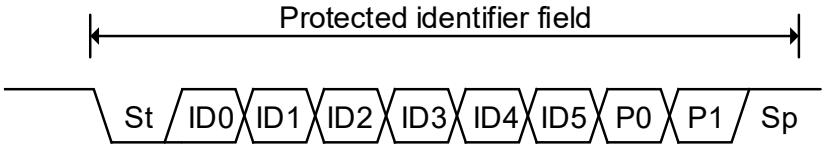
When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the Parity bit of the corresponding frame. If a parity error is detected, the Parity Error flag (the PERR bit in the USARTn.RXDATAH register) is set.

If LIN Constrained Auto-Baud mode is enabled (RXMODE = 0x03 in the USARTn.CTRLB register), a parity check is only performed on the protected identifier field. A parity error is detected if one of the equations below is not true, which sets the Parity Error flag.

$$P0 = ID0 \text{ XOR } ID1 \text{ XOR } ID2 \text{ XOR } ID4$$

$$P1 = \text{ NOT } \left(ID1 \text{ XOR } ID3 \text{ XOR } ID4 \text{ XOR } ID5\right)$$

**Figure 27-14. Protected Identifier Field and Mapping of Identifier and Parity Bits**




### 27.3.4.2 Start-of-Frame Detection

The Start-of-Frame Detection feature enables the USART to wake up from Standby sleep mode upon data reception.

When a high-to-low transition is detected on the RXD pin, the oscillator is powered up, and the USART peripheral clock is enabled. After start-up, the rest of the data frame can be received, provided that the baud rate is slow enough in relation to the oscillator start-up time. The start-up time of the oscillators varies with supply voltage and temperature. For details on oscillator start-up time characteristics, refer to the *Electrical Characteristics* section.

If a false Start bit is detected, the device will, if another wake-up source has not been triggered, go back into the Standby sleep mode.

The Start-of-Frame detection works in Asynchronous mode only. It is enabled by writing the Start-of-Frame Detection Enable (SFDEN) bit in the USARTn.CTRLB register. If a Start bit is detected while the device is in Standby sleep mode, the USART Receive Start Interrupt Flag (RXSIF) bit is set.

The USART Receive Complete Interrupt Flag (RXCIF) bit and the RXSIF bit share the same interrupt line, but each has its dedicated interrupt settings. The table below shows the USART Start Frame Detection modes, depending on the interrupt setting.

**Table 27-6. USART Start Frame Detection Modes**

| SFDEN | RXSIF Interrupt | RXCIF Interrupt | Comment |
|---|---|---|---|
| 0 | x | x | Standard mode |
| 1 | Disabled | Disabled | Only the oscillator is powered during the frame reception. If the interrupts are disabled and buffer overflow is ignored, all incoming frames will be lost |
| 1 | Disabled | Enabled | System/all clocks are awakened on Receive Complete interrupt |
| 1 | Enabled | x | System/all clocks are awakened when a Start bit is detected |

**Note:** The SLEEP instruction will not shut down the oscillator if there is ongoing communication.

### 27.3.4.3 Multiprocessor Communication

The Multiprocessor Communication mode (MPCM) effectively reduces the number of incoming frames that have to be handled by the receiver in a system with multiple microcontrollers communicating via the same serial bus. This

mode is enabled by writing a '1' to the MPCM bit in the Control B register (USARTn.CTRLB). In this mode, a dedicated bit in the frames is used to indicate whether the frame is an address or data frame type.

If the receiver is set up to receive frames that contain five to eight data bits, the first Stop bit is used to indicate the frame type. If the receiver is set up for frames with nine data bits, the ninth bit is used to indicate frame type. When the frame type bit is '1', the frame contains an address. When the frame type bit is '0', the frame is a data frame. If 5- to 8-bit character frames are used, the transmitter must be set to use two Stop bits, since the first Stop bit is used for indicating the frame type.

If a particular slave MCU has been addressed, it will receive the following data frames as usual, while the other slave MCUs will ignore the frames until another address frame is received.

#### 27.3.4.3.1 Using Multiprocessor Communication

The following procedure should be used to exchange data in Multiprocessor Communication mode (MPCM):

1. All slave MCUs are in Multiprocessor Communication mode.
2. The master MCU sends an address frame, and all slaves receive and read this frame.
3. Each slave MCU determines if it has been selected.
4. The addressed MCU will disable MPCM and receive all data frames. The other slave MCUs will ignore the data frames.
5. When the addressed MCU has received the last data frame, it must enable MPCM again and wait for a new address frame from the master.

The process then repeats from step 2.

### 27.3.5 Events

The USART can generate the events described in the table below.

**Table 27-7. Event Generators in USART**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| USARTn | XCK | The clock signal in SPI Master mode and Synchronous USART Master mode | Pulse | CLK_PER | One XCK period |

The table below describes the event user and its associated functionality.

**Table 27-8. Event Users in USART**

| User Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| USARTn | IREI | USARTn IrDA event input | Pulse | Sync |

### 27.3.6 Interrupts

**Table 27-9. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| RXC | Receive Complete interrupt | • There is unread data in the receive buffer (RXCIE)<br>• Receive of Start-of-Frame detected (RXSIE)<br>• Auto-Baud Error/ISFIF flag set (ABEIE) |
| DRE | Data Register Empty interrupt | The transmit buffer is empty/ready to receive new data (DREIE) |
| TXC | Transmit Complete interrupt | The entire frame in the Transmit Shift register has been shifted out and there are no new data in the transmit buffer (TXCIE) |

When an Interrupt condition occurs, the corresponding Interrupt flag is set in the STATUS register (USARTn.STATUS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the Control A register (USARTn.CTRLA).

An interrupt request is generated when the corresponding interrupt source is enabled, and the Interrupt flag is set. The interrupt request remains active until the Interrupt flag is cleared. See the USARTn.STATUS register for details on how to clear Interrupt flags.

## 27.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | RXDATAL | 7:0 | DATA[7:0] | | | | | | | |
| 0x01 | RXDATAH | 7:0 | RXCIF | BUFOVF | | | | FERR | PERR | DATA[8] |
| 0x02 | TXDATAL | 7:0 | DATA[7:0] | | | | | | | |
| 0x03 | TXDATAH | 7:0 | | | | | | | | DATA[8] |
| 0x04 | STATUS | 7:0 | RXCIF | TXCIF | DREIF | RXSIF | ISFIF | | BDF | WFB |
| 0x05 | CTRLA | 7:0 | RXCIE | TXCIE | DREIE | RXSIE | LBME | ABEIE | | RS485 |
| 0x06 | CTRLB | 7:0 | RXEN | TXEN | | SFDEN | ODME | RXMODE[1:0] | | MPCM |
| 0x07 | CTRLC | 7:0 | CMODE[1:0] | | PMODE[1:0] | | SBMODE | CHSIZE[2:0] | | |
| 0x07 | CTRLC | 7:0 | CMODE[1:0] | | | | | UDORD | UCPHA | |
| 0x08 | BAUD | 7:0 | BAUD[7:0] | | | | | | | |
| | | 15:8 | BAUD[15:8] | | | | | | | |
| 0x0A | CTRLD | 7:0 | ABW[1:0] | | | | | | | |
| 0x0B | DBGCTRL | 7:0 | | | | | | | | DBGRUN |
| 0x0C | EVCTRL | 7:0 | | | | | | | | IREI |
| 0x0D | TXPLCTRL | 7:0 | TXPL[7:0] | | | | | | | |
| 0x0E | RXPLCTRL | 7:0 | | RXPL[6:0] | | | | | | |

## 27.5 Register Description

### 27.5.1 Receiver Data Register Low Byte

**Name:** RXDATAL
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

This register contains the eight LSbs of the data received by the USART receiver. The USART receiver is double-buffered, and this register always represents the data for the oldest received frame. If the data for only one frame is present in the receive buffer, this register contains that data.

The buffer shifts out the data either when RXDATAL or RXDATAH is read, depending on the configuration. The register which does not lead to data being shifted should be read first to be to able read both bytes before shifting.

When the Character Size (CHSIZE) bits in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), a read of RXDATAH shifts the receive buffer. Otherwise, RXDATAL shifts the buffer.

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

**Bits 7:0 – DATA[7:0]** Receiver Data Register

### 27.5.2 Receiver Data Register High Byte

**Name:** RXDATAH
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

This register contains the MSb of the data received by the USART receiver, as well as status bits reflecting the status of the received data frame. The USART receiver is double-buffered, and this register always represents the data and status bits for the oldest received frame. If the data and status bits for only one frame is present in the receive buffer, this register contains that data.

The buffer shifts out the data either when RXDATAL or RXDATAH is read, depending on the configuration. The register which does not lead to data being shifted should be read first to be able to read both bytes before shifting.

When the Character Size (CHSIZE) bits in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), a read of RXDATAH shifts the receive buffer. Otherwise, RXDATAL shifts the buffer.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXCIF | BUFOVF | | | | FERR | PERR | DATA[8] |
| Access | R | R | | | | R | R | R |
| Reset | 0 | 0 | | | | 0 | 0 | 0 |

**Bit 7 – RXCIF** USART Receive Complete Interrupt Flag
This flag is set when there are unread data in the receive buffer and cleared when the receive buffer is empty.

**Bit 6 – BUFOVF** Buffer Overflow
This flag is set if a buffer overflow is detected. A buffer overflow occurs when the receive buffer is full, a new frame is waiting in the receive shift register, and a new Start bit is detected. This flag is cleared when the Receiver Data (USARTn.RXDATAL and USARTn.RXDATAH) registers are read.
This flag is not used in the Master SPI mode of operation.

**Bit 2 – FERR** Frame Error
This flag is set if the first Stop bit is '0' and cleared when it is correctly read as '1'.
This flag is not used in the Master SPI mode of operation.

**Bit 1 – PERR** Parity Error
This flag is set if parity checking is enabled, and the received data has a parity error. This flag is otherwise cleared. For details on parity calculation, refer to 27.3.4.1 Parity.
This flag is not used in the Master SPI mode of operation.

**Bit 0 – DATA[8]** Receiver Data Register
When using a 9-bit frame size, this bit holds the ninth bit (MSb) of the received data.
When the Receiver Mode (RXMODE) bits in the Control B (USARTn.CTRLB) register is configured to LIN Constrained Auto-Baud (LINAUTO) mode, this bit indicates if the received data are within the response space of a LIN frame. This bit is cleared if the received data are in the protected identifier field and is otherwise set.

### 27.5.3 Transmit Data Register Low Byte

**Name:** TXDATAL
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

The data written to this register is automatically loaded into the dedicated shift register. The shift register outputs each of the bits serially to the TXD pin.

When using a 9-bit frame size, the ninth bit (MSb) should be written to USARTn.TXDATAH. In that case, the buffer shifts data either when TXDATAL or TXDATAH is written, depending on the configuration. The register which does not lead to data being shifted should be written first to be able to write both registers before shifting.

When the Character Size (CHSIZE) bits in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), a write of TXDATAH shifts the transmit buffer. Otherwise, TXDATAL shifts the buffer.

This register may only be written when the Data Register Empty Interrupt Flag (DREIF) in the Status (USARTn.STATUS) register is set.

| 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 7:0 – DATA[7:0]** Transmit Data Register

### 27.5.4 Transmit Data Register High Byte

**Name:** TXDATAH
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

The data written to this register is automatically loaded into the dedicated shift register. The shift register outputs each of the bits serially to the TXD pin.

When using a 9-bit frame size, the ninth bit (MSb) should be written to USARTn.TXDATAH. In that case, the buffer shifts data either when TXDATAL or TXDATAH is written, depending on the configuration. The register which does not lead to data being shifted should be written first to be able to write both bytes before shifting.

When the Character Size (CHSIZE) bits in the Control C (USARTn.CTRLC) register is configured to 9-bit (low byte first), a write of TXDATAH shifts the transmit buffer. Otherwise, TXDATAL shifts the buffer.

This register may only be written when the Data Register Empty Interrupt Flag (DREIF) in the Status (USARTn.STATUS) register is set.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | DATA[8] |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – DATA[8]** Transmit Data Register

### 27.5.5 USART Status Register

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXCIF | TXCIF | DREIF | RXSIF | ISFIF | | BDF | WFB |
| Access | R | R/W | R | R/W | R/W | | R/W | W |
| Reset | 0 | 0 | 1 | 0 | 0 | | 0 | 0 |

**Bit 7 – RXCIF** USART Receive Complete Interrupt Flag
This flag is set when there are unread data in the receive buffer and cleared when the receive buffer is empty.

**Bit 6 – TXCIF** USART Transmit Complete Interrupt Flag
This flag is set when the entire frame in the Transmit Shift register has been shifted out, and there are no new data in the transmit buffer (TXDATAL and TXDATAH) registers. It is cleared by writing a '1' to it.

**Bit 5 – DREIF** USART Data Register Empty Interrupt Flag
This flag is set when the transmit buffer (TXDATAL and TXDATAH) registers are empty and cleared when they contain data that has not yet been moved into the transmit shift register.

**Bit 4 – RXSIF** USART Receive Start Interrupt Flag
This flag is set when Start-of-Frame detection is enabled, the device is in Standby sleep mode, and a valid start bit is detected. It is cleared by writing a '1' to it.
This flag is not used in the Master SPI mode operation.

**Bit 3 – ISFIF** Inconsistent Synchronization Field Interrupt Flag
This flag is set if an auto-baud mode is enabled, and the synchronization field is too short or too long to give a valid baud setting. It will also be set when USART is set to LINAUTO mode, and the SYNC character differs from data value 0x55. This flag is cleared by writing a '1' to it. See section 27.3.3.2.5 Auto-Baud for more information.

**Bit 1 – BDF** Break Detected Flag
This flag is set if an auto-baud mode is enabled and a valid break and synchronization character is detected, and is cleared when the next data is received. It can also be cleared by writing a '1' to it. See section 27.3.3.2.5 Auto-Baud for more information.

**Bit 0 – WFB** Wait For Break
This bit controls whether the Wait For Break feature is enabled or not. Refer to section 27.3.3.2.5 Auto-Baud for more information.

| Value | Description |
|---|---|
| 0 | Wait For Break is disabled |
| 1 | Wait For Break is enabled |

### 27.5.6 Control A

**Name:** CTRLA
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXCIE | TXCIE | DREIE | RXSIE | LBME | ABEIE | | RS485 |
| 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 – RXCIE** Receive Complete Interrupt Enable
This bit controls whether the Receive Complete Interrupt is enabled or not. When enabled, the interrupt will be triggered when the RXCIF bit in the USARTn.STATUS register is set.

| Value | Description |
|---|---|
| 0 | The Receive Complete Interrupt is disabled |
| 1 | The Receive Complete Interrupt is enabled |

**Bit 6 – TXCIE** Transmit Complete Interrupt Enable
This bit controls whether the Transmit Complete Interrupt is enabled or not. When enabled, the interrupt will be triggered when the TXCIF bit in the USARTn.STATUS register is set.

| Value | Description |
|---|---|
| 0 | The Transmit Complete Interrupt is disabled |
| 1 | The Transmit Complete Interrupt is enabled |

**Bit 5 – DREIE** Data Register Empty Interrupt Enable
This bit controls whether the Data Register Empty Interrupt is enabled or not. When enabled, the interrupt will be triggered when the DREIF bit in the USARTn.STATUS register is set.

| Value | Description |
|---|---|
| 0 | The Data Register Empty Interrupt is disabled |
| 1 | The Data Register Empty Interrupt is enabled |

**Bit 4 – RXSIE** Receiver Start Frame Interrupt Enable
This bit controls whether the Receiver Start Frame Interrupt is enabled or not. When enabled, the interrupt will be triggered when the RXSIF bit in the USARTn.STATUS register is set.

| Value | Description |
|---|---|
| 0 | The Receiver Start Frame Interrupt is disabled |
| 1 | The Receiver Start Frame Interrupt is enabled |

**Bit 3 – LBME** Loop-back Mode Enable
This bit controls whether the Loop-back mode is enabled or not. When enabled, an internal connection between the TXD pin and the USART receiver is created, and the input from the RXD pin to the USART receiver is disconnected.

| Value | Description |
|---|---|
| 0 | Loop-back mode is disabled |
| 1 | Loop-back mode is enabled |

**Bit 2 – ABEIE** Auto-baud Error Interrupt Enable
This bit controls whether the Auto-baud Error Interrupt is enabled or not. When enabled, the interrupt will be triggered when the ISFIF bit in the USARTn.STATUS register is set.

| Value | Description |
|---|---|
| 0 | The Auto-baud Error Interrupt is disabled |
| 1 | The Auto-baud Error Interrupt is enabled |

**Bit 0 – RS485** RS-485 Mode
This bit controls whether the RS-485 mode is enabled or not. Refer to section RS-485 Mode for more information.

| Value | Description |
|---|---|
| 0 | RS-485 mode is disabled |
| 1 | RS-485 mode is enabled |

### 27.5.7 Control B

**Name:** CTRLB
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXEN | TXEN | | SFDEN | ODME | RXMODE[1:0] | | MPCM |
| 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 – RXEN** Receiver Enable
This bit controls whether the USART receiver is enabled or not. Refer to section 27.3.2.4.2 Disabling the Receiver for more information.

| Value | Description |
|---|---|
| 0 | The USART receiver is disabled |
| 1 | The USART receiver is enabled |

**Bit 6 – TXEN** Transmitter Enable
This bit controls whether the USART transmitter is enabled or not. Refer to section 27.3.2.3.1 Disabling the Transmitter for more information.

| Value | Description |
|---|---|
| 0 | The USART transmitter is disabled |
| 1 | The USART transmitter is enabled |

**Bit 4 – SFDEN** Start-of-Frame Detection Enable
This bit controls whether the USART Start-of-Frame Detection mode is enabled or not. Refer to section 27.3.4.2 Start-of-Frame Detection for more information.

| Value | Description |
|---|---|
| 0 | The USART Start-of-Frame Detection mode is disabled |
| 1 | The USART Start-of-Frame Detection mode is enabled |

**Bit 3 – ODME** Open Drain Mode Enable
This bit controls whether Open Drain mode is enabled or not. See section One-Wire Mode for more information.

| Value | Description |
|---|---|
| 0 | Open Drain mode is disabled |
| 1 | Open Drain mode is enabled |

**Bits 2:1 – RXMODE[1:0]** Receiver Mode
Writing these bits selects the receiver mode of the USART.

- Writing the bits to 0x00 enables Normal-Speed (NORMAL) mode. When the USART Communication Mode (CMODE) bits in the Control C (USARTn.CTRLC) register is configured to Asynchronous USART (ASYNCHRONOUS) or Infrared Communication (IRCOM), the RXMODE bits should always be written to 0x00.
- Writing the bits to 0x01 enables Double-Speed (CLK2X) mode. Refer to section 27.3.3.2.4 Double-Speed Operation for more information.
- Writing the bits to 0x02 enables Generic Auto-Baud (GENAUTO) mode. Refer to section 27.3.3.2.5 Auto-Baud for more information.
- Writing the bits to 0x03 enables Lin Constrained Auto-Baud (LINAUTO) mode. Refer to section 27.3.3.2.5 Auto-Baud for more information.

| Value | Name | Description |
|---|---|---|
| 0x00 | NORMAL | Normal-Speed mode |
| 0x01 | CLK2X | Double-Speed mode |
| 0x02 | GENAUTO | Generic Auto-Baud mode |
| 0x03 | LINAUTO | LIN Constrained Auto-Baud mode |

**Bit 0 – MPCM** Multi-Processor Communication Mode
This bit controls whether the Multi-Processor Communication mode is enabled or not. Refer to section 27.3.4.3 Multiprocessor Communication for more information.

| Value | Description |
|---|---|
| 0 | Multi-Processor Communication mode is disabled |
| 1 | Multi-Processor Communication mode is enabled |

### 27.5.8 Control C - Normal Mode

**Name:** CTRLC
**Offset:** 0x07
**Reset:** 0x03
**Property:** -

This register description is valid for all modes except the Master SPI mode. When the USART Communication Mode bits (CMODE) in this register are written to 'MSPI', see CTRLC - Master SPI mode for the correct description.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMODE[1:0] | | PMODE[1:0] | | SBMODE | CHSIZE[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 | 1 | 1 |

**Bits 7:6 – CMODE[1:0]** USART Communication Mode
These bits select the communication mode of the USART.
Writing a 0x03 to these bits alters the available bit fields in this register, see CTRLC - Master SPI mode.

| Value | Name | Description |
|---|---|---|
| 0x00 | ASYNCHRONOUS | Asynchronous USART |
| 0x01 | SYNCHRONOUS | Synchronous USART |
| 0x02 | IRCOM | Infrared Communication |
| 0x03 | MSPI | Master SPI |

**Bits 5:4 – PMODE[1:0]** Parity Mode
These bits enable and select the type of parity generation. See section 27.3.4.1 Parity for more information.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLED | Disabled |
| 0x1 | - | Reserved |
| 0x2 | EVEN | Enabled, even parity |
| 0x3 | ODD | Enabled, odd parity |

**Bit 3 – SBMODE** Stop Bit Mode
This bit selects the number of Stop bits to be inserted by the transmitter.
The receiver ignores this setting.

| Value | Description |
|---|---|
| 0 | 1 Stop bit |
| 1 | 2 Stop bits |

**Bits 2:0 – CHSIZE[2:0]** Character Size
These bits select the number of data bits in a frame. The receiver and transmitter use the same setting. For 9BIT character size, the order of which byte to read or write first, low or high byte of RXDATA or TXDATA, can be configured.

| Value | Name | Description |
|---|---|---|
| 0x00 | 5BIT | 5-bit |
| 0x01 | 6BIT | 6-bit |
| 0x02 | 7BIT | 7-bit |
| 0x03 | 8BIT | 8-bit |
| 0x04 | - | Reserved |
| 0x05 | - | Reserved |
| 0x06 | 9BITL | 9-bit (Low byte first) |
| 0x07 | 9BITH | 9-bit (High byte first) |

### 27.5.9 Control C - Master SPI Mode

**Name:** CTRLC
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

This register description is valid only when the USART is in Master SPI mode (CMODE written to MSPI). For other CMODE values, see CTRLC - Normal Mode.

See 27.3.3.1.3 USART in Master SPI Mode for a full description of the Master SPI mode operation.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMODE[1:0] | | | | | UDORD | UCPHA | |
| Access | R/W | R/W | | | | R/W | R/W | |
| Reset | 0 | 0 | | | | 0 | 0 | |

**Bits 7:6 – CMODE[1:0]** USART Communication Mode
These bits select the communication mode of the USART.
Writing a value different than 0x03 to these bits alters the available bit fields in this register, see CTRLC - Normal Mode.

| Value | Name | Description |
|---|---|---|
| 0x00 | ASYNCHRONOUS | Asynchronous USART |
| 0x01 | SYNCHRONOUS | Synchronous USART |
| 0x02 | IRCOM | Infrared Communication |
| 0x03 | MSPI | Master SPI |

**Bit 2 – UDORD** USART Data Order
This bit controls the frame format. The receiver and transmitter use the same setting. Changing the setting of the UDORD bit will corrupt all ongoing communication for both the receiver and the transmitter.

| Value | Description |
|---|---|
| 0 | MSb of the data word is transmitted first |
| 1 | LSb of the data word is transmitted first |

**Bit 1 – UCPHA** USART Clock Phase
This bit controls the phase of the interface clock. Refer to section Clock Generation for more information.

| Value | Description |
|---|---|
| 0 | Data are sampled on the leading (first) edge |
| 1 | Data are sampled on the trailing (last) edge |

### 27.5.10 Baud Register

**Name:** BAUD
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

The USARTn.BAUDL and USARTn.BAUDH register pair represents the 16-bit value, USARTn.BAUD. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

Ongoing transmissions of the transmitter and receiver will be corrupted if the baud rate is changed. Writing to this register will trigger an immediate update of the baud rate prescaler. For more information on how to set the baud rate, see Table 27-1, Equations for Calculating Baud Rate Register Setting.

| 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 15:8 – BAUD[15:8]** USART Baud Rate High Byte
These bits hold the MSB of the 16-bit Baud register.

**Bits 7:0 – BAUD[7:0]** USART Baud Rate Low Byte
These bits hold the LSB of the 16-bit Baud register.

### 27.5.11 Control D

**Name:** CTRLD
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

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

**Bits 7:6 – ABW[1:0]** Auto-baud Window Size
These bits control the tolerance for the difference between the baud rates between the two synchronizing devices when using Lin Constrained Auto-baud mode. The tolerance is based on the number of baud samples between every two bits. When baud rates are identical, there should be 32 baud samples between each bit pair since each bit is sampled 16 times.

| Value | Name | Description |
|---|---|---|
| 0x00 | WDW0 | 32±6 (18% tolerance) |
| 0x01 | WDW1 | 32±5 (15% tolerance) |
| 0x02 | WDW2 | 32±7 (21% tolerance) |
| 0x03 | WDW3 | 32±8 (25% tolerance) |

### 27.5.12 Debug Control Register

**Name:** DBGCTRL
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Debug Run

| Value | Description |
|---|---|
| 0 | The peripheral is halted in Break Debug mode and ignores events. |
| 1 | The peripheral will continue to run in Break Debug mode when the CPU is halted. |

### 27.5.13 IrDA Control Register

**Name:** EVCTRL
**Offset:** 0x0C
**Reset:** 0x00
**Property:** -

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

**Bit 0 – IREI** IrDA Event Input Enable
This bit controls whether the IrDA event input is enabled or not. See section 27.3.3.2.7 IRCOM Mode of Operation for more information.

| Value | Description |
|---|---|
| 0 | IrDA Event input is enabled |
| 1 | IrDA Event input is disabled |

### 27.5.14 IRCOM Transmitter Pulse Length Control Register

**Name:** TXPLCTRL
**Offset:** 0x0D
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXPL[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 – TXPL[7:0]** Transmitter Pulse Length
This 8-bit value sets the pulse modulation scheme for the transmitter. Setting this register will have effect only if IRCOM mode is selected by the USART, and it must be configured before the USART transmitter is enabled (TXEN).

| Value | Description |
|---|---|
| 0x00 | 3/16 of the baud rate period pulse modulation is used |
| 0x01-0xFE | Fixed pulse length coding is used. The 8-bit value sets the number of peripheral clock periods for the pulse. The start of the pulse will be synchronized with the rising edge of the baud rate clock. |
| 0xFF | Pulse coding disabled. RX and TX signals pass through the IRCOM module unaltered. This enables other features through the IRCOM module, such as half-duplex USART, loop-back testing, and USART RX input from an event channel. |

### 27.5.15 IRCOM Receiver Pulse Length Control Register

**Name:** RXPLCTRL
**Offset:** 0x0E
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | RXPL[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 6:0 – RXPL[6:0]** Receiver Pulse Length
This 7-bit value sets the filter coefficient for the IRCOM transceiver. Setting this register will only have effect if IRCOM mode is selected by a USART, and it must be configured before the USART receiver is enabled (RXEN).

| Value | Description |
|---|---|
| 0x00 | Filtering disabled |
| 0x01-0x7F | Filtering enabled. The value of RXPL+1 represents the number of samples required for a received pulse to be accepted. |

# 28. SPI - Serial Peripheral Interface

## 28.1 Features

- Full Duplex, Three-Wire Synchronous Data Transfer
- Master or Slave Operation
- LSb First or MSb First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double-Speed (CK/2) Master SPI Mode

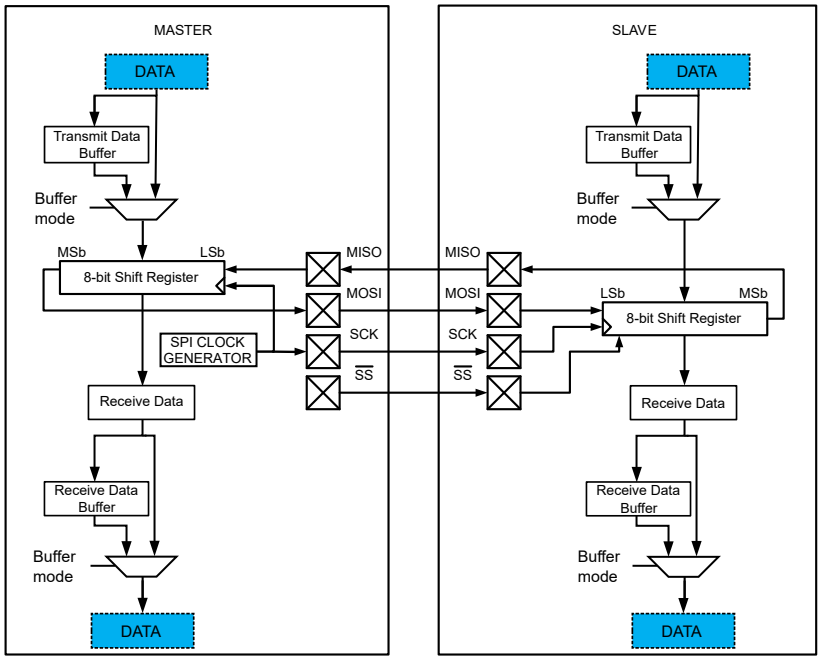
## 28.2 Overview

The Serial Peripheral Interface (SPI) is a high-speed synchronous data transfer interface using three or four pins. It allows full duplex communication between an AVR® device and peripheral devices, or between several microcontrollers. The SPI peripheral can be configured as either master or slave. The master initiates and controls all data transactions.

The interconnection between master and slave devices with SPI is shown in the block diagram. The system consists of two shift registers and a master clock generator. The SPI master initiates the communication cycle by pulling the desired slave's Slave Select ($\overline{SS}$) signal low. The master and slave prepare the data to be sent to their respective shift registers, and the master generates the required clock pulses on the SCK line to exchange data. Data are always shifted from master to slave on the master output, slave input (MOSI) line, and from slave to master on the master input, slave output (MISO) line.

### 28.2.1 Block Diagram

**Figure 28-1. SPI Block Diagram**




The SPI is built around an 8-bit shift register that will shift data out and in at the same time. The Transmit Data register and the Receive Data register are not physical registers but are mapped to other registers when written or

read: Writing the Transmit Data (SPIn.DATA) register will write the shift register in Normal mode and the Transmit Buffer register in Buffer mode. Reading the Receive Data (SPIn.DATA) register will read the Receive Data register in Normal mode and the Receive Data Buffer in Buffer mode.

In Master mode, the SPI has a clock generator to generate the SCK clock. In Slave mode, the received SCK clock is synchronized and sampled to trigger the shifting of data in the shift register.

### 28.2.2 Signal Description

**Table 28-1. Signals in Master and Slave Mode**

| Signal | Description | Pin Configuration | |
|---|---|---|---|
| | | Master Mode | Slave Mode |
| MOSI | Master Out Slave In | User defined(1) | Input |
| MISO | Master In Slave Out | Input | User defined(1,2) |
| SCK | Serial Clock | User defined(1) | Input |
| $\overline{SS}$ | Slave Select | User defined(1) | Input |

**Notes:**

1. If the pin data direction is configured as output, the pin level is controlled by the SPI.
2. If the SPI is in Slave mode and the MISO pin data direction is configured as output, the $\overline{SS}$ pin controls the MISO pin output in the following way:
   - If the $\overline{SS}$ pin is driven low, the MISO pin is controlled by the SPI.
   - If the $\overline{SS}$ pin is driven high, the MISO pin is tri-stated.

When the SPI module is enabled, the pin data direction for the signals marked with “Input” in Table 28-1 is overridden.

## 28.3 Functional Description

### 28.3.1 Initialization

Initialize the SPI to a basic functional state by following these steps:

1. Configure the $\overline{SS}$ pin in the port peripheral.
2. Select the SPI master/slave operation by writing the Master/Slave Select (MASTER) bit in the Control A (SPIn.CTRLA) register.
3. In Master mode, select the clock speed by writing the Prescaler (PRESC) bits and the Clock Double (CLK2X) bit in SPIn.CTRLA.
4. Optional: Select the Data Transfer mode by writing to the MODE bits in the Control B (SPIn.CTRLB) register.
5. Optional: Write the Data Order (DORD) bit in SPIn.CTRLA.
6. Optional: Set up the Buffer mode by writing the BUFEN and BUFWR bits in the Control B (SPIn.CTRLB) register.
7. Optional: To disable the multi-master support in Master mode, write ‘`1`’ to the Slave Select Disable (SSD) bit in SPIn.CTRLB.
8. Enable the SPI by writing a ‘`1`’ to the ENABLE bit in SPIn.CTRLA.

### 28.3.2 Operation

#### 28.3.2.1 Master Mode Operation

When the SPI is configured in Master mode, a write to the SPIn.DATA register will start a new transfer. The SPI master can operate in two modes, Normal and Buffer, as explained below.

##### 28.3.2.1.1 Normal Mode

In Normal mode, the system is single-buffered in the transmit direction and double-buffered in the receive direction. This influences the data handling in the following ways:

1. New bytes to be sent cannot be written to the DATA (SPIn.DATA) register before the entire transfer has completed. A premature write will cause corruption of the transmitted data and the Write Collision (WRCOL) flag in SPIn.INTFLAGS will be set.
2. Received bytes are written to the Receive Data Buffer register immediately after the transmission is completed.
3. The Receive Data Buffer register has to be read before the next transmission is completed, or the data will be lost. This register is read by reading SPIn.DATA.
4. The Transmit Data Buffer and Receive Data Buffer registers are not used in Normal mode.

After a transfer has completed, the Interrupt Flag (IF) will be set in the Interrupt Flags (SPIn.INTFLAGS) register. This will cause the corresponding interrupt to be executed if this interrupt and the global interrupts are enabled. Setting the Interrupt Enable (IE) bit in the Interrupt Control (SPIn.INTCTRL) register will enable the interrupt.

#### 28.3.2.1.2 Buffer Mode

The Buffer mode is enabled by writing the BUFEN bit in the SPIn.CTRLB register to '1'. The BUFWR bit in SPIn.CTRLB has no effect in Master mode. In Buffer mode, the system is double-buffered in the transmit direction and triple-buffered in the receive direction. This influences the data handling the following ways:

1. New bytes can be written to the DATA (SPIn.DATA) register as long as the Data Register Empty Interrupt Flag (DREIF) in the Interrupt Flag (SPIn.INTFLAGS) register is set. The first write will be transmitted right away, and the following write will go to the Transmit Data Buffer register.
2. A received byte is placed in a two-entry Receive First-In, First-Out (RX FIFO) queue comprised of the Receive Data register and Receive Data Buffer immediately after the transmission is completed.
3. The DATA register is used to read from the RX FIFO. The RX FIFO must be read at least every second transfer to avoid any loss of data.

When both the shift register and the Transmit Data Buffer register become empty, the Transfer Complete Interrupt Flag (TXCIF) in the Interrupt Flags (SPIn.INTFLAGS) register will be set. This will cause the corresponding interrupt to be executed if this interrupt and the global interrupts are enabled. Setting the Transfer Complete Interrupt Enable (TXCIE) in the Interrupt Control (SPIn.INTCTRL) register enables the Transfer Complete Interrupt.

#### 28.3.2.1.3 $\overline{SS}$ Pin Functionality in Master Mode - Multi-Master Support

In Master mode, the Slave Select Disable (SSD) bit in the Control B (SPIn.CTRLB) register controls how the SPI uses the $\overline{SS}$ pin.

- If SSD in SPIn.CTRLB is '0', the SPI can use the $\overline{SS}$ pin to transition from Master to Slave mode. This allows multiple SPI masters on the same SPI bus.
- If SSD in SPIn.CTRLB is '0', and the $\overline{SS}$ pin is configured as an output pin, it can be used as a regular I/O pin or by other peripheral modules, and will not affect the SPI system.
- If SSD in SPIn.CTRLB is '1', the SPI does not use the $\overline{SS}$ pin, and it can be used as a regular I/O pin, or by other peripheral modules.

If the SSD bit in SPIn.CTRLB is '0', and the $\overline{SS}$ is configured as an input pin, the $\overline{SS}$ pin must be held high to ensure Master SPI operation. A low level will be interpreted as another Master is trying to take control of the bus. This will switch the SPI into Slave mode, and the hardware of the SPI will perform the following actions:

1. The Master (MASTER) bit in the SPI Control A (SPIn.CTRLA) register is cleared, and the SPI system becomes a slave. The direction of the SPI pins will be switched when the conditions in Table 28-2 are met.
2. The Interrupt Flag (IF) bit in the Interrupt Flags (SPIn.INTFLAGS) register will be set. If the interrupt is enabled and the global interrupts are enabled, the interrupt routine will be executed.

**Table 28-2. Overview of the $\overline{SS}$ Pin Functionality when the SSD Bit in SPIn.CTRLB is '0'**

| $\overline{SS}$ Configuration | $\overline{SS}$ Pin-Level | Description |
|---|---|---|
| Input | High | Master activated (selected) |
| | Low | Master deactivated, switched to Slave mode |
| Output | High | Master activated (selected) |
| | Low | |

**Note:** If the device is in Master mode and it cannot be ensured that the $\overline{SS}$ pin will stay high between two transmissions, the status of the Master (MASTER) bit in SPIn.CTRLA has to be checked before a new byte is written. After the Master bit has been cleared by a low level on the $\overline{SS}$ line, it must be set by the application to re-enable the SPI Master mode.

#### 28.3.2.2 Slave Mode

In Slave mode, the SPI peripheral receives SPI clock and Slave Select from a Master. Slave mode supports three operational modes: One Normal mode and two configurations for the Buffered mode. In Slave mode, the control logic will sample the incoming signal on the SCK pin. To ensure correct sampling of this clock signal, the minimum low and high periods must each be longer than two peripheral clock cycles.

##### 28.3.2.2.1 Normal Mode

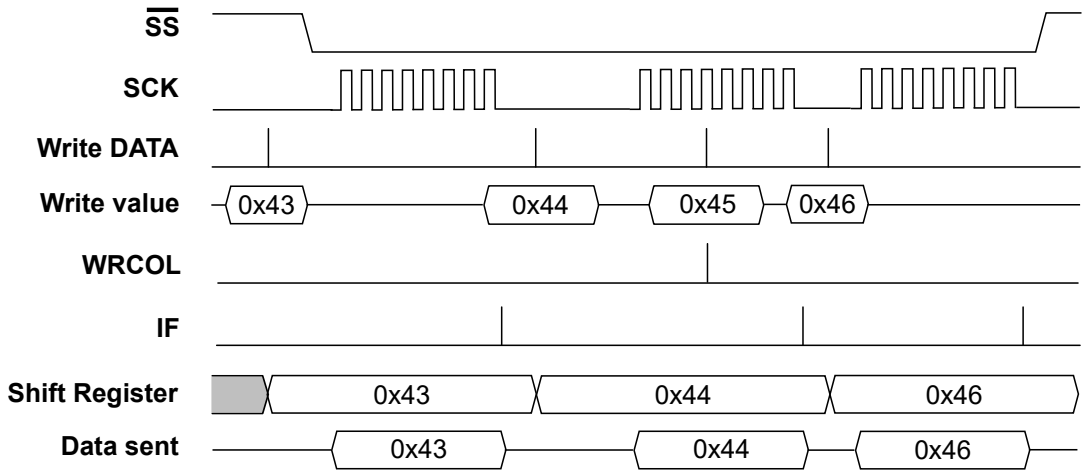
In Normal mode, the SPI peripheral will remain Idle as long as the $\overline{SS}$ pin is driven high. In this state, the software may update the contents of the DATA register, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. If the $\overline{SS}$ pin is driven low, the slave will start to shift out data on the first SCK clock pulse. When one byte has been completely shifted, the SPI Interrupt Flag (IF) in SPIn.INTFLAGS is set.

The user application may continue placing new data to be sent into the DATA register before reading the incoming data. New bytes to be sent cannot be written to the DATA register before the entire transfer has completed. A premature write will be ignored, and the hardware will set the Write Collision (WRCOL) flag in SPIn.INTFLAGS.

When the $\overline{SS}$ pin is driven high, the SPI logic is halted, and the SPI slave will not receive any new data. Any partially received packet in the shift register will be lost.

Figure 28-2 shows a transmission sequence in Normal mode. Notice how the value 0x45 is written to the DATA register but never transmitted.

**Figure 28-2. SPI Timing Diagram in Normal Mode (Buffer Mode Not Enabled)**




The figure above shows three transfers and one write to the DATA register while the SPI is busy with a transfer. This write will be ignored, and the Write Collision (WRCOL) flag in SPIn.INTFLAGS is set.

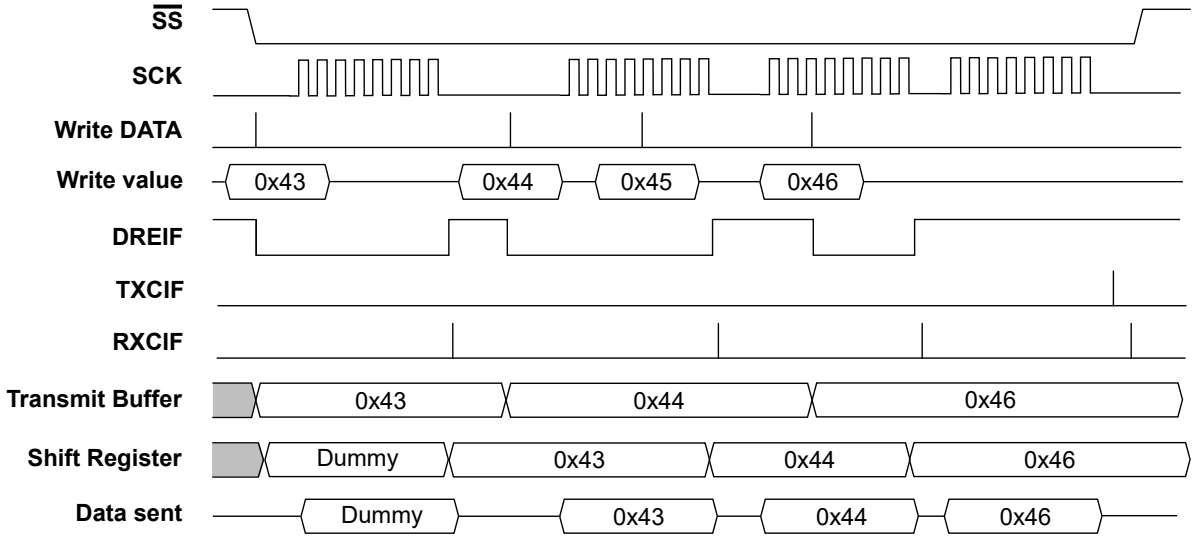
##### 28.3.2.2.2 Buffer Mode

To avoid data collisions, the SPI peripheral can be configured in Buffered mode by writing a '1' to the Buffer Mode Enable (BUFEN) bit in the Control B (SPIn.CTRLB) register. In this mode, the SPI has additional interrupt flags and extra buffers. The extra buffers are shown in Figure 28-1. There are two different modes for the Buffer mode, selected with the Buffer mode Wait for Receive (BUFWR) bit. The two different modes are described below with timing diagrams.

**Slave Buffer Mode with Wait for Receive Bit Written to '0'**

In Slave mode, if the Buffer mode Wait for Receive (BUFWR) bit in SPIn.CTRLB is written to '0', a dummy byte will be sent before the transmission of user data starts. Figure 28-3 shows a transmission sequence with this configuration. Notice how the value 0x45 is written to the Data (SPIn.DATA) register but never transmitted.

**Figure 28-3. SPI Timing Diagram in Buffer Mode with BUFWR in SPIn.CTRLB Written to '0'**



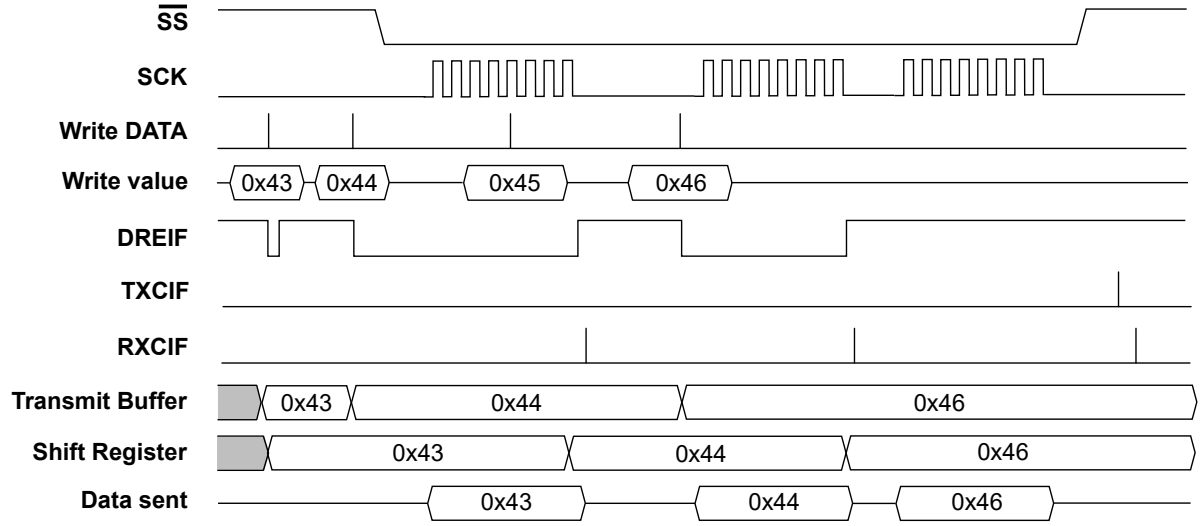

When the Wait for Receive (BUFWR) bit in SPIn.CTRLB is written to '0', all writes to the Data (SPIn.DATA) register go to the Transmit Data Buffer register. The figure above shows that the value 0x43 is written to the Data (SPIn.DATA) register, but it is not immediately transferred to the shift register, so the first byte sent will be a dummy byte. The value of the dummy byte equals the values that were in the shift register at the time. After the first dummy transfer is completed, the value 0x43 is transferred to the shift register. Then 0x44 is written to the Data (SPIn.DATA) register and goes to the Transmit Data Buffer register. A new transfer is started, and 0x43 will be sent. The value 0x45 is written to the Data (SPIn.DATA) register, but the Transmit Data Buffer register is not updated since it is already full containing 0x44 and the Data Register Empty Interrupt Flag (DREIF) in SPIn.INTFLAGS is low. The value 0x45 will be lost. After the transfer, the value 0x44 is moved to the shift register. During the next transfer, 0x46 is written to the Data (SPIn.DATA) register, and 0x44 is sent out. After the transfer is complete, 0x46 is copied into the shift register and sent out in the next transfer.

The DREIF goes low every time the Transmit Data Buffer register is written and goes high after a transfer when the previous value in the Transmit Data Buffer register is copied into the shift register. The Receive Complete Interrupt Flag (RXCIF) in SPIn.INTFLAGS is set one cycle after the DREIF goes high. The Transfer Complete Interrupt Flag is set one cycle after the Receive Complete Interrupt Flag is set when both the value in the shift register and the Transmit Data Buffer register have been sent.

**Slave Buffer Mode with Wait for Receive Bit Written to '1'**

In Slave mode, if the Buffer mode Wait for Receive (BUFWR) bit in SPIn.CRTLB is written to '1', the transmission of user data starts as soon as the $\overline{SS}$ pin is driven low. Figure 28-4 shows a transmission sequence with this configuration. Notice how the value 0x45 is written to the Data (SPIn.DATA) register but never transmitted.

**Figure 28-4. SPI Timing Diagram in Buffer Mode with CTRLB.BUFWR Written to '1'**




All writes to the Data (SPIn.DATA) register go to the Transmit Data Buffer register. The figure above shows that the value 0x43 is written to the Data (SPIn.DATA) register, and since the $\overline{SS}$ pin is high, it is copied to the shift register in the next cycle. Then the next write (0x44) will go to the Transmit Data Buffer register. During the first transfer, the value 0x43 will be shifted out. In the figure above, the value 0x45 is written to the Data (SPIn.DATA) register, but the Transmit Data Buffer register is not updated since the DREIF is low. After the transfer is completed, the value 0x44 from the Transmit Data Buffer register is copied to the shift register. The value 0x46 is written to the Transmit Data Buffer register. During the next two transfers, 0x44 and 0x46 are shifted out. The flags behave identically to the Buffer Mode Wait for Receive (BUFWR) bit in SPIn.CTRLB set to '0'.

#### 28.3.2.2.3 $\overline{SS}$ Pin Functionality in Slave Mode

The Slave Select ($\overline{SS}$) pin plays a central role in the operation of the SPI. Depending on the mode the SPI is in, and the configuration of this pin, it can be used to activate or deactivate devices. The $\overline{SS}$ pin is used as a Chip Select pin.

In Slave mode, $\overline{SS}$, MOSI, and SCK are always inputs. The behavior of the MISO pin depends on the configured data direction of the pin in the port peripheral and the value of $\overline{SS}$. When the $\overline{SS}$ pin is driven low, the SPI is activated and will respond to received SCK pulses by clocking data out on MISO, if the user has configured the data direction of the MISO pin as output. When the $\overline{SS}$ pin is driven high, the SPI is deactivated, meaning that it will not receive incoming data. If the MISO pin data direction is configured as output, the MISO pin will be tri-stated. Table 28-3 shows an overview of the $\overline{SS}$ pin functionality.

**Table 28-3. Overview of the $\overline{SS}$ Pin Functionality**

| $\overline{SS}$ Configuration | $\overline{SS}$ Pin-Level | Description | MISO Pin Mode | |
|---|---|---|---|---|
| | | | Port Direction = Output | Port Direction = Input |
| Always Input | High | Slave deactivated (deselected) | Tri-stated | Input |
| | Low | Slave activated (selected) | Output | Input |

**Note:** In Slave mode, the SPI state machine will be reset when the $\overline{SS}$ pin is driven high. If the $\overline{SS}$ pin is driven high during a transmission, the SPI will stop sending and receiving data immediately and both data received and data sent must be considered lost. As the $\overline{SS}$ pin is used to signal the start and end of a transfer, it is useful for achieving packet/byte synchronization and keeping the Slave bit counter synchronized with the master clock generator.

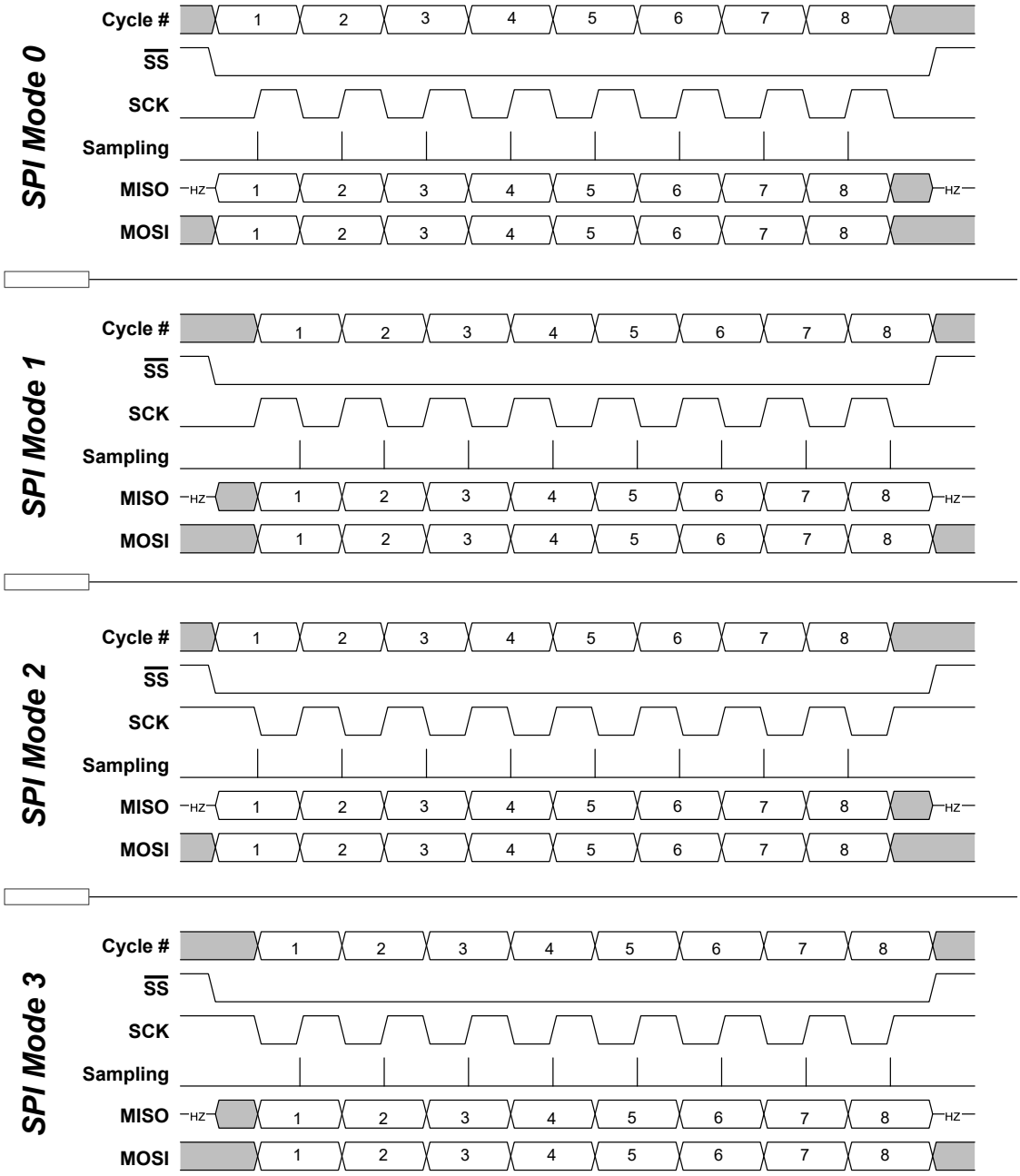
#### 28.3.2.3 Data Modes

There are four combinations of SCK phase and polarity with respect to serial data. The desired combination is selected by writing to the MODE bits in the Control B (SPIn.CTRLB) register.

The SPI data transfer formats are shown below. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize.

The leading edge is the first clock edge of a clock cycle. The trailing edge is the last clock edge of a clock cycle.

**Figure 28-5. SPI Data Transfer Modes**




### 28.3.2.4 Events

The SPI can generate the following events:

**Table 28-4. Event Generators in SPI**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Module | Event | | | | |
| SPIn | SCK | SPI Master clock | Level | CLK_PER | Minimum two CLK_PER periods |

The SPI has no event users.

Refer to the *Event System* section for more details regarding event types and Event System configuration.

#### 28.3.2.5 Interrupts

**Table 28-5. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions | |
|---|---|---|---|
| | | Normal Mode | Buffer Mode |
| SPIn | SPI interrupt | • IF: Interrupt Flag interrupt<br>• WRCOL: Write Collision interrupt | • SSI: Slave Select Trigger Interrupt<br>• DRE: Data Register Empty interrupt<br>• TXC: Transfer Complete interrupt<br>• RXC: Receive Complete interrupt |

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

## 28.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | DORD | MASTER | CLK2X | | PRESC[1:0] | | ENABLE |
| 0x01 | CTRLB | 7:0 | BUFEN | BUFWR | | | | SSD | MODE[1:0] | |
| 0x02 | INTCTRL | 7:0 | RXCIE | TXCIE | DREIE | SSIE | | | | IE |
| 0x03 | INTFLAGS | 7:0 | IF | WRCOL | | | | | | |
| 0x03 | INTFLAGS | 7:0 | RXCIF | TXCIF | DREIF | SSIF | | | | BUFOVF |
| 0x04 | DATA | 7:0 | DATA[7:0] | | | | | | | |

## 28.5 Register Description

### 28.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | DORD | MASTER | CLK2X | | PRESC[1:0] | | ENABLE |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 6 – DORD** Data Order

| Value | Description |
|---|---|
| 0 | The MSb of the data word is transmitted first |
| 1 | The LSb of the data word is transmitted first |

**Bit 5 – MASTER** Master/Slave Select
This bit selects the desired SPI mode.
If $\overline{SS}$ is configured as input and driven low while this bit is '1', then this bit is cleared and the IF in SPIn.INTFLAGS is set. The user has to write MASTER = 1 again to re-enable SPI Master mode.
This behavior is controlled by the Slave Select Disable (SSD) bit in SPIn.CTRLB.

| Value | Description |
|---|---|
| 0 | SPI Slave mode selected |
| 1 | SPI Master mode selected |

**Bit 4 – CLK2X** Clock Double
When this bit is written to '1' the SPI speed (SCK frequency, after internal prescaler) is doubled in Master mode.

| Value | Description |
|---|---|
| 0 | SPI speed (SCK frequency) is not doubled |
| 1 | SPI speed (SCK frequency) is doubled in Master mode |

**Bits 2:1 – PRESC[1:0]** Prescaler
This bit field controls the SPI clock rate configured in Master mode. These bits have no effect in Slave mode. The relationship between SCK and the peripheral clock frequency ($f_{CLK_PER}$) is shown below.
The output of the SPI prescaler can be doubled by writing the CLK2X bit to '1'.

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV4 | CLK_PER/4 |
| 0x1 | DIV16 | CLK_PER/16 |
| 0x2 | DIV64 | CLK_PER/64 |
| 0x3 | DIV128 | CLK_PER/128 |

**Bit 0 – ENABLE** SPI Enable

| Value | Description |
|---|---|
| 0 | SPI is disabled |
| 1 | SPI is enabled |

### 28.5.2 Control B

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BUFEN | BUFWR | | | | SSD | MODE[1:0] | |
| Access | R/W | R/W | | | | R/W | R/W | R/W |
| Reset | 0 | 0 | | | | 0 | 0 | 0 |

**Bit 7 – BUFEN** Buffer Mode Enable
Writing this bit to '1' enables Buffer mode. This will enable two receive buffers and one transmit buffer. Both will have separate interrupt flags, transmit complete and receive complete.

**Bit 6 – BUFWR** Buffer Mode Wait for Receive
When writing this bit to '0' the first data transferred will be a dummy sample.

| Value | Description |
|---|---|
| 0 | One SPI transfer must be completed before the data are copied into the shift register. |
| 1 | If writing to the Data register when the SPI is enabled and $\overline{SS}$ is high, the first write will go directly to the shift register. |

**Bit 2 – SSD** Slave Select Disable
If this bit is set when operating as SPI Master (MASTER = 1 in SPIn.CTRLA), $\overline{SS}$ does not disable Master mode.

| Value | Description |
|---|---|
| 0 | Enable the Slave Select line when operating as SPI master |
| 1 | Disable the Slave Select line when operating as SPI master |

**Bits 1:0 – MODE[1:0]** Mode
These bits select the Transfer mode. The four combinations of SCK phase and polarity with respect to the serial data are shown below. These bits decide whether the first edge of a clock cycle (leading edge) is rising or falling and whether data setup and sample occur on the leading or trailing edge. When the leading edge is rising, the SCK signal is low when idle, and when the leading edge is falling, the SCK signal is high when idle.

| Value | Name | Description |
|---|---|---|
| 0x0 | 0 | Leading edge: Rising, sample<br>Trailing edge: Falling, setup |
| 0x1 | 1 | Leading edge: Rising, setup<br>Trailing edge: Falling, sample |
| 0x2 | 2 | Leading edge: Falling, sample<br>Trailing edge: Rising, setup |
| 0x3 | 3 | Leading edge: Falling, setup<br>Trailing edge: Rising, sample |

### 28.5.3 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXCIE | TXCIE | DREIE | SSIE | | | | IE |
| Access | R/W | R/W | R/W | R/W | | | | R/W |
| Reset | 0 | 0 | 0 | 0 | | | | 0 |

**Bit 7 – RXCIE** Receive Complete Interrupt Enable
In Buffer mode, this bit enables the Receive Complete interrupt. The enabled interrupt will be triggered when the RXCIF in the SPIn.INTFLAGS register is set. In the Non-Buffer mode, this bit is '0'.

**Bit 6 – TXCIE** Transfer Complete Interrupt Enable
In Buffer mode, this bit enables the Transfer Complete interrupt. The enabled interrupt will be triggered when the TXCIF in the SPIn.INTFLAGS register is set. In the Non-Buffer mode, this bit is '0'.

**Bit 5 – DREIE** Data Register Empty Interrupt Enable
In Buffer mode, this bit enables the Data Register Empty interrupt. The enabled interrupt will be triggered when the DREIF in the SPIn.INTFLAGS register is set. In the Non-Buffer mode, this bit is '0'.

**Bit 4 – SSIE** Slave Select Trigger Interrupt Enable
In Buffer mode, this bit enables the Slave Select interrupt. The enabled interrupt will be triggered when the SSIF in the SPIn.INTFLAGS register is set. In the Non-Buffer mode, this bit is '0'.

**Bit 0 – IE** Interrupt Enable
This bit enables the SPI interrupt when the SPI is not in Buffer mode. The enabled interrupt will be triggered when RXCIF/IF is set in the SPIn.INTFLAGS register.

### 28.5.4 Interrupt Flags - Normal Mode

**Name:** INTFLAGS
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IF | WRCOL | | | | | | |
| Access | R/W | R/W | | | | | | |
| Reset | 0 | 0 | | | | | | |

**Bit 7 – IF** Interrupt Flag
This flag is set when a serial transfer is complete, and one byte is completely shifted in/out of the SPIn.DATA register. If $\overline{SS}$ is configured as input and is driven low when the SPI is in Master mode, this will also set this flag. The IF is cleared by hardware when executing the corresponding interrupt vector. Alternatively, the IF can be cleared by first reading the SPIn.INTFLAGS register when IF is set, and then accessing the SPIn.DATA register.

**Bit 6 – WRCOL** Write Collision
The WRCOL flag is set if the SPIn.DATA register is written before a complete byte has been shifted out. This flag is cleared by first reading the SPIn.INTFLAGS register when WRCOL is set, and then accessing the SPIn.DATA register.

### 28.5.5 Interrupt Flags - Buffer Mode

**Name:** INTFLAGS
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXCIF | TXCIF | DREIF | SSIF | | | | BUFOVF |
| Access | R/W | R/W | R/W | R/W | | | | R/W |
| Reset | 0 | 0 | 0 | 0 | | | | 0 |

**Bit 7 – RXCIF** Receive Complete Interrupt Flag
This flag is set when there are unread data in the Receive Data Buffer register and cleared when the Receive Data Buffer register is empty (that is, it does not contain any unread data).
When interrupt-driven data reception is used, the Receive Complete Interrupt routine must read the received data from the DATA register in order to clear RXCIF. If not, a new interrupt will occur directly after the return from the current interrupt. This flag can also be cleared by writing a '1' to its bit location.

**Bit 6 – TXCIF** Transfer Complete Interrupt Flag
This flag is set when all the data in the Transmit shift register has been shifted out, and there is no new data in the transmit buffer (SPIn.DATA). The flag is cleared by writing a '1' to its bit location.

**Bit 5 – DREIF** Data Register Empty Interrupt Flag
This flag indicates whether the Transmit Data Buffer register is ready to receive new data. The flag is '1' when the transmit buffer is empty and '0' when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. The DREIF is cleared after a Reset to indicate that the transmitter is ready.
The DREIF is cleared by writing to DATA. When interrupt-driven data transmission is used, the Data Register Empty Interrupt routine must either write new data to DATA in order to clear DREIF or disable the Data Register Empty interrupt. If not, a new interrupt will occur directly after the return from the current interrupt.

**Bit 4 – SSIF** Slave Select Trigger Interrupt Flag
This flag indicates that the SPI has been in Master mode and the $\overline{SS}$ pin has been pulled low externally, so the SPI is now working in Slave mode. The flag will only be set if the Slave Select Disable (SSD) bit is not '1'. The flag is cleared by writing a '1' to its bit location.

**Bit 0 – BUFOVF** Buffer Overflow
This flag indicates data loss due to a Receive Data Buffer full condition. This flag is set if a Buffer Overflow condition is detected. A Buffer Overflow occurs when the receive buffer is full (two bytes), and a third byte has been received in the shift register. If there is no transmit data, the Buffer Overflow will not be set before the start of a new serial transfer. This flag is cleared when the DATA register is read, or by writing a '1' to its bit location.

### 28.5.6 Data

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

| 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 7:0 – DATA[7:0]** SPI Data
The DATA register is used for sending and receiving data. Writing to the register initiates the data transmission when in Master mode, while preparing data for sending in Slave mode. The byte written to the register shifts out on the SPI output line when a transaction is initiated.
The SPIn.DATA register is not a physical register. Depending on what mode is configured, it is mapped to other registers as described below.

- Normal mode:
  - Writing the DATA register will write the shift register
  - Reading from DATA will read from the Receive Data register
- Buffer mode:
  - Writing the DATA register will write to the Transmit Data Buffer register.
  - Reading from DATA will read from the Receive Data Buffer register. The contents of the Receive Data register will then be moved to the Receive Data Buffer register.

# 29. TWI - Two-Wire Interface

## 29.1 Features

- Two-Wire Communication Interface
- Philips $I^2C$ Compatible
  - Standard mode
  - Fast mode
  - Fast mode Plus
- System Management Bus (SMBus) 2.0 Compatible
  - Support arbitration between Start/repeated Start and data bit
  - Slave arbitration allows support for the Address Resolution Protocol (ARP)
  - Configurable SMBus Layer 1 time-outs in hardware
  - Independent time-outs for Dual mode
- Independent Master and Slave Operation
  - Combined (same pins) or Dual mode (separate pins)
  - Single or multi-master bus operation with full arbitration support
- Hardware Support for Slave Address Match
  - Operates in all Sleep modes
  - 7-bit address recognition
  - General call address recognition
  - Support for address range masking or secondary address match
- Input Filter for Bus Noise Suppression
- Smart Mode Support

## 29.2 Overview

The Two-Wire Interface (TWI) is a bidirectional, two-wire communication interface (bus) with a Serial Data Line (SDA) and a Serial Clock Line (SCL).

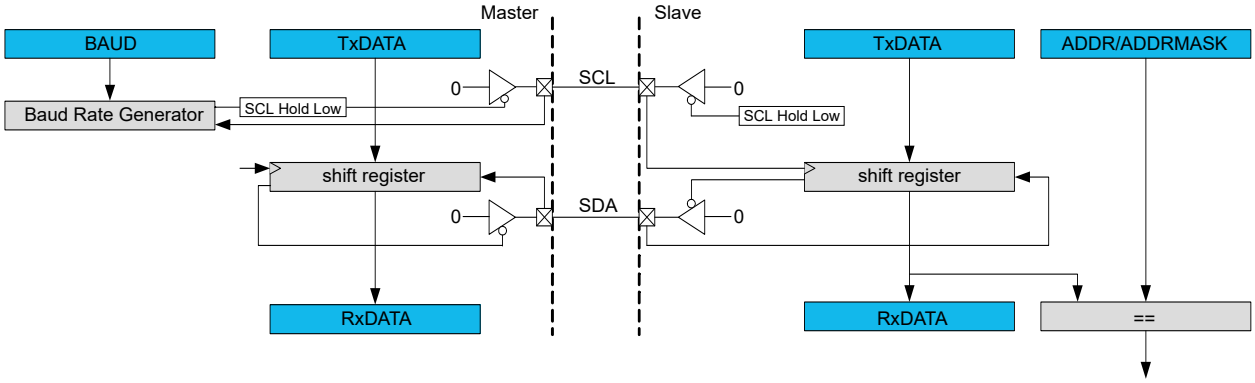
The TWI bus connects one or several slave devices to one or several master devices. Any device connected to the bus can act as a master, a slave, or both. The master generates the SCL by using a Baud Rate Generator (BRG) and initiates data transactions by addressing one slave and telling whether it wants to transmit or receive data. The BRG is capable of generating the Standard mode (Sm) and Fast mode (Fm, Fm+) bus frequencies from 100 kHz up to 1 MHz.

The TWI will detect Start and Stop conditions, bus collisions and bus errors. Arbitration lost, errors, collision, and clock hold are also detected and indicated in separate status flags available in both Master and Slave modes.

The TWI supports multi-master bus operation and arbitration. An arbitration scheme handles the case where more than one master tries to transmit data at the same time. The TWI also supports Smart mode, which can auto-trigger operations and thus reduce software complexity. The TWI supports Dual mode with simultaneous master and slave operations, which are implemented as independent units with separate enabling and configuration. The TWI supports Quick Command mode where the master can address a slave without exchanging data.

### 29.2.1 Block Diagram

**Figure 29-1. TWI Block Diagram**




### 29.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| SCL | Serial Clock Line | Digital I/O |
| SDA | Serial Data Line | Digital I/O |

## 29.3 Functional Description

### 29.3.1 General TWI Bus Concepts

The TWI provides a simple, bidirectional, two-wire communication bus consisting of:

- Serial Data Line (SDA) for packet transfer
- Serial Clock Line (SCL) for the bus clock

The two lines are open-collector lines (wired-AND).

The TWI bus topology is a simple and efficient method of interconnecting multiple devices on a serial bus. A device connected to the bus can be a master or a slave. Only master devices can control the bus and the bus communication.

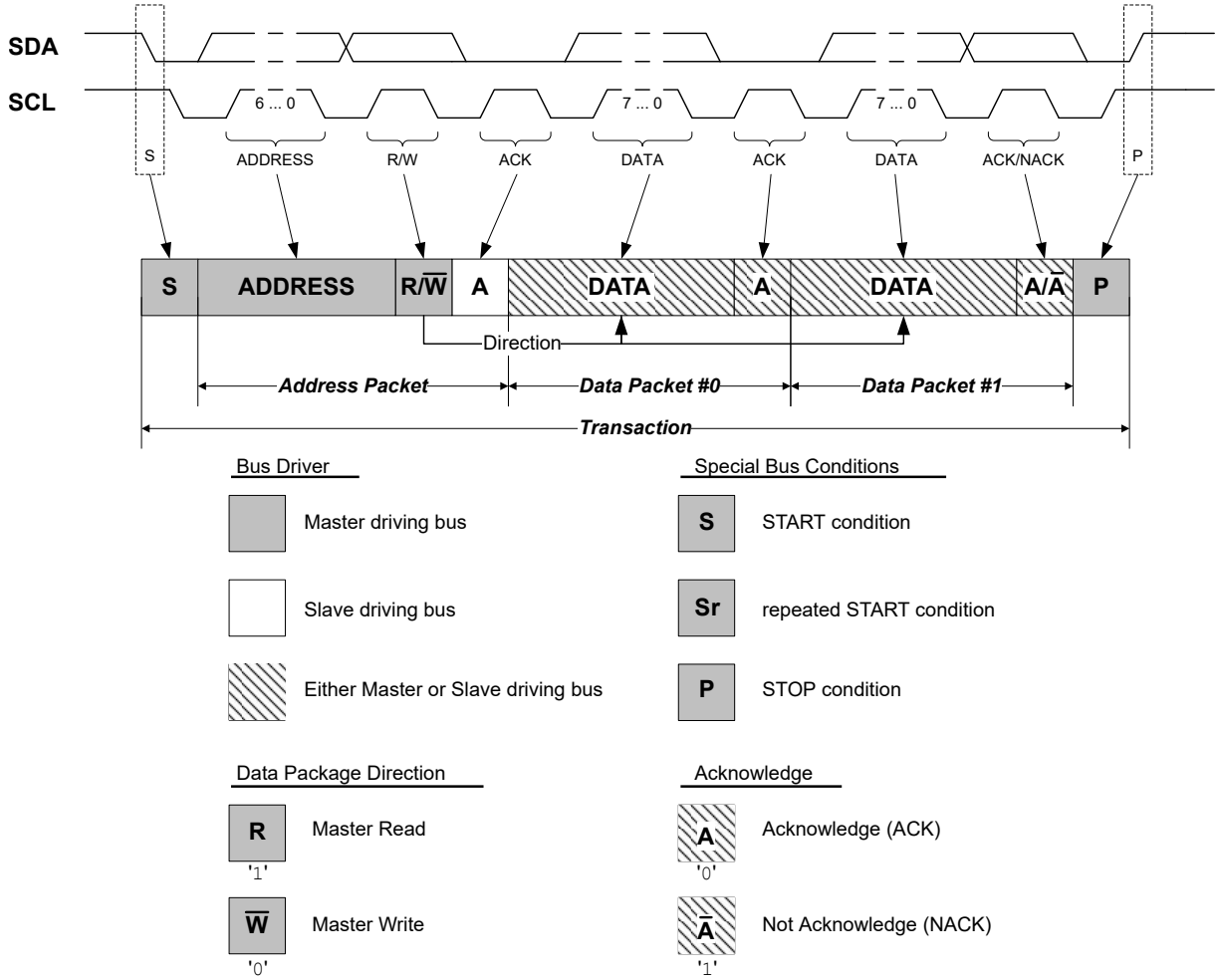
A unique address is assigned to each slave device connected to the bus, and the master will use it to control the slave and initiate a transaction. Several masters can be connected to the same bus. This is called a multi-master environment. An arbitration mechanism is provided for resolving bus ownership among masters, since only one master device may own the bus at any given time.

A master indicates the start of a transaction by issuing a Start condition (S) on the bus. The master provides the clock signal for the transaction. An address packet with a 7-bit slave address (ADDRESS) and a direction bit, representing whether the master wishes to read or write data (R/$\overline{W}$), are then sent.

The addressed $I^2C$ slave will then acknowledge (ACK) the address, and data packet transactions can begin. Every 9-bit data packet consists of eight data bits followed by a 1-bit reply indicating whether the data was acknowledged or not by the receiver.

After all the data packets (DATA) are transferred, the master issues a Stop condition (P) on the bus to end the transaction.

**Figure 29-2. Basic TWI Transaction Diagram Topology for a 7-bit Address Bus**




## 29.3.2 TWI Basic Operation

### 29.3.2.1 Initialization

If used, the following bits must be configured before enabling the TWI device:

- The SDA Setup Time (SDASETUP) bit from the Control A (TWIn.CTRLA) register
- The SDA Hold Time (SDAHOLD) bit field from the Control A (TWIn.CTRLA) register
- The FM Plus Enable (FMPEN) bit from the Control A (TWIn.CTRLA) register

#### 29.3.2.1.1 Master Initialization

The Master Baud Rate (TWIn.MBAUD) register must be written to a value that will result in a valid TWI bus clock frequency. Writing a '`1`' to the Enable TWI Master (ENABLE) bit in the Master Control A (TWIn.MCTRLA) register will enable the TWI master. The Bus State (BUSSTATE) bit field from the Master Status (TWIn.MSTATUS) register must be set to `0x1`, to force the bus state to Idle.

#### 29.3.2.1.2 Slave Initialization

The address of the slave must be written in the Slave Address (TWIn.SADDR) register. Writing a '`1`' to the Enable TWI Slave (ENABLE) bit in the Slave Control A (TWIn.SCTRLA) register will enable the TWI slave. The slave device will wait for a master device to issue a Start condition and the matching slave address.

### 29.3.2.2 TWI Master Operation

The TWI master is byte-oriented, with an optional interrupt after each byte. There are separate interrupt flags for the master write and read operation. Interrupt flags can also be used for polled operation. There are dedicated status flags for indicating ACK/NACK received, bus error, arbitration lost, clock hold, and bus state.

When an interrupt flag is set to '1', the SCL is forced low. This will give the master time to respond or handle any data, and will, in most cases, require software interaction. Clearing the interrupt flags releases the SCL. The number of interrupts generated is kept to a minimum by an automatic handling of most conditions.

#### 29.3.2.2.1 Clock Generation

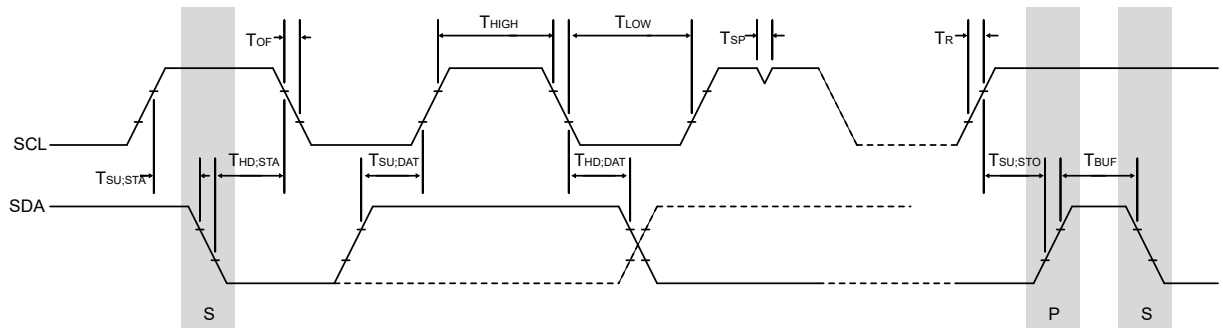
The TWI supports several transmission modes with different frequency limitations:

- Standard mode (Sm) up to 100 kHz
- Fast mode (Fm) up to 400 kHz
- Fast mode Plus (Fm+) up to 1 MHz

The Master Baud Rate (TWIn.MBAUD) register must be written to a value that will result in a TWI bus clock frequency equal to or less than those frequency limits, depending on the transmission mode.

The low ($T_{LOW}$) and high ($T_{HIGH}$) times are determined by the Master Baud Rate (TWIn.MBAUD) register, while the rise ($T_R$) and fall ($T_{OF}$) times are determined by the bus topology.

**Figure 29-3. SCL Timing**




- $T_{LOW}$ is the low period of SCL clock
- $T_{HIGH}$ is the high period of SCL clock
- $T_R$ is determined by the bus impedance; for internal pull-ups. Refer to *Electrical Characteristics* for details.
- $T_{OF}$ is the output fall time and is determined by the open-drain current limit and bus impedance. Refer to *Electrical Characteristics* for details.

**Properties of the SCL Clock**

The SCL frequency is given by:

$$f_{\mathrm{SCL}} = \frac{1}{T_{\mathrm{LOW}} + T_{\mathrm{HIGH}} + T_{\mathrm{OF}} + T_{\mathrm{R}}} [\mathrm{Hz}]$$

The SCL clock is designed to have a 50/50 duty cycle, where the low period of the duty cycle comprises of $T_{OF}$ and $T_{LOW}$. $T_{HIGH}$ will not start until a high state of SCL has been detected. The BAUD bit field in the TWIn.MBAUD register and the SCL frequency are related by the following formula:

$$f_{SCL} = \frac{f_{CLK_PER}}{10 + 2 \times BAUD + f_{CLK_PER} \times T_R} \qquad (1)$$

Equation 1 can be transformed to express BAUD:

$$BAUD = \frac{f_{CLK_PER}}{2 \times f_{SCL}} - \left(5 + \frac{f_{CLK_PER} \times T_R}{2}\right) \qquad (2)$$

**Calculation of the BAUD Value**

To ensure operation within the specifications of the desired speed mode (Sm, Fm, Fm+), follow these steps:

1. Calculate a value for the BAUD bit field using equation 2
2. Calculate $T_{LOW}$ using the BAUD value from step 1:

$$T_{LOW} = \frac{BAUD + 5}{f_{CLK_PER}} - T_{OF} \qquad (3)$$

3. Check if your $T_{LOW}$ from equation 3 is above the specified minimum of the desired mode ($T_{LOW_Sm}$= 4700 ns, $T_{LOW_Fm}$= 1300 ns, $T_{LOW_Fm+}$= 500 ns)
   - If the calculated $T_{LOW}$ is above the limit, use the BAUD value from equation 2
   - If the limit is not met, calculate a new BAUD value using equation 4 below, where $T_{LOW_mode}$ is either $T_{LOW_Sm}$, $T_{LOW_Fm}$, or $T_{LOW_Fm+}$ from the mode specifications:

$$BAUD = f_{CLK_PER} \times (T_{LOW_mode} + T_{OF}) - 5 \qquad (4)$$

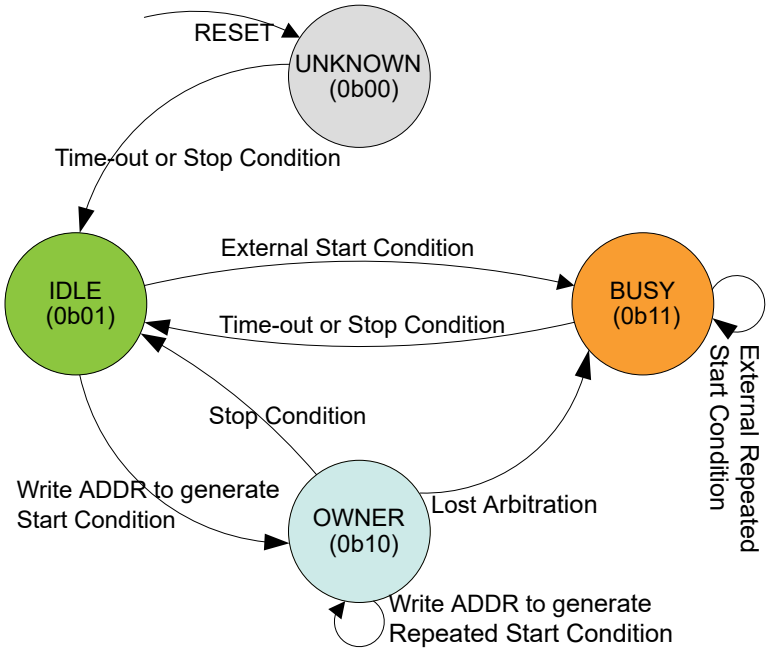
#### 29.3.2.2.2 TWI Bus State Logic

The bus state logic continuously monitors the activity on the TWI bus when the master is enabled. It continues to operate in all Sleep modes, including Power-Down.

The bus state logic includes Start and Stop condition detectors, collision detection, inactive bus time-out detection, and a bit counter. These are used to determine the bus state. The software can get the current bus state by reading the Bus State (BUSSTATE) bit field in the Master Status (TWIn.MSTATUS) register.

The bus state can be Unknown, Idle, Busy or Owner, and it is determined according to the state diagram shown below.

**Figure 29-4. Bus State Diagram**




1. **Unknown**: The bus state machine is active when the TWI master is enabled. After the TWI master has been enabled, the bus state is Unknown. The bus state will also be set to Unknown after a System Reset is performed or after the TWI master is disabled.
2. **Idle**: The bus state machine can be forced to enter the Idle state by writing `0x1` to the Bus State (BUSSTATE) bit field. The bus state logic cannot be forced into any other state. If no state is set by the application software, the bus state will become Idle when the first Stop condition is detected. If the Inactive Bus Time-Out (TIMEOUT) bit field from the Master Control A (TWIn.MCTRLA) register is configured to a nonzero value, the bus state will change to Idle on the occurrence of a time-out. When the bus is Idle, it is ready for a new transaction.
3. **Busy**: If a Start condition, generated externally, is detected when the bus is Idle, the bus state becomes Busy. The bus state changes back to Idle when a Stop condition is detected or when a time-out, if configured, is set.
4. **Owner**: If a Start condition is generated internally when the bus is Idle, the bus state becomes Owner. If the complete transaction is performed without interference, the master issues a Stop condition and the bus state

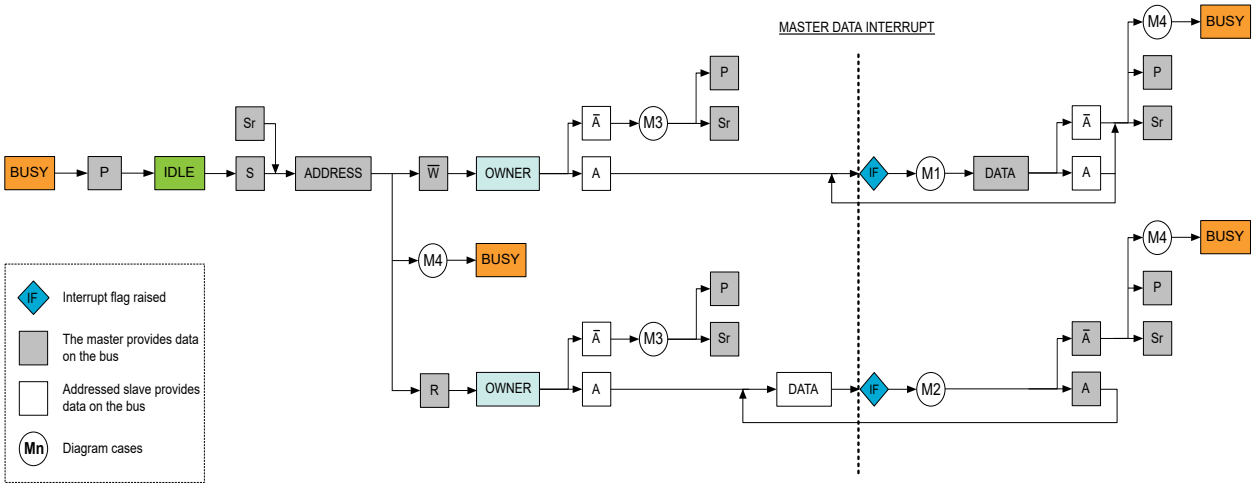
changes back to Idle. If a collision is detected, the arbitration is lost and the bus state becomes Busy until a Stop condition is detected.

#### 29.3.2.2.3 Transmitting Address Packets

The master starts performing a bus transaction when the Master Address (TWIn.MADDR) register is written with the slave address and the R/$\overline{W}$ direction bit. The value of the MADDR register is then copied in the Master Data (TWIn.MDATA) register. If the bus state is Busy, the TWI master will wait until the bus state becomes Idle before issuing the Start condition. The TWI will issue a Start condition, and the shift register performs a byte transmit operation on the bus.

Depending on the arbitration and the R/$\overline{W}$ direction bit, one of four cases (M1 to M4) arises after the transmission of the address packet.

**Figure 29-5. TWI Master Operation**




***Case M1: Address Packet Transmit Complete - Direction Bit Set to '0'***

If a slave device responds to the address packet with an ACK, the Write Interrupt Flag (WIF) is set to '1', the Received Acknowledge (RXACK) flag is set to '0', and the Clock Hold (CLKHOLD) flag is set to '1'. The WIF, RXACK and CLKHOLD flags are located in the Master Status (TWIn.MSTATUS) register.

The clock hold is active at this point, forcing the SCL low. This will stretch the low period of the clock to slow down the overall clock frequency, forcing delays required to process the data and preventing further activity on the bus.

The software can prepare to:

- Transmit data packets to the slave

***Case M2: Address Packet Transmit Complete - Direction Bit Set to '1'***

If a slave device responds to the address packet with an ACK, the RXACK flag is set to '0', and the slave can start sending data to the master without any delays because the slave owns the bus at this moment. The clock hold is active at this point, forcing the SCL low.

The software can prepare to:

- Read the received data packet from the slave

***Case M3: Address Packet Transmit Complete - Address not Acknowledged by Slave***

If no slave device responds to the address packet, the WIF and the RXACK flags will be set to '1'. The clock hold is active at this point, forcing the SCL low.

The missing ACK response can indicate that the $I^2C$ slave is busy with other tasks, or it is in a Sleep mode, and it is not able to respond.

The software can prepare to take one of the following actions:

- Retransmit the address packet
- Complete the transaction by issuing a Stop condition in the Command (MCMD) bit field from the Master Control B (TWIn.MCTRLB) register, which is the recommended action

***Case M4: Arbitration Lost or Bus Error***

If arbitration is lost, both the WIF and the Arbitration Lost (ARBLOST) flags in the Master Status (TWIn.MSTATUS) register are set to '1'. The SDA is disabled and the SCL is released. The bus state changes to Busy, and the master is no longer allowed to perform any operation on the bus until the bus state is changed back to Idle.

A bus error will behave similarly to the arbitration lost condition. In this case, the Bus Error (BUSERR) flag in the Master Status (TWIn.MSTATUS) register is set to '1', in addition to the WIF and ARBLOST flags.

The software can prepare to:

- Abort the operation and wait until the bus state changes to Idle by reading the Bus State (BUSSTATE) bit field in the Master Status (TWIn.MSTATUS) register

#### 29.3.2.2.4 Transmitting Data Packets

Assuming the above M1 case, the TWI master can start transmitting data by writing to the Master Data (TWIn.MDATA) register, which will also clear the Write Interrupt Flag (WIF). During the data transfer, the master is continuously monitoring the bus for collisions and errors. The WIF flag will be set to '1' after the data packet transfer has been completed.

If the transmission is successful and the master receives an ACK bit from the slave, the Received Acknowledge (RXACK) flag will be set to '0', meaning that the slave is ready to receive new data packets.

The software can prepare to take one of the following actions:

- Transmit a new data packet
- Transmit a new address packet
- Complete the transaction by issuing a Stop condition in the Command (MCMD) bit field from the Master Control B (TWIn.MCTRLB) register

If the transmission is successful and the master receives a NACK bit from the slave, the RXACK flag will be set to '1', meaning that the slave is not able to or does not need to receive more data.

The software can prepare to take one of the following actions:

- Transmit a new address packet
- Complete the transaction by issuing a Stop condition in the Command (MCMD) bit field from the Master Control B (TWIn.MCTRLB) register

The RXACK status is valid only if the WIF flag is set to '1' and the Arbitration Lost (ARBLOST) and Bus Error (BUSERR) flags are set to '0'.

The transmission can be unsuccessful if a collision is detected. Then, the master will lose arbitration, the Arbitration Lost (ARBLOST) flag will be set to '1', and the bus state changes to Busy. An arbitration lost during the sending of the data packet is treated the same way as the above M4 case.

The WIF, ARBLOST, BUSERR and RXACK flags are all located in the Master Status (TWIn.MSTATUS) register.

#### 29.3.2.2.5 Receiving Data Packets

Assuming the M2 case above, the clock is released for one byte, allowing the slave to put one byte of data on the bus. The master will receive one byte of data from the slave, and the Read Interrupt Flag (RIF) will be set to '1' together with the Clock Hold (CLKHOLD) flag. The action selected by the Acknowledge Action (ACKACT) bit in the Master Control B (TWIn.MCTRLB) register is automatically sent on the bus when a command is written to the Command (MCMD) bit field in the TWIn.MCTRLB register.

The software can prepare to take one of the following actions:

- Respond with an ACK by writing '0' to the ACKACT bit in the TWIn.MCTRLB register and prepare to receive a new data packet
- Respond with a NACK by writing '1' to the ACKACT bit and then transmit a new address packet
- Respond with a NACK by writing '1' to the ACKACT bit and then complete the transaction by issuing a Stop condition in the MCMD bit field from the TWIn.MCTRLB register

A NACK response might not be successfully executed, as arbitration can be lost during the transmission. If a collision is detected, the master loses arbitration, and the Arbitration Lost (ARBLOST) flag is set to '1' and the bus state changes to Busy. The Master Write Interrupt Flag (WIF) is set if the arbitration was lost when sending a NACK or a

bus error occurred during the procedure. An arbitration lost during the sending of the data packet is treated in the same way as the above M4 case.

The RIF, CLKHOLD, ARBLOST and WIF flags are all located in the Master Status (TWIn.MSTATUS) register.

**Note:** The RIF and WIF flags are mutually exclusive and cannot be set simultaneously.

### 29.3.2.3 TWI Slave Operation

The TWI slave is byte-oriented with optional interrupts after each byte. There are separate interrupt flags for the slave data and for address/Stop recognition. Interrupt flags can also be used for polled operation. There are dedicated status flags for indicating ACK/NACK received, clock hold, collision, bus error, and R/$\overline{W}$ direction bit.

When an interrupt flag is set to '`1`', the SCL is forced low. This will give the slave time to respond or handle any data, and will, in most cases, require software interaction. The number of interrupts generated is kept to a minimum by automatic handling of most conditions.

The Address Recognition Mode (PMEN) bit in the Slave Control A (TWIn.SCTRLA) register can be configured to allow the slave to respond to all received addresses.

#### 29.3.2.3.1 Receiving Address Packets

When the TWI is configured as a slave, it will wait for a Start condition to be detected. When this happens, the successive address packet will be received and checked by the address match logic. The slave will ACK a correct address and store the address in the Slave Data (TWIn.SDATA) register. If the received address is not a match, the slave will not acknowledge or store the address, but wait for a new Start condition.

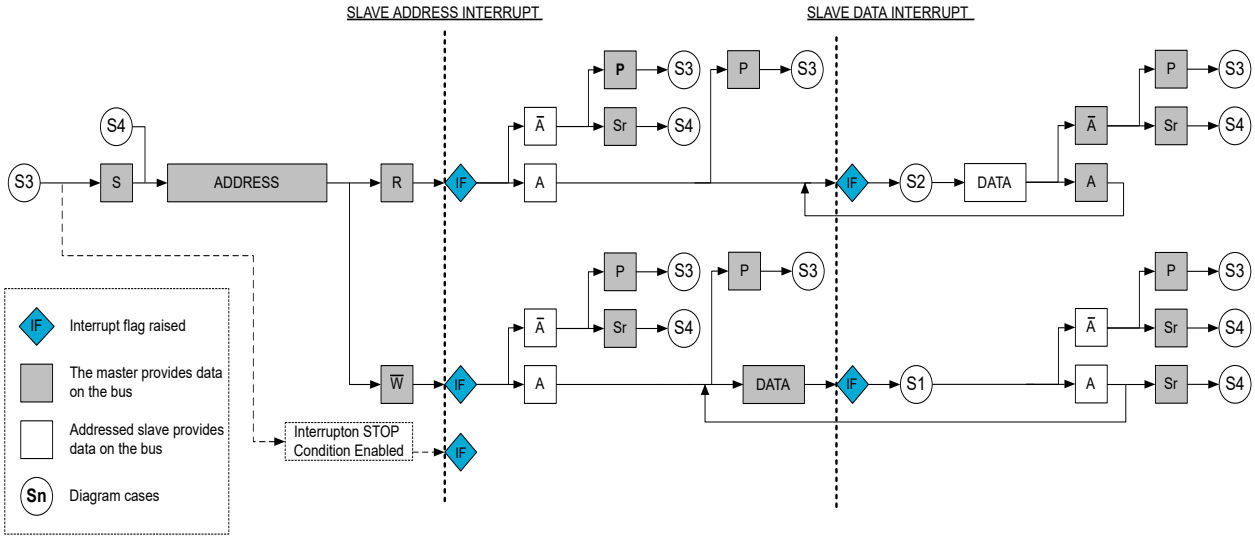
The Address or Stop Interrupt Flag (APIF) in the Slave Status (TWIn.SSTATUS) register is set to '`1`' when a Start condition is succeeded by one of the following:

- A valid address match with the address stored in the Address (ADDR[7:1]) bit field in the Slave Address (TWIn.SADDR) register
- The General Call Address `0x00`, and the Address (ADDR[0]) bit in the Slave Address (TWIn.SADDR) register is set to '`1`'
- A valid address match with the secondary address stored in the Address Mask (ADDRMASK) bit field and the Address Mask Enable (ADDREN) bit is set to '`1`' in the Slave Address Mask (TWIn.SADDRMASK) register
- Any address if the Address Recognition Mode (PMEN) bit in the Slave Control A (TWIn.SCTRLA) register is set to '`1`'

A Start condition immediately followed by a Stop condition is an illegal operation, and the Bus Error (BUSERR) flag in the Slave Status (TWIn.SSTATUS) register is set.

Depending on the Read/Write Direction (DIR) bit in the Slave Status (TWIn.SSTATUS) register and the bus condition, one of four distinct cases (S1 to S4) arises after the reception of the address packet.

**Figure 29-6. TWI Slave Operation**



***Case S1: Address Packet Accepted - Direction Bit Set to '0'***
If an ACK is sent by the slave after the address packet is received and the Read/Write Direction (DIR) bit in the Slave Status (TWIn.SSTATUS) register is set to '0', the master indicates a write operation.

The clock hold is active at this point, forcing the SCL low. This will stretch the low period of the clock to slow down the overall clock frequency, forcing delays required to process the data and preventing further activity on the bus.

The software can prepare to:
- Read the received data packet from the master

***Case S2: Address Packet Accepted - Direction Bit Set to '1'***
If an ACK is sent by the slave after the address packet is received and the DIR bit is set to '1', the master indicates a read operation, and the Data Interrupt Flag (DIF) in the Slave Status (TWIn.SSTATUS) register will be set to '1'.

The clock hold is active at this point, forcing the SCL low.

The software can prepare to:
- Transmit data packets to the master

***Case S3: Stop Condition Received***
When the Stop condition is received, the Address or Stop (AP) flag will be set to '0', indicating that a Stop condition, and not an address match, activated the Address or Stop Interrupt Flag (APIF).

The AP and APIF flags are located in the Slave Status (TWIn.SSTATUS) register.

The software can prepare to:
- Wait until a new address packet will be addressed to it

***Case S4: Collision***
If the slave is not able to send a high-level data bit or a NACK, the Collision (COLL) bit in the Slave Status (TWIn.SSTATUS) register is set to '1'. The slave will commence its operation as normal, except no low values will be shifted out on the SDA. The data and acknowledge output from the slave logic will be disabled. The clock hold is released. A Start or repeated Start condition will be accepted.

The COLL bit is intended for systems where the Address Resolution Protocol (ARP) is employed. A detected collision in non-ARP situations indicates that there has been a protocol violation and must be treated as a bus error.

#### 29.3.2.3.2 Receiving Data Packets
Assuming the above S1 case, the slave must be ready to receive data. When a data packet is received, the Data Interrupt Flag (DIF) in the Slave Status (TWIn.SSTATUS) register is set to '1'. The action selected by the Acknowledge Action (ACKACT) bit in the Slave Control B (TWIn.SCTRLB) register is automatically sent on the bus when a command is written to the Command (SCMD) bit field in the TWIn.SCTRLB register.

The software can prepare to take one of the following actions:
- Respond with an ACK by writing '0' to the ACKACT bit in the TWIn.SCTRLB register, indicating that the slave is ready to receive more data
- Respond with a NACK by writing '1' to the ACKACT bit, indicating that the slave cannot receive any more data and the master must issue a Stop or repeated Start condition

#### 29.3.2.3.3 Transmitting Data Packets
Assuming the above S2 case, the slave can start transmitting data by writing to the Slave Data (TWIn.SDATA) register. When a data packet transmission is completed, the Data Interrupt Flag (DIF) in the Slave Status (TWIn.SSTATUS) register is set to '1'.

The software can prepare to take one of the following actions:
- Check if the master responded with an ACK by reading the Received Acknowledge (RXACK) bit from the Slave Status (TWIn.SSTATUS) register and start transmitting new data packets
- Check if the master responded with a NACK by reading the RXACK and stop transmitting data packets. The master must send a Stop or repeated Start condition after the NACK.

### 29.3.3 Additional Features

#### 29.3.3.1 SMBus

If the TWI is used in an SMBus environment, the Inactive Bus Time-Out (TIMEOUT) bit field from the Master Control A (TWIn.MCTRLA) register must be configured. It is recommended to write to the Master Baud Rate (TWIn.MBAUD) register before setting the time-out because it is dependent on the baud rate setting.

A frequency of 100 kHz can be used for the SMBus environment. For the Standard mode (Sm) and Fast mode (Fm), the operating frequency has slew rate limited output, while for the Fast mode Plus (Fm+), it has x10 output drive strength.

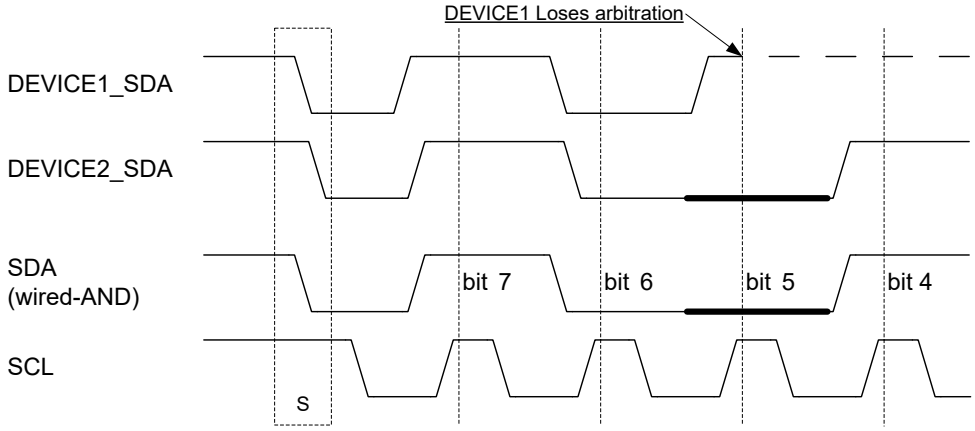
The TWI also allows for an SMBus compatible SDA hold time configured in the SDA Hold Time (SDAHOLD) bit field from the Control A (TWIn.CTRLA) register.

#### 29.3.3.2 Multi Master

A master can start a bus transaction only if it has detected that the bus is in the Idle state. As the TWI bus is a multi-master bus, more devices may try to initiate a transaction at the same time. This results in multiple masters owning the bus simultaneously. The TWI solves this problem by using an arbitration scheme where the master loses control of the bus if it is not able to transmit a high-level data bit on the SDA and the Bus State (BUSSTATE) bit field from the Master Status (TWIn.MSTATUS) register will be changed to Busy. The masters that lose the arbitration must wait until the bus becomes Idle before attempting to reacquire the bus ownership.

Both devices can issue a Start condition, but DEVICE1 loses arbitration when attempting to transmit a high level (bit 5) while DEVICE2 is transmitting a low level.

**Figure 29-7. TWI Arbitration**




#### 29.3.3.3 Smart Mode

The TWI interface has a Smart mode that simplifies the application code and minimizes the user interaction needed to adhere to the $I^2C$ protocol.

For the TWI Master, the Smart mode will automatically send the ACK action as soon as the Master Data (TWIn.MDATA) register is read. This feature is only active when the Acknowledge Action (ACKACT) bit in the Master Control B (TWIn.MCTRLB) register is set to ACK. If the ACKACT bit is set to NACK, the TWI Master will not generate a NACK after the MDATA register is read. This feature is enabled when the Smart Mode Enable (SMEN) bit in the Master Control A (TWIn.MCTRLA) register is set to '1'.

For the TWI Slave, the Smart mode will automatically send the ACK action as soon as the Slave Data (TWIn.SDATA) register is read. The Smart mode will automatically set the Data Interrupt Flag (DIF) to '0' in the Slave Status (TWIn.SSTATUS) register if the TWIn.SDATA register is read or written. This feature is enabled when the Smart Mode Enable (SMEN) bit in the Slave Control A (TWIn.SCTRLA) register is set to '1'.

#### 29.3.3.4 Dual Mode

The TWI supports Dual mode operation where the master and the slave will operate simultaneously and independently. In this case, the Control A (TWIn.CTRLA) register will configure the master device, and the Dual Mode Control (TWIn.DUALCTRL) register will configure the slave device. See the 29.3.2.1 Initialization section for more details about the master configuration.

If used, the following bits must be configured before enabling the TWI Dual mode:

- The SDA Hold Time (SDAHOLD) bit field
- The FM Plus Enable (FMPEN) bit from the DUALCTRL register

The Dual mode can be enabled by writing a '1' to the Dual Control Enable (ENABLE) bit in the DUALCTRL register.

#### 29.3.3.5 Quick Command Mode

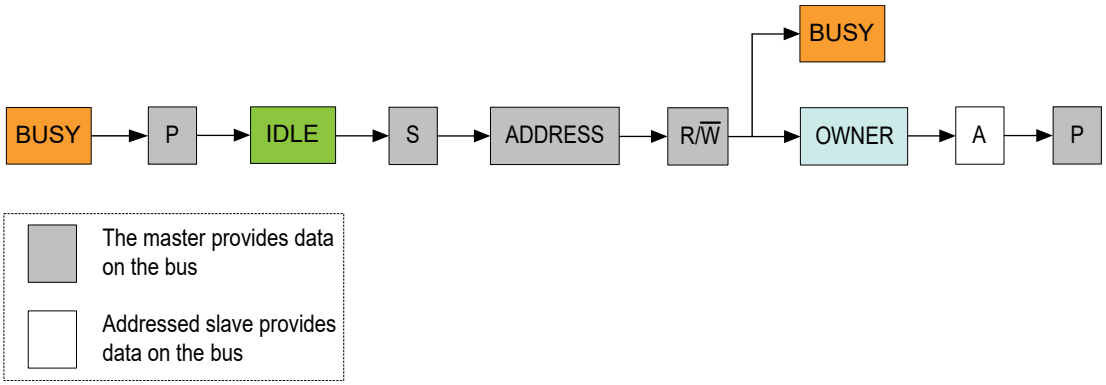
With Quick Command mode, the R/$\overline{W}$ bit from the address packet denotes the command. This mode is enabled by writing '1' to the Quick Command Enable (QCEN) bit in the Master Control A (TWIn.MCTRLA) register. There are no data sent or received.

The Quick Command mode is SMBus specific, where the R/$\overline{W}$ bit can be used to turn a device function on/off or to enable/disable a low-power Standby mode. This mode can be enabled to auto-trigger operations and reduce the software complexity.

After the master receives an ACK from the slave, either the Read Interrupt Flag (RIF) or Write Interrupt Flag (WIF) will be set, depending on the value of the R/$\overline{W}$ bit. When either the RIF or WIF flag is set after issuing a Quick Command, the TWI will accept a Stop command by writing the Command (MCMD) bit field in the Master Control B (TWIn.MCTRLB) register.

The RIF and WIF flags, together with the value of the last Received Acknowledge (RXACK) flag are all located in the Master Status (TWIn.MSTATUS) register.

**Figure 29-8. Quick Command Frame Format**



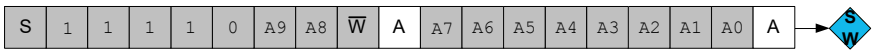

#### 29.3.3.6 10-bit Address

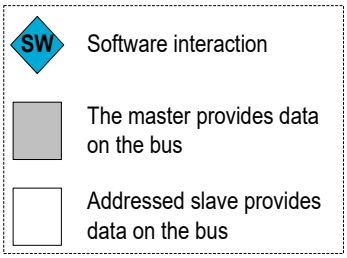
Regardless of whether the transaction is a read or write, the master must start by sending the 10-bit address with the R/$\overline{W}$ direction bit set to '0'.

The slave address match logic supports recognition of 7-bit addresses and general call address. The Slave Address (TWIn.SADDR) register is used by the slave address match logic to determine if a master device has addressed the TWI slave.

The TWI slave address match logic only supports recognition of the first byte of a 10-bit address and the second byte must be handled in software. The first byte of the 10-bit address will be recognized if the upper five bits of the Slave Address (TWIn.SADDR) register are 0b11110. Thus, the first byte will consist of five indication bits, the two Most Significant bits (MSb) of the 10-bits address, and the R/$\overline{W}$ direction bit. The Least Significant Byte (LSB) of the address that follows from the master will come in the form of a data packet.

**Figure 29-9. 10-bit Address Transmission**




### 29.3.4 Interrupts

**Table 29-1. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| Slave | TWI Slave interrupt | • DIF: Data Interrupt Flag in TWIn.SSTATUS is set to '1'<br>• APIF: Address or Stop Interrupt Flag in TWIn.SSTATUS is set to '1' |
| Master | TWI Master interrupt | • RIF: Read Interrupt Flag in TWIn.MSTATUS is set to '1'<br>• WIF: Write Interrupt Flag in TWIn.MSTATUS is set to '1' |

When an interrupt condition occurs, the corresponding interrupt flag is set in the Master Status (TWIn.MSTATUS) register or the Slave Status (TWIn.SSTATUS) register.
When several interrupt request conditions are supported by an interrupt vector, the interrupt requests are ORed together into one combined interrupt request to the interrupt controller. The user must read the Interrupt flags from the TWIn.MSTATUS register or the TWIn.SSTATUS register, to determine which of the interrupt conditions are present.

### 29.3.5 Sleep Mode Operation

The bus state logic and the address recognition hardware continue to operate in all sleep modes. If a slave device is in Sleep mode and a Start condition followed by the address of the slave is detected, clock stretching is active during the wake-up period until the main clock is available. The master will stop operation in all sleep modes. When the Dual mode is active, the device will wake up only when the Start condition is sent on the bus of the slave.

### 29.3.6 Debug Operation

During run-time debugging, the TWI will continue normal operation. Halting the CPU in Debugging mode will halt the normal operation of the TWI. The TWI can be forced to operate with halted CPU by writing a '1' to the Debug Run (DBGRUN) bit in the Debug Control (TWIn.DBGCTRL) register. When the CPU is halted in Debug mode and the DBGRUN bit is '1', reading or writing the Master Data (TWIn.MDATA) register or the Slave Data (TWIn.SDATA) register will neither trigger a bus operation, nor cause transmit and clear flags. If the TWI is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

## 29.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | INPUTLVL | | SDASETUP | SDAHOLD[1:0] | | FMPEN | |
| 0x01 | DUALCTRL | 7:0 | | INPUTLVL | | | SDAHOLD[1:0] | | FMPEN | ENABLE |
| 0x02 | DBGCTRL | 7:0 | | | | | | | | DBGRUN |
| 0x03 | MCTRLA | 7:0 | RIEN | WIEN | | QCEN | TIMEOUT[1:0] | | SMEN | ENABLE |
| 0x04 | MCTRLB | 7:0 | | | | | FLUSH | ACKACT | MCMD[1:0] | |
| 0x05 | MSTATUS | 7:0 | RIF | WIF | CLKHOLD | RXACK | ARBLOST | BUSERR | BUSSTATE[1:0] | |
| 0x06 | MBAUD | 7:0 | BAUD[7:0] | | | | | | | |
| 0x07 | MADDR | 7:0 | ADDR[7:0] | | | | | | | |
| 0x08 | MDATA | 7:0 | DATA[7:0] | | | | | | | |
| 0x09 | SCTRLA | 7:0 | DIEN | APIEN | PIEN | | | PMEN | SMEN | ENABLE |
| 0x0A | SCTRLB | 7:0 | | | | | | ACKACT | SCMD[1:0] | |
| 0x0B | SSTATUS | 7:0 | DIF | APIF | CLKHOLD | RXACK | COLL | BUSERR | DIR | AP |
| 0x0C | SADDR | 7:0 | ADDR[7:0] | | | | | | | |
| 0x0D | SDATA | 7:0 | DATA[7:0] | | | | | | | |
| 0x0E | SADDRMASK | 7:0 | ADDRMASK[6:0] | | | | | | | ADDREN |

## 29.5 Register Description

### 29.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | INPUTLVL | | SDASETUP | SDAHOLD[1:0] | | FMPEN | |
| Access | | R/W | | R/W | R/W | R/W | R/W | |
| Reset | | 0 | | 0 | 0 | 0 | 0 | |

**Bit 6 – INPUTLVL** Input Voltage Transition Level
This bit is used to select between $I^2C$ and SMBUS.

| Value | Name | Description |
|---|---|---|
| 0 | I2C | $I^2C$ input voltage transition level |
| 1 | SMBUS | SMBus 3.0 input voltage transition level |

**Bit 4 – SDASETUP** SDA Setup Time
By default, there are four clock cycles of setup time on the SDA out signal while reading from the slave part of the TWI module.

| Value | Name | Description |
|---|---|---|
| 0 | 4CYC | SDA setup time is four clock cycles |
| 1 | 8CYC | SDA setup time is eight clock cycles |

**Bits 3:2 – SDAHOLD[1:0]** SDA Hold Time
This bit field selects the SDA hold time for the TWI. See the *Electrical Characteristics* section for details.

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | Hold time OFF |
| 0x1 | 50NS | Short hold time |
| 0x2 | 300NS | Meets the SMBus 2.0 specifications under typical conditions |
| 0x3 | 500NS | Meets the SMBus 2.0 across all corners |

**Bit 1 – FMPEN** FM Plus Enable
Writing a '1' to this bit selects the 1 MHz bus speed for the TWI in default configuration or for the TWI Master in Dual mode configuration.

| Value | Name | Description |
|---|---|---|
| 0 | OFF | Operating in Standard mode or Fast mode |
| 1 | ON | Operating in Fast mode Plus |

### 29.5.2 Dual Mode Control Configuration

**Name:** DUALCTRL
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | INPUTLVL | | | SDAHOLD[1:0] | | FMPEN | ENABLE |
| Access | | R/W | | | R/W | R/W | R/W | R/W |
| Reset | | 0 | | | 0 | 0 | 0 | 0 |

**Bit 6 – INPUTLVL** Input Voltage Transition Level
This bit is used to select between $I^2C$ and SMBUS.

| Value | Name | Description |
|---|---|---|
| 0 | I2C | $I^2C$ input voltage transition level |
| 1 | SMBUS | SMBus 3.0 input voltage transition level |

**Bits 3:2 – SDAHOLD[1:0]** SDA Hold Time
This bit field selects the SDA hold time for the TWI Slave. This bit field is ignored if the Dual mode is not enabled.

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | Hold time OFF |
| 0x1 | 50NS | Short hold time |
| 0x2 | 300NS | Meets the SMBus 2.0 specifications under typical conditions |
| 0x3 | 500NS | Meets the SMBus 2.0 across all corners |

**Bit 1 – FMPEN** FM Plus Enable
Writing a '1' to this bit selects the 1 MHz bus speed for the TWI Slave.

| Value | Name | Description |
|---|---|---|
| 0 | OFF | Operating in Standard mode or Fast mode |
| 1 | ON | Operating in Fast mode Plus |

**Bit 0 – ENABLE** Dual Control Enable
Writing a '1' to this bit will enable the Dual mode configuration.

### 29.5.3 Debug Control

**Name:** DBGCTRL
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Debug Run
See the 29.3.6 Debug Operation section for more details.

| Value | Description |
|---|---|
| 0 | The TWI is halted in Break Debug mode and ignores events |
| 1 | The TWI will continue to run in Break Debug mode when the CPU is halted |

### 29.5.4 Master Control A

**Name:** MCTRLA
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RIEN | WIEN | | QCEN | TIMEOUT[1:0] | | SMEN | ENABLE |
| 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 – RIEN** Read Interrupt Enable
A TWI master read interrupt will be generated only if this bit and the Global Interrupt Enable (I) bit in the Status (CPU.SREG) register are set to '1'.
Writing a '1' to this bit enables the interrupt on the Read Interrupt Flag (RIF) in the Master Status (TWIn.MSTATUS) register. When the master read interrupt occurs, the RIF flag is set to '1'.

**Bit 6 – WIEN** Write Interrupt Enable
A TWI master write interrupt will be generated only if this bit and the Global Interrupt Enable (I) bit in the Status (CPU.SREG) register are set to '1'.
Writing a '1' to this bit enables the interrupt on the Write Interrupt Flag (WIF) in the Master Status (TWIn.MSTATUS) register. When the master write interrupt occurs, the WIF flag is set to '1'.

**Bit 4 – QCEN** Quick Command Enable
Writing a '1' to this bit enables the Quick Command mode. If the Quick Command mode is enabled and a slave acknowledges the address, the corresponding Read Interrupt Flag (RIF) or Write Interrupt Flag (WIF) will be set depending on the value of R/$\overline{W}$ bit.
The software must issue a Stop command by writing to the Command (MCMD) bit field in the Master Control B (TWIn.MCTRLB) register.

**Bits 3:2 – TIMEOUT[1:0]** Inactive Bus Time-Out
Setting this bit field to a nonzero value will enable the inactive bus time-out supervisor. If the bus is inactive for longer than the TIMEOUT setting, the bus state logic will enter the Idle state.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLED | Bus time-out disabled - $I^2C$ |
| 0x1 | 50US | 50 µs - SMBus (assume the baud rate is set to 100 kHz) |
| 0x2 | 100US | 100 µs (assume the baud rate is set to 100 kHz) |
| 0x3 | 200US | 200 µs (assume the baud rate is set to 100 kHz) |

**Bit 1 – SMEN** Smart Mode Enable
Writing a '1' to this bit enables the Master Smart mode. When the Smart mode is enabled, the existing value in the Acknowledge Action (ACKACT) bit from the Master Control B (TWIn.MCTRLB) register is sent immediately after reading the Master Data (TWIn.MDATA) register.

**Bit 0 – ENABLE** Enable TWI Master
Writing a '1' to this bit enables the TWI as master.

### 29.5.5 Master Control B

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | FLUSH | ACKACT | MCMD[1:0] | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bit 3 – FLUSH** Flush
This bit clears the internal state of the master and the bus states changes to Idle. The TWI will transmit invalid data if the Master Data (TWIn.MDATA) register is written before the Master Address (TWIn.MADDR) register.
Writing a '1' to this bit generates a strobe for one clock cycle, disabling the master, and then re-enabling the master.
Writing a '0' to this bit has no effect.

**Bit 2 – ACKACT** Acknowledge Action
The ACKACT[1] bit represents the behavior in the Master mode under certain conditions defined by the bus state and the software interaction. If the Smart Mode Enable (SMEN) bit in the Master Control A (TWIn.MCTRLA) register is set to '1', the acknowledge action is performed when the Master Data (TWIn.MDATA) register is read, else a command must be written to the Command (MCDM) bit field in the Master Control B (TWIn.MCTRLB) register.
The acknowledge action is not performed when the Master Data (TWIn.MDATA) register is written, since the master is sending data.

| Value | Name | Description |
|---|---|---|
| 0 | ACK | Send ACK |
| 1 | NACK | Send NACK |

**Bits 1:0 – MCMD[1:0]** Command
The MCMD[1] bit field is a strobe. This bit field is always read as '0'.
Writing to this bit field triggers a master operation as defined by the table below.

**Table 29-2. Command Settings**

| MCMD[1:0] | Group Configuration | DIR | Description |
|---|---|---|---|
| 0x0 | NOACT | X | Reserved |
| 0x1 | REPSTART | X | Execute Acknowledge Action followed by repeated Start condition |
| 0x2 | RECVTRANS | $\overline{W}$ | Execute Acknowledge Action (no action) followed by a byte write operation[2] |
| | | R | Execute Acknowledge Action followed by a byte read operation |
| 0x3 | STOP | X | Execute Acknowledge Action followed by issuing a Stop condition |

**Notes:**
1. The ACKACT bit and the MCMD bit field can be written at the same time.
2. For a master write operation, the TWI will wait for new data to be written to the Master Data (TWIn.MDATA) register.

### 29.5.6 Master Status

**Name:** MSTATUS
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RIF | WIF | CLKHOLD | RXACK | ARBLOST | BUSERR | BUSSTATE[1:0] | |
| Access | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – RIF** Read Interrupt Flag
This flag is set to '1' when the master byte read operation is successfully completed.
The RIF flag can be used for a master read interrupt. More information can be found in the Read Interrupt Enable (RIEN) bit from the Master Control A (TWIn.MCTRLA) register.
This flag is automatically cleared when accessing several other TWI registers. The RIF flag can be cleared by choosing one of the following methods:

1. Writing a '1' to it.
2. Writing to the Master Address (TWIn.MADDR) register.
3. Writing/Reading the Master Data (TWIn.MDATA) register.
4. Writing to the Command (MCMD) bit field from the Master Control B (TWIn.MCTRLB) register.

**Bit 6 – WIF** Write Interrupt Flag
This flag is set to '1' when a master transmit address or byte write operation is completed, regardless of the occurrence of a bus error or arbitration lost condition.
The WIF flag can be used for a master write interrupt. More information can be found from the Write Interrupt Enable (WIEN) bit in the Master Control A (TWIn.MCTRLA) register.
This flag can be cleared by choosing one of the methods described for the RIF flag.

**Bit 5 – CLKHOLD** Clock Hold
When this bit is read as '1', it indicates that the master is currently holding the SCL low, stretching the TWI clock period.
This bit can be cleared by choosing one of the methods described for the RIF flag.

**Bit 4 – RXACK** Received Acknowledge
When this flag is read as '0', it indicates that the most recent Acknowledge bit from the slave was ACK and the slave is ready for more data.
When this flag is read as '1', it indicates that the most recent Acknowledge bit from the slave was NACK and the slave is not able to or does not need to receive more data.

**Bit 3 – ARBLOST** Arbitration Lost
When this bit is read as '1', it indicates that the master has lost arbitration. This can happen in one of the following cases:

1. While transmitting a high data bit.
2. While transmitting a NACK bit.
3. While issuing a Start condition (S).
4. While issuing a repeated Start (Sr).

This flag can be cleared by choosing one of the methods described for the RIF flag.

**Bit 2 – BUSERR** Bus Error
The BUSERR flag indicates that an illegal bus operation has occurred. An illegal bus operation is detected if a protocol violating the Start (S), repeated Start (Sr), or Stop (P) conditions is detected on the TWI bus lines. A Start condition directly followed by a Stop condition is one example of a protocol violation.

The BUSERR flag can be cleared by choosing one of the following methods:

1. Writing a '1' to it.
2. Writing to the Master Address (TWIn.MADDR) register.

The TWI bus error detector is part of the TWI Master circuitry. For the bus errors to be detected, the TWI Master must be enabled (ENABLE bit in TWIn.MCTRLA is '1'), and the main clock frequency must be at least four times the SCL frequency.

**Bits 1:0 – BUSSTATE[1:0]** Bus State
This bit field indicates the current TWI bus state.

| Value | Name | Description |
|---|---|---|
| 0x0 | UNKNOWN | Unknown bus state |
| 0x1 | IDLE | Idle bus state |
| 0x2 | OWNER | This TWI controls the bus |
| 0x3 | BUSY | Busy bus state |

### 29.5.7 Master Baud Rate

**Name:** MBAUD
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

| 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 Rate
This bit field is used to derive the SCL high and low time. It must be written while the master is disabled. The master can be disabled by writing '0' to the Enable TWI Master (ENABLE) bit from the Master Control A (TWIn.MCTRLA) register.
Refer to the 29.3.2.2.1 Clock Generation section for more information on how to calculate the frequency of the SCL.

### 29.5.8 Master Address

**Name:** MADDR
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

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

**Bits 7:0 – ADDR[7:0]** Address
This register contains the address of the external slave device. When this bit field is written, the TWI will issue a Start condition, and the shift register performs a byte transmit operation on the bus depending on the bus state.
This register can be read at any time without interfering with the ongoing bus activity since a read access does not trigger the master logic to perform any bus protocol related operations.
The master control logic uses the bit 0 of this register as the R/$\overline{W}$ direction bit.

### 29.5.9 Master Data

**Name:** MDATA
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

| 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 7:0 – DATA[7:0]** Data
This bit field provides direct access to the master's physical shift register, which is used to shift out data on the bus (transmit) and to shift in data received from the bus (receive). The direct access implies that the MDATA register cannot be accessed during byte transmissions.
Reading valid data or writing data to be transmitted can only be successful when the CLKHOLD bit is read as '1' or when an interrupt occurs.
A write access to the MDATA register will command the master to perform a byte transmit operation on the bus, directly followed by receiving the Acknowledge bit from the slave. This is independent of the Acknowledge Action (ACKACT) bit from the Master Control B (TWIn.MCTRLB) register. The write operation is performed regardless of winning or losing arbitration before the Write Interrupt Flag (WIF) is set to '1'.
If the Smart Mode Enable (SMEN) bit in the Master Control A (TWIn.MCTRLA) register is set to '1', a read access to the MDATA register will command the master to perform an acknowledge action. This is dependent on the setting of the Acknowledge Action (ACKACT) bit from the Master Control B (TWIn.MCTRLB) register.
**Notes:**

1. The WIF and RIF interrupt flags are cleared automatically if the MDATA register is read while ACKACT is set to '1'.
2. The ARBLOST and BUSEER flags are left unchanged.
3. The WIF, RIF, ARBLOST, and BUSERR flags together with the Clock Hold (CLKHOLD) bit are all located in the Master Status (TWIn.MSTATUS) register.

### 29.5.10 Slave Control A

**Name:** SCTRLA
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DIEN | APIEN | PIEN | | | PMEN | SMEN | ENABLE |
| Access | R/W | R/W | R/W | | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | | 0 | 0 | 0 |

**Bit 7 – DIEN** Data Interrupt Enable
Writing this bit to '1' enables an interrupt on the Data Interrupt Flag (DIF) from the Slave Status (TWIn.SSTATUS) register.
A TWI slave data interrupt will be generated only if this bit, the DIF flag, and the Global Interrupt Enable (I) bit in Status (CPU.SREG) register are all '1'.

**Bit 6 – APIEN** Address or Stop Interrupt Enable
Writing this bit to '1' enables an interrupt on the Address or Stop Interrupt Flag (APIF) from the Slave Status (TWIn.SSTATUS) register.
A TWI slave address or stop interrupt will be generated only if this bit, the APIF flag, and the Global Interrupt Enable (I) bit in the Status (CPU.SREG) register are all '1'.
**Notes:**

1. The slave stop interrupt shares the interrupt flag and vector with the slave address interrupt.
2. The Stop Interrupt Enable (PIEN) bit in the Slave Control A (TWIn.SCTRLA) register must be written to '1' for the APIF to be set on a Stop condition.
3. When the interrupt occurs, the Address or Stop (AP) bit in the Slave Status (TWIn.SSTATUS) register will determine whether an address match or a Stop condition caused the interrupt.

**Bit 5 – PIEN** Stop Interrupt Enable
Writing this bit to '1' allows the Address or Stop Interrupt Flag (APIF) in the Slave Status (TWIn.SSTATUS) register to be set when a Stop condition occurs. To use this feature, the main clock frequency must be at least four times the SCL frequency.

**Bit 2 – PMEN** Address Recognition Mode
If this bit is written to '1', the slave address match logic responds to all received addresses.
If this bit is written to '0', the address match logic uses the Slave Address (TWIn.SADDR) register to determine which address to recognize as the slave's address.

**Bit 1 – SMEN** Smart Mode Enable
Writing this bit to '1' enables the slave Smart mode. When the Smart mode is enabled, issuing a command by writing to the Command (SCMD) bit field in the Slave Control B (TWIn.SCTRLB) register or accessing the Slave Data (TWIn.SDATA) register resets the interrupt, and the operation continues. If the Smart mode is disabled, the slave always waits for a new slave command before continuing.

**Bit 0 – ENABLE** Enable TWI Slave
Writing this bit to '1' enables the TWI slave.

### 29.5.11 Slave Control B

**Name:** SCTRLB
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | ACKACT | SCMD[1:0] | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bit 2 – ACKACT** Acknowledge Action
The ACKACT[(1)] bit represents the behavior of the slave device under certain conditions defined by the bus protocol state and the software interaction. If the Smart Mode Enable (SMEN) bit in the Slave Control A (TWIn.SCTRLA) register is set to '1', the acknowledge action is performed when the Slave Data (TWIn.SDATA) register is read, else a command must be written to the Command (SCMD) bit field in the Slave Control B (TWIn.SCTRLB) register.
The acknowledge action is not performed when the Slave Data (TWIn.SDATA) register is written, since the slave is sending data.

| Value | Name | Description |
|---|---|---|
| 0 | ACK | Send ACK |
| 1 | NACK | Send NACK |

**Bits 1:0 – SCMD[1:0]** Command
The SCMD[(1)] bit field is a strobe. This bit field is always read as '0'.
Writing to this bit field triggers a slave operation as defined by the table below.

**Table 29-3. Command Settings**

| SCMD[1:0] | Group Configuration | DIR | Description |
|---|---|---|---|
| 0x0 | NOACT | X | No action |
| 0x1 | — | X | Reserved |
| 0x2 | COMPTRANS | Used to complete a transaction | |
| | | $\overline{W}$ | Execute Acknowledge Action succeeded by waiting for any Start (S/Sr) condition |
| | | R | Wait for any Start (S/Sr) condition |
| 0x3 | RESPONSE | Used in response to an address interrupt (APIF) | |
| | | $\overline{W}$ | Execute Acknowledge Action succeeded by reception of next byte |
| | | R | Execute Acknowledge Action succeeded by slave data interrupt |
| | | Used in response to a data interrupt (DIF) | |
| | | $\overline{W}$ | Execute Acknowledge Action succeeded by reception of next byte |
| | | R | Execute a byte read operation followed by Acknowledge Action |

**Note:** 1. The ACKACT bit and the SCMD bit field can be written at the same time. The ACKACT will be updated before the command is triggered.

### 29.5.12 Slave Status

**Name:** SSTATUS
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DIF | APIF | CLKHOLD | RXACK | COLL | BUSERR | DIR | AP |
| Access | R/W | R/W | R | R | R/W | R/W | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DIF** Data Interrupt Flag
This flag is set to '1' when the slave byte transmit or receive operation is successfully completed without any bus errors. This flag can be set to '1' with an unsuccessful transaction in case of a collision detection. More information can be found in the Collision (COLL) bit description.
The DIF flag can generate a slave data interrupt. More information can be found in Data Interrupt Enable (DIEN) bit from the Slave Control A (TWIn.SCTRLA) register.
This flag is automatically cleared when accessing several other TWI registers. The DIF flag can be cleared by choosing one of the following methods:

1. Writing/Reading the Slave Data (TWIn.SDATA) register.
2. Writing to the Command (SCMD) bit field from the Slave Control B (TWIn.SCTRLB) register.

**Bit 6 – APIF** Address or Stop Interrupt Flag
This flag is set to '1' when the slave address has been received or by a Stop condition.
The APIF flag can generate a slave address or stop interrupt. More information can be found in the Address or Stop Interrupt Enable (APIEN) bit from the Slave Control A (TWIn.SCTRLA) register.
This flag can be cleared by choosing one of the methods described for the DIF flag.

**Bit 5 – CLKHOLD** Clock Hold
When this bit is read as '1', it indicates that the slave is currently holding the SCL low, stretching the TWI clock period.
This bit is set to '1' when an address or data interrupt occurs. Resetting the corresponding interrupt will indirectly set this bit to '0'.

**Bit 4 – RXACK** Received Acknowledge
When this flag is read as '0', it indicates that the most recent Acknowledge bit from the master was ACK.
When this flag is read as '1', it indicates that the most recent Acknowledge bit from the master was NACK.

**Bit 3 – COLL** Collision
When this bit is read as '1', it indicates that the slave has not been able to do one of the following:

1. Transmit high bits on the SDA. The Data Interrupt Flag (DIF) will be set to '1' at the end as a result of the internal completion of an unsuccessful transaction.
2. Transmit the NACK bit. The collision occurs because the slave address match already took place, and the APIF flag is set to '1' as a result.

Writing a '1' to this bit will clear the COLL flag. The flag is automatically cleared if any Start condition (S/Sr) is detected.
**Note:** The APIF and DIF flags can only generate interrupts whose handlers can be used to check for the collision.

**Bit 2 – BUSERR** Bus Error
The BUSERR flag indicates that an illegal bus operation has occurred. Illegal bus operation is detected if a protocol violating the Start (S), repeated Start (Sr), or Stop (P) conditions is detected on the TWI bus lines. A Start condition directly followed by a Stop condition is one example of a protocol violation. Writing a '1' to this bit will clear the BUSERR flag.

The TWI bus error detector is part of the TWI Master circuitry. For the bus errors to be detected by the slave, the TWI Dual mode or the TWI Master must be enabled, and the main clock frequency must be at least four times the SCL frequency. The TWI Dual mode can be enabled by writing '1' to the ENABLE bit in the TWIn.DUALCTRL register. The TWI Master can be enabled by writing '1' to the ENABLE bit in the TWIn.MCTRLA register.

**Bit 1 – DIR** Read/Write Direction
This bit indicates the current TWI bus direction. The DIR bit reflects the direction bit value from the last address packet received from a master TWI device.
When this bit is read as '1', it indicates that a master read operation is in progress.
When this bit is read as '0', it indicates that a master write operation is in progress.

**Bit 0 – AP** Address or Stop
When the TWI slave Address or Stop Interrupt Flag (APIF) is set '1', this bit determines whether the interrupt is due to an address detection or a Stop condition.

| Value | Name | Description |
|---|---|---|
| 0 | STOP | A Stop condition generated the interrupt on the APIF flag |
| 1 | ADR | Address detection generated the interrupt on the APIF flag |

### 29.5.13 Slave Address

**Name:** SADDR
**Offset:** 0x0C
**Reset:** 0x00
**Property:** -

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

**Bits 7:0 – ADDR[7:0]** Address
The Slave Address (TWIn.SADDR) register is used by the slave address match logic to determine if a master device has addressed the TWI slave. The Address or Stop Interrupt Flag (APIF) and the Address or Stop (AP) bit in the Slave Status (TWIn.SSTATUS) register are set to '`1`' if an address packet is received.
The upper seven bits (ADDR[7:1]) of the SADDR register represent the main slave address.
The Least Significant bit (ADDR[0]) of the SADDR register is used for the recognition of the General Call Address (`0x00`) of the $I^2C$ protocol. This feature is enabled when this bit is set to '`1`'.

### 29.5.14 Slave Data

**Name:** SDATA
**Offset:** 0x0D
**Reset:** 0x00
**Property:** -

| 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 7:0 – DATA[7:0]** Data
This bit field provides access to the slave data register.
Reading valid data or writing data to be transmitted can only be successfully achieved when the SCL is held low by the slave (i.e., when the slave CLKHOLD bit is set to '1'). It is not necessary to check the Clock Hold (CLKHOLD) bit from the Slave Status (TWIn.SSTATUS) register in software before accessing the SDATA register if the software keeps track of the present protocol state by using interrupts or observing the interrupt flags.
If the Smart Mode Enable (SMEN) bit in the Slave Control A (TWIn.SCTRLA) register is set to '1', a read access to the SDATA register, when the clock hold is active, auto-triggers bus operations and will command the slave to perform an acknowledge action. This is dependent on the setting of the Acknowledge Action (ACKACT) bit from the Slave Control B (TWIn.SCTRLB) register.

### 29.5.15 Slave Address Mask

**Name:** SADDRMASK
**Offset:** 0x0E
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADDRMASK[6:0] | | | | | | | ADDREN |
| Access | R/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 7:1 – ADDRMASK[6:0]** Address Mask
The ADDRMASK bit field acts as a second address match or an address mask register depending on the ADDREN bit.
If the ADDREN bit is written to '0', the ADDRMASK bit field can be loaded with a 7-bit Slave Address mask. Each of the bits in the Slave Address Mask (TWIn.SADDRMASK) register can mask (disable) the corresponding address bits in the TWI Slave Address (TWIn.SADDR) register. When a bit from the mask is written to '1', the address match logic ignores the comparison between the incoming address bit and the corresponding bit in the Slave Address (TWIn.SADDR) register. In other words, masked bits will always match, making it possible to recognize ranges of addresses.
If the ADDREN bit is written to '1', the Slave Address Mask (TWIn.SADDRMASK) register can be loaded with a second slave address in addition to the Slave Address (TWIn.SADDR) register. In this mode, the slave will have two unique addresses, one in the Slave Address (TWIn.SADDR) register and the other one in the Slave Address Mask (TWIn.SADDRMASK) register.

**Bit 0 – ADDREN** Address Mask Enable
If this bit is written to '0', the TWIn.SADDRMASK register acts as a mask to the TWIn.SADDR register.
If this bit is written to '1', the slave address match logic responds to the two unique addresses in slave TWIn.SADDR and TWIn.SADDRMASK.

# 30. CRCSCAN - Cyclic Redundancy Check Memory Scan

## 30.1 Features

- CRC-16-CCITT or CRC-32 (IEEE 802.3)
- Check of the Entire Flash Section, Application Code, and/or Boot Section
- Selectable NMI Trigger on Failure
- User-Configurable Check During Internal Reset Initialization

## 30.2 Overview

The Cyclic Redundancy Check (CRC) is an important safety feature. It scans the Nonvolatile Memory (NVM) making sure the code is correct.

The device will not execute code if Flash fault has occurred. By ensuring no code corruption has occurred, a potentially unintended behavior in the application that can cause a dangerous situation can be avoided. The CRC scan can be set up to scan the entire Flash, only the boot section, or both the boot and application code sections.

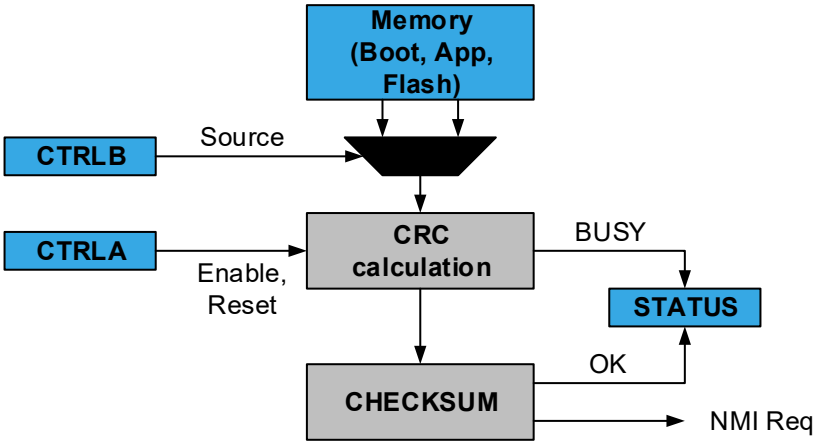
The CRC generates a checksum that is compared to a pre-calculated one. If the two checksums match, the Flash is OK, and the application code can start running.

The BUSY bit in the Status (CRCSCAN.STATUS) register indicates if a CRC scan is ongoing or not, while the OK bit in the Status (CRCSCAN.STATUS) register indicates if the checksum comparison matches or not.

The CRCSCAN can be set up to generate a Non-Maskable Interrupt (NMI) if the checksums do not match.

### 30.2.1 Block Diagram

**Figure 30-1. Cyclic Redundancy Check Block Diagram**




## 30.3 Functional Description

### 30.3.1 Initialization

To enable a CRC in software (or via the debugger):

1. Write the Source (SRC) bit field of the Control B (CRCSCAN.CTRLB) register to select the desired source settings.
2. Enable the CRCSCAN by writing a '1' to the ENABLE bit in the Control A (CRCSCAN.CTRLA) register.
3. The CRC will start after three cycles. The CPU will continue executing during these three cycles.

Selection between CRC32 and CRC16 is done through fuse settings. The CRCSCAN can be configured to perform a code memory scan before the device leaves Reset. If this check fails, the CPU is not allowed to start normal code execution. This feature is enabled and controlled by the CRCSRC field in FUSE.SYSCFG0 (see the *Fuses* section for more information).

If the CRCSCAN is enabled, a successful CRC check will have the following outcome:

- Normal code execution starts
- The ENABLE bit in CRCSCAN.CTRLA will be '1'
- The SRC bit field in CRCSCAN.CTRLB will reflect the checked section(s)
- The OK flag in CRCSCAN.STATUS will be '1'

If the CRCSCAN is enabled, a non-successful CRC check will have the following outcome:

- Normal code execution does not start. The CPU will hang executing no code.
- The ENABLE bit in CRCSCAN.CTRLA will be '1'
- The SRC bit field in CRCSCAN.CTRLB will reflect the checked section(s)
- The OK flag in CRCSCAN.STATUS will be '0'
- This condition may be observed using the debug interface

### 30.3.2 Operation

When operating, the CRCSCAN has priority access to the Flash and will stall the CPU until completed.

The CRC will use three clock cycles for each 16-bit fetch. The CRCSCAN can be configured to do a scan from start-up.

An *n*-bit CRC applied to a data block of arbitrary length will detect any single alteration (error burst) up to *n* bits in length. For longer error bursts a fraction $1-2^{-n}$ will be detected.

The CRC generator supports CRC-16-CCITT and CRC-32 (IEEE 802.3).

The polynomial options are:

- CRC-16-CCITT: $x^{16} + x^{12} + x^5 + 1$
- CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

The CRC reads byte-by-byte the content of the section(s) it is set up to check, starting with byte 0, and generates a new checksum per byte. The byte is sent through a shift register as depicted below, starting with the Most Significant bit. If the last bytes in the section contain the correct checksum, the CRC will pass. See 30.3.2.1 Checksum for how to place the checksum. The initial value of the Checksum register is 0xFFFF.

#### 30.3.2.1 Checksum

The pre-calculated checksum must be present in the last location of the section to be checked. If the BOOT section is to be checked, the checksum must be saved in the last bytes of the BOOT section. The same is done for APPLICATION and the entire Flash. Table 30-1 shows explicitly how the checksum must be stored for the different sections. Refer to the CRCSCAN.CTRLB register description for how to configure the sections to be checked.

**Table 30-1. Placement of the Pre-Calculated Checksum for CRC16 in Flash**

| Section to Check | CHECKSUM[15:8] | CHECKSUM[7:0] |
|---|---|---|
| BOOT | BOOTEND-1 | BOOTEND |
| BOOT and APPLICATION | APPEND-1 | APPEND |
| Full Flash | FLASHEND-1 | FLASHEND |

**Table 30-2. Placement of the Pre-Calculated Checksum for CRC32 in Flash**

| Section to Check | CHECKSUM[31:24] | CHECKSUM[23:16] | CHECKSUM[15:8] | CHECKSUM[7:0] |
|---|---|---|---|---|
| BOOT | BOOTEND | BOOTEND-1 | BOOTEND-2 | BOOTEND-3 |
| BOOT and APPLICATION | APPEND | APPEND-1 | APPEND-2 | APPEND-3 |
| Full Flash | FLASHEND | FLASHEND-1 | FLASHEND-2 | FLASHEND-3 |

### 30.3.3 Interrupts

**Table 30-3. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| NMI | Non-Maskable Interrupt | CRC failure |

When the interrupt condition occurs the OK flag in the Status (CRCSCAN.STATUS) register is cleared to '0'.

A Non-Maskable Interrupt (NMI) is enabled by writing a '1' to the respective Enable (NMIEN) bit in the Control A (CRCSCAN.CTRLA) register, but can only be disabled with a System Reset. An NMI is generated when the OK flag in the CRCSCAN.STATUS register is cleared, and the NMIEN bit is '1'. The NMI request remains active until a System Reset and cannot be disabled.

An NMI can be triggered even if interrupts are not globally enabled.

### 30.3.4 Sleep Mode Operation

In all CPU Sleep modes, the CRCSCAN is halted and will resume operation when the CPU wakes up.

The CRCSCAN starts operation three cycles after writing the Enable (ENABLE) bit in the Control A (CRCSCAN.CTRLA) register. During these three cycles, it is possible to enter Sleep mode. In this case:

1. The CRCSCAN will not start until the CPU is woken up.
2. Any interrupt handler will execute after CRCSCAN has finished.

### 30.3.5 Debug Operation

Whenever the debugger reads or writes a peripheral or memory location, the CRCSCAN will be disabled.

If the CRCSCAN is busy when the debugger accesses the device, the CRCSCAN will restart the ongoing operation when the debugger accesses an internal register or when the debugger disconnects.

The BUSY bit in the Status (CRCSCAN.STATUS) register will read '1' if the CRCSCAN was busy when the debugger caused it to disable, but it will not actively check any section as long as the debugger keeps it disabled. There are synchronized CRC status bits in the debugger's internal register space, which can be read by the debugger without disabling the CRCSCAN. Reading the debugger's internal CRC status bits will make sure that the CRCSCAN is enabled.

It is possible to write the CRCSCAN.STATUS register directly from the debugger:

- BUSY bit in CRCSCAN.STATUS:
  - Writing the BUSY bit to '0' will stop the ongoing CRC operation (so that the CRCSCAN does not restart its operation when the debugger allows it).
  - Writing the BUSY bit to '1' will make the CRC start a single check with the settings in the Control B (CRCSCAN.CTRLB) register, but not until the debugger allows it.

  As long as the BUSY bit in CRCSCAN.STATUS is '1', CRCSCAN.CTRLB and the Non-Maskable Interrupt Enable (NMIEN) bit in the Control A (CRCSCAN.CTRLA) register cannot be altered.
- OK bit in CRCSCAN.STATUS:
  - Writing the OK bit to '0' can trigger a Non-Maskable Interrupt (NMI) if the NMIEN bit in CRCSCAN.CTRLA is '1'. If an NMI has been triggered, no writes to the CRCSCAN are allowed.
  - Writing the OK bit to '1' will make the OK bit read as '1' when the BUSY bit in CRCSCAN.STATUS is '0'.

Writes to CRCSCAN.CTRLA and CRCSCAN.CTRLB from the debugger are treated in the same way as writes from the CPU.

## 30.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | RESET | | | | | | NMIEN | ENABLE |
| 0x01 | CTRLB | 7:0 | | | | | | | SRC[1:0] | |
| 0x02 | STATUS | 7:0 | | | | | | | OK | BUSY |

## 30.5 Register Description

### 30.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

If an NMI has been triggered this register is not writable.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RESET | | | | | | NMIEN | ENABLE |
| Access | R/W | | | | | | R/W | R/W |
| Reset | 0 | | | | | | 0 | 0 |

**Bit 7 – RESET** Reset CRCSCAN
Writing this bit to '1' resets the CRCSCAN. The CRCSCAN Control and Status (CRCSCAN.CTRLA, CRCSCAN.CTRLB, CRCSCAN.STATUS) register will be cleared one clock cycle after the RESET bit is written to '1'.
If NMIEN is '0', this bit is writable both when the CRCSCAN is busy (the BUSY bit in CRCSCAN.STATUS is '1') and not busy (the BUSY bit is '0'), and will take effect immediately.
If NMIEN is '1', this bit is only writable when the CRCSCAN is not busy (the BUSY bit in CRCSCAN.STATUS is '0').
The RESET bit is a strobe bit.

**Bit 1 – NMIEN** Enable NMI Trigger
When this bit is written to '1', any CRC failure will trigger an NMI.
This bit can only be cleared by a System Reset. It is not cleared by a write to the RESET bit.
This bit can only be written to '1' when the CRCSCAN is not busy (the BUSY bit in CRCSCAN.STATUS is '0').

**Bit 0 – ENABLE** Enable CRCSCAN
Writing this bit to '1' enables the CRCSCAN with the current settings. It will stay '1' even after a CRC check has completed, but writing it to '1' again will start a new check.
Writing the bit to '0' has no effect.
The CRCSCAN can be configured to run a scan during the microcontroller (MCU) start-up sequence to verify the Flash sections before letting the CPU start normal code execution (see the 30.3.1 Initialization section). If this feature is enabled, the ENABLE bit will read as '1' when normal code execution starts.
To see whether the CRCSCAN is busy with an ongoing check, poll the BUSY bit in the Status (CRCSCAN.STATUS) register.

### 30.5.2 Control B

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

The Control B register contains the source settings for the CRC. It is not writable when the CRCSCAN is busy, or when an NMI has been triggered.

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

**Bits 1:0 – SRC[1:0]** CRC Source
The SRC bit field selects which section of the Flash will be checked by the CRCSCAN. To set up section sizes, refer to the *Fuses* section.
The CRCSCAN can be enabled during internal Reset initialization to verify Flash sections before letting the CPU start (see the *Fuses* section). If the CRCSCAN is enabled during internal Reset initialization, the SRC bit field will read out as FLASH, BOOTAPP, or BOOT when normal code execution starts (depending on the configuration).

| Value | Name | Description |
|---|---|---|
| 0x0 | FLASH | The CRC is performed on the entire Flash (boot, application code, and application data sections). |
| 0x1 | BOOTAPP | The CRC is performed on the boot and application code sections of Flash. |
| 0x2 | BOOT | The CRC is performed on the boot section of Flash. |
| 0x3 | - | Reserved. |

### 30.5.3 Status

**Name:** STATUS
**Offset:** 0x02
**Reset:** 0x02
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | OK | BUSY |
| Access | | | | | | | R | R |
| Reset | | | | | | | 1 | 0 |

**Bit 1 – OK** CRC OK
When this bit is read as '1', the previous CRC completed successfully. The bit is set to '1' by default before a CRC scan is run. The bit is not valid unless BUSY is '0'.

**Bit 0 – BUSY** CRC Busy
When this bit is read as '1', the CRCSCAN is busy. As long as the module is busy, the access to the control registers is limited.

# 31. CCL – Configurable Custom Logic

## 31.1 Features

- Glue Logic for General Purpose PCB Design
- 6 Programmable Look-Up Tables (LUTs)
- Combinatorial Logic Functions: Any Logic Expression which is a Function of up to Three Inputs.
- Sequencer Logic Functions:
  - Gated D flip-flop
  - JK flip-flop
  - Gated D latch
  - RS latch
- Flexible LUT Input Selection:
  - I/Os
  - Events
  - Subsequent LUT output
  - Internal peripherals such as:
    - Analog comparator
    - Timers/Counters
    - USART
    - SPI
- Clocked by a System Clock or other Peripherals
- Output can be Connected to I/O Pins or an Event System
- Optional Synchronizer, Filter, or Edge Detector Available on Each LUT Output
- Optional Interrupt Generation from Each LUT Output:
  - Rising edge
  - Falling edge
  - Both edges

## 31.2 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. The CCL can serve as ‘glue logic’ between the device peripherals and external devices. The CCL can eliminate the need for external logic components, and can also help the designer to overcome real-time constraints by combining Core Independent Peripherals (CIPs) to handle the most time-critical parts of the application independent of the CPU.

The CCL peripheral provides a number of Look-up Tables (LUTs). Each LUT consists of three inputs, a truth table, a synchronizer/filter, and an edge detector. Each LUT can generate an output as a user programmable logic expression with three inputs. The output is generated from the inputs using the combinatorial logic and can be filtered to remove spikes. The CCL can be configured to generate an interrupt request on changes in the LUT outputs.

Neighboring LUTs can be combined to perform specific operations. A sequencer can be used for generating complex waveforms.

### 31.2.1 Block Diagram

**Figure 31-1. Configurable Custom Logic**

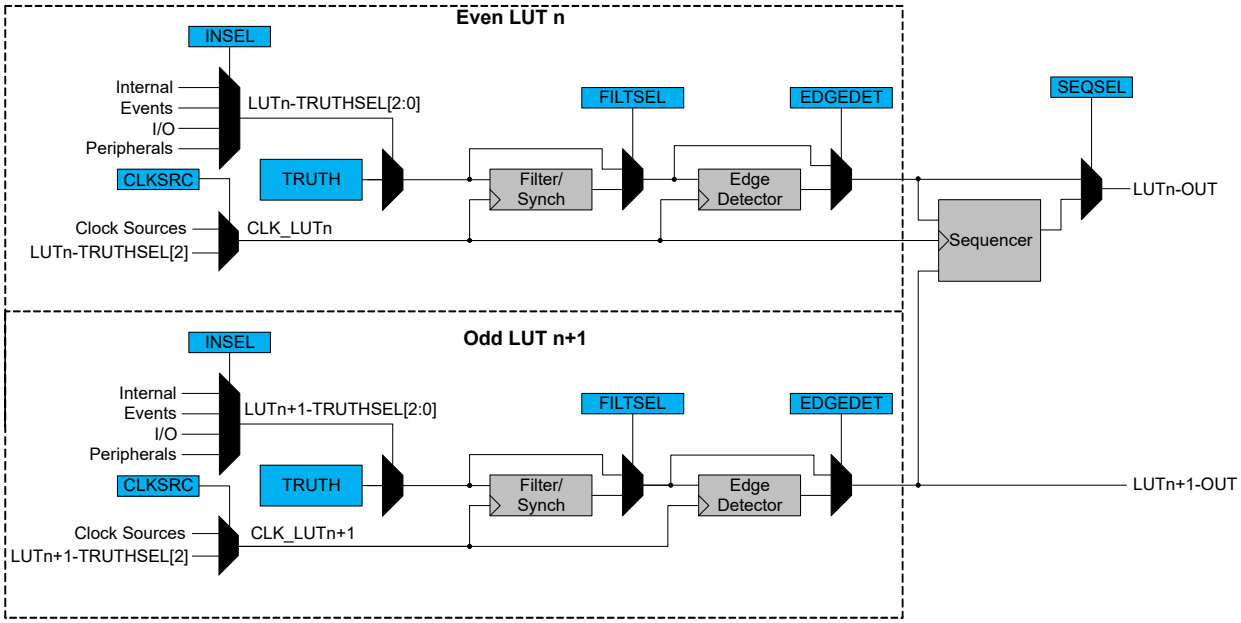



**Table 31-2. Sequencer and LUT Connection**

| Sequencer | Even and Odd LUT |
|---|---|
| SEQ0 | LUT0 and LUT1 |
| SEQ1 | LUT2 and LUT3 |
| SEQ2 | LUT4 and LUT5 |

### 31.2.2 Signal Description

| Name | Type | Description |
|---|---|---|
| LUTn-OUT | Digital output | Output from the look-up table |
| LUTn-IN[2:0] | Digital input | Input to the look-up table. LUTn-IN[2] can serve as CLK_LUTn. |

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

#### 31.2.2.1 CCL Input Selection MUX

The following peripherals outputs are available as inputs into the CCL LUT.

| Value | Input Source | INSEL0[3:0] | INSEL1[3:0] | INSEL2[3:0] |
|---|---|---|---|---|
| 0x00 | MASK | Masked input | | |
| 0x01 | FEEDBACK | LUTn | | |
| 0x02 | LINK | LUT[n+1] | | |
| 0x03 | EVENTA | EVENTA | | |
| 0x04 | EVENTB | EVENTB | | |
| 0x05 | INn | LUTn-IN0 | LUTn-IN1 | LUTn-IN2 |
| 0x06 | ACn | AC0 OUT | AC1 OUT | AC2 OUT |

.....continued

| Value | Input Source | INSEL0[3:0] | INSEL1[3:0] | INSEL2[3:0] |
|---|---|---|---|---|
| 0x07 | ZCDn | ZCD0 OUT | ZCD1 OUT | ZCD2 OUT |
| 0x08 | USARTn(1) | USART0 TXD | USART1 TXD | USART2 TXD |
| 0x09 | SPI0(2) | SPI0 MOSI | SPI0 MOSI | SPI0 SCK |
| 0x0A | TCA0 | WO0 | WO1 | WO2 |
| 0x0B | TCA1 | WO0 | WO1 | WO2 |
| 0x0C | TCBn | TCB0 WO | TCB1 WO | TCB2 WO |
| 0x0D | TCD0 | WOA | WOB | WOC |

**Notes:**

1. USART connections to the CCL work only in asynchronous/synchronous USART master mode.
2. SPI connections to the CCL work only in master SPI mode.

## 31.3 Functional Description

### 31.3.1 Operation

#### 31.3.1.1 Enable-Protected Configuration

The configuration of the LUTs and sequencers is enable-protected, meaning that they can only be configured when the corresponding even LUT is disabled (ENABLE=0 in the LUT n Control A register, CCL.LUTnCTRLA). This is a mechanism to suppress the undesired output from the CCL under (re-)configuration.

The following bits and registers are enable-protected:

- Sequencer Selection (SEQSEL) in the Sequencer Control n register (CCL.SEQCTRLn)
- LUT n Control x registers (CCL.LUTnCTRLx), except the ENABLE bit in CCL.LUTnCTRLA

The enable-protected bits in the CCL.LUTnCTRLx registers can be written at the same time as ENABLE in CCL.LUTnCTRLA is written to '1', but not at the same time as ENABLE is written to '0'.

The enable protection is denoted by the enable-protected property in the register description.

#### 31.3.1.2 Enabling, Disabling, and Resetting

The CCL is enabled by writing a '1' to the ENABLE bit in the Control register (CCL.CTRLA). The CCL is disabled by writing a '0' to that ENABLE bit.

Each LUT is enabled by writing a '1' to the LUT Enable bit (ENABLE) in the LUT n Control A register (CCL.LUTnCTRLA). Each LUT is disabled by writing a '0' to the ENABLE bit in CCL.LUTnCTRLA.

#### 31.3.1.3 Truth Table Logic

The truth table in each LUT unit can generate a combinational logic output as a function of up to three inputs (LUTn-TRUTHSEL[2:0]). The unused inputs can be turned off (tied low). The truth table for the combinational logic expression is defined by the bits in the CCL.TRUTHn registers. Each combination of the input bits (LUTn-TRUTHSEL[2:0]) corresponds to one bit in the TRUTHn register, as shown in the table below.

**Figure 31-2. Truth Table Output Value Selection of a LUT**

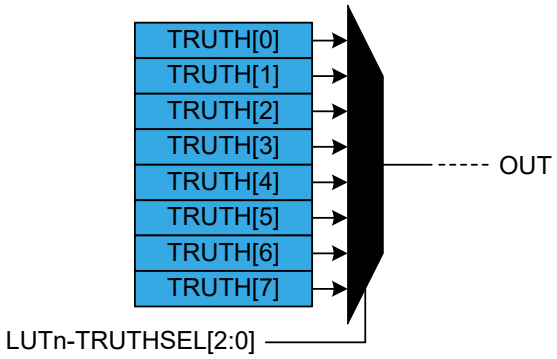



**Table 31-3. Truth Table of a LUT**

| LUTn-TRUTHSEL[2] | LUTn-TRUTHSEL[1] | LUTn-TRUTHSEL[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] |

### 31.3.1.4 Truth Table Inputs Selection

**Input Overview**

The inputs can be individually:

- OFF
- Driven by peripherals
- Driven by internal events from the Event System
- Driven by I/O pin inputs
- Driven by other LUTs

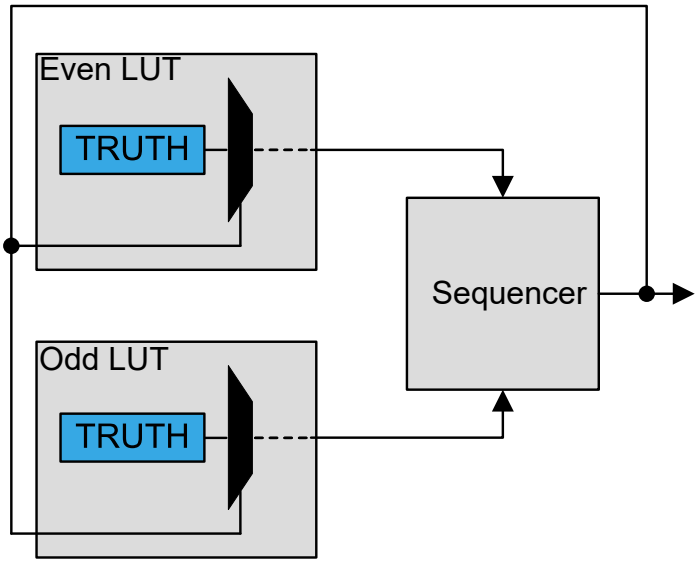
The input for each LUT is configured by writing the Input Source Selection bits in the LUT Control registers:

- INSEL0 in CCL.LUTnCTRLB
- INSEL1 in CCL.LUTnCTRLB
- INSEL2 in CCL.LUTnCTRLC

**Internal Feedback Inputs (FEEDBACK)**

The output from a sequencer can be used as an input source for the two LUTs it is connected to.

**Figure 31-3. Feedback Input Selection**




When selected (INSELy=FEEDBACK in LUTnCTRLx), the sequencer (SEQ) output is used as input for the corresponding LUTs.

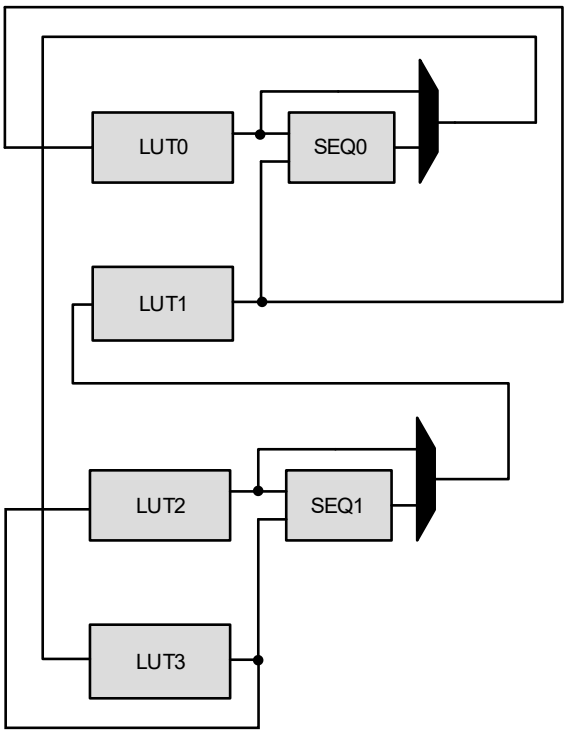
**Linked LUT (LINK)**

When selecting the LINK input option, the next LUT's direct output is used as LUT input. In general, LUT[n+1] is linked to the input of LUT[n]. LUT0 is linked to the input of the last LUT.

**Example 31-1. Linking all LUTs on a Device with Four LUTs**

- LUT1 is the input for LUT0
- LUT2 is the input for LUT1
- LUT3 is the input for LUT2
- LUT0 is the input for LUT3 (wrap-around)

**Figure 31-4. Linked LUT Input Selection**




**Event Input Selection (EVENTx)**
Events from the Event System can be used as inputs to the LUTs by writing to the INSELn bit groups in the LUT n Control B and C registers.

**I/O Pin Inputs (IO)**
When selecting the IO option, the LUT input will be connected to its corresponding I/O pin. Refer to the I/O Multiplexing section in the data sheet for more details about where the LUTn-INy pins are located.

**Peripherals**
The different peripherals on the three input lines of each LUT are selected by writing to the Input Select (INSEL) bits in the LUT Control registers (LUTnCTRLB and LUTnCTRLC).

#### 31.3.1.5 Filter

By default, the LUT output is a combinational function of the LUT inputs. This may cause some short glitches when the inputs change the value. These glitches can be removed by clocking through filters if demanded by application needs.

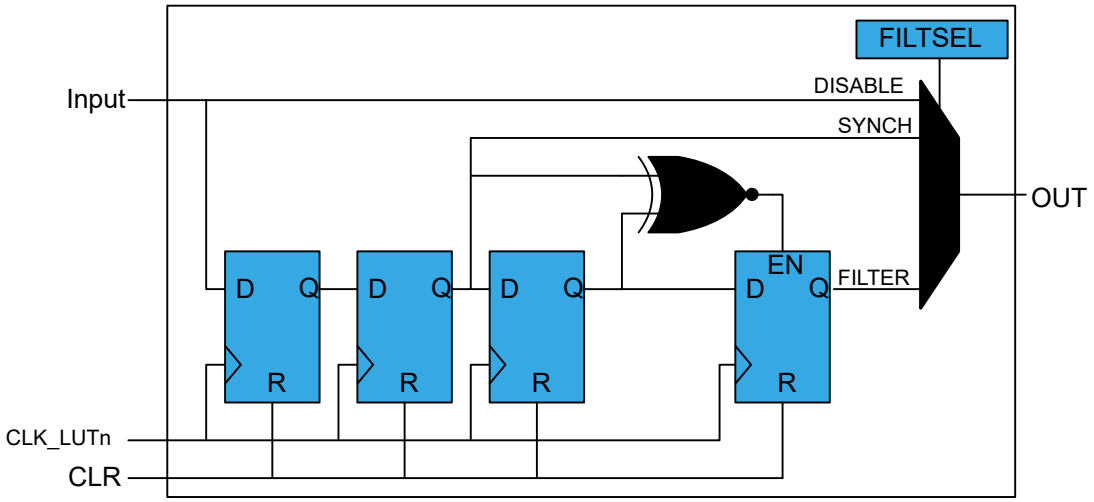
The Filter Selection bits (FILTSEL) in the LUT n Control A registers (CCL.LUTnCTRLA) define the digital filter options.

When FILTSEL=SYNCH, the output is synchronized with CLK_LUTn. The output will be delayed by two positive CLK_LUTn edges.

When FILTSEL=FILTER, only the input that is persistent for more than two positive CLK_LUTn edges will pass through the gated flip-flop to the output. The output will be delayed by four positive CLK_LUTn edges.

One clock cycle later, after the corresponding LUT is disabled, all internal filter logic is cleared.

**Figure 31-5. Filter**



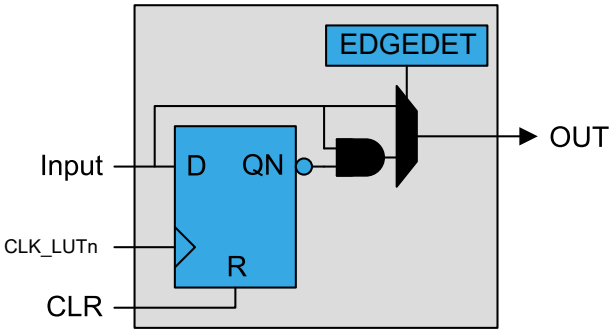

### 31.3.1.6 Edge Detector

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 can be programmed to provide inverted output.

The edge detector is enabled by writing '1' to the Edge Detection bit (EDGEDET) in the LUT n Control A register (CCL.LUTnCTRLA). In order to avoid unpredictable behavior, a valid filter option must be enabled.

The edge detection is disabled by writing a '0' to EDGEDET in CCL.LUTnCTRLA. After disabling a LUT, the corresponding internal edge detector logic is cleared one clock cycle later.

**Figure 31-6. Edge Detector**




### 31.3.1.7 Sequencer Logic

Each LUT pair can be connected to a sequencer. The sequencer can function as either D flip-flop, JK flip-flop, gated D latch, or RS latch. The function is selected by writing the Sequencer Selection (SEQSEL) bit group in the Sequencer Control register (CCL.SEQCTRLn).

The sequencer receives its input from either the LUT, filter or edge detector, depending on the configuration.

A sequencer is clocked by the same clock as the corresponding even LUT. The clock source is selected by the Clock Source (CLKSRC) bit group in the LUT n Control A register (CCL.LUTnCTRLA).

The flip-flop output (OUT) is refreshed on the rising edge of the clock. When the even LUT is disabled, the latch is cleared asynchronously. The flip-flop Reset signal (R) is kept enabled for one clock cycle.

**Gated D Flip-Flop (DFF)**

The D input is driven by the even LUT output, and the G input is driven by the odd LUT output.

**Figure 31-7. D Flip-Flop**

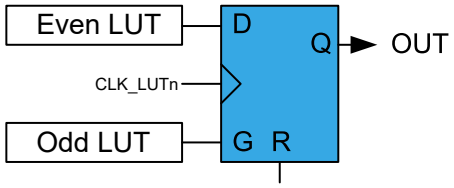



**Table 31-4. DFF Characteristics**

| R | G | D | OUT |
|---|---|---|---|
| 1 | X | X | Clear |
| 0 | 1 | 1 | Set |
| 0 | 1 | 0 | Clear |
| 0 | 0 | X | Hold state (no change) |

**JK Flip-Flop (JK)**
The J input is driven by the even LUT output, and the K input is driven by the odd LUT output.

**Figure 31-8. JK Flip-Flop**

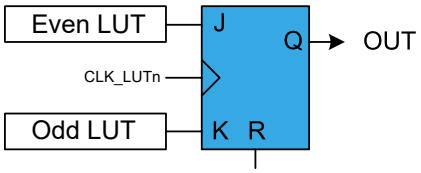



**Table 31-5. JK Characteristics**

| 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)**
The D input is driven by the even LUT output, and the G input is driven by the odd LUT output.

**Figure 31-9. D Latch**

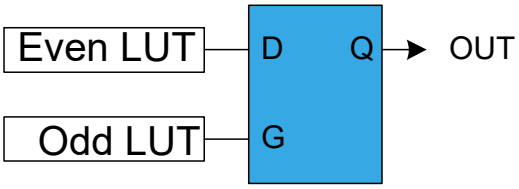



**Table 31-6. D Latch Characteristics**

| G | D | OUT |
|---|---|---|
| 0 | X | Hold state (no change) |
| 1 | 0 | Clear |

| .....continued | | |
|---|---|---|
| G | D | OUT |
| 1 | 1 | Set |

**RS Latch (RS)**
The S input is driven by the even LUT output, and the R input is driven by the odd LUT output.

**Figure 31-10. RS Latch**

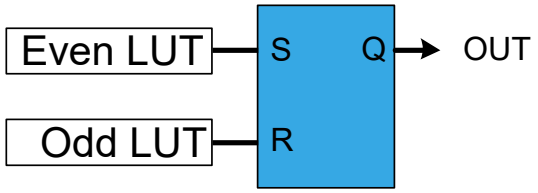


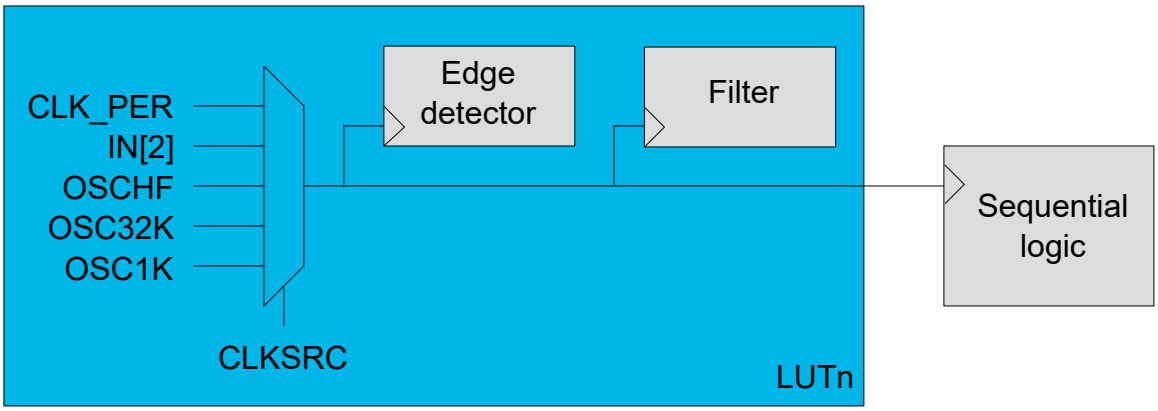

**Table 31-7. RS Latch Characteristics**

| S | R | OUT |
|---|---|---|
| 0 | 0 | Hold state (no change) |
| 0 | 1 | Clear |
| 1 | 0 | Set |
| 1 | 1 | Forbidden state |

### 31.3.1.8 Clock Source Settings

The filter, edge detector, and sequencer are, by default, clocked by the peripheral clock (CLK_PER). It is also possible to use other clock inputs (CLK_LUTn) to clock these blocks. This is configured by writing the Clock Source (CLKSRC) bits in the LUT Control A register.

**Figure 31-11. Clock Source Settings**




When the Clock Source (CLKSRC) bit is written to 0x1, LUTn-TRUTHSEL[2] is used to clock the corresponding filter and edge detector (CLK_LUTn). The sequencer is clocked by the CLK_LUTn of the even LUT in the pair. When CLKSRC is written to 0x1, LUTn-TRUTHSEL[2] is treated as OFF (low) in the TRUTH table.

The CCL peripheral must be disabled while changing the clock source to avoid undefined outputs from the peripheral.

### 31.3.2 Interrupts

**Table 31-8. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| CCL | CCL interrupt | INTn in INTFLAG is raised as configured by the INTMODEn bits in the CCL.INTCTRLn register |

When an interrupt condition occurs, the corresponding interrupt flag is set in the peripheral's Interrupt Flags (*peripheral*.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the peripheral's Interrupt Control (*peripheral*.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled, and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

When several interrupt request conditions are supported by an interrupt vector, the interrupt requests are ORed together into one combined interrupt request to the interrupt controller. The user must read the peripheral's INTFLAGS register to determine which of the interrupt conditions are present.

### 31.3.3 Events

The CCL can generate the events shown in the table below.

**Table 31-9. Event Generators in the CCL**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| CCL | LUTn | LUT output level | Level | Asynchronous | Depends on the CCL configuration |

The CCL has the event users below for detecting and acting upon input events.

**Table 31-10. Event Users in the CCL**

| User Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| CCL | LUTnx | LUTn input x or clock signal | No detection | Async |

The event signals are passed directly to the LUTs without synchronization or input detection logic.

Two event users are available for each LUT. They can be selected as LUTn inputs by writing to the INSELn bit groups in the LUT n Control B and Control C registers (CCL.LUTnCTRLB or LUTnCTRLC).

Refer to the Event System (EVSYS) section for more details regarding the event types and the EVSYS configuration.

### 31.3.4 Sleep Mode Operation

Writing the Run In Standby bit (RUNSTDBY) in the Control A register (CCL.CTRLA) to '1' will allow the selected clock source to be enabled in Standby Sleep mode.

If RUNSTDBY is '0' the peripheral clock will be disabled in Standby Sleep mode. If the filter, edge detector, and/or sequencer are enabled, the LUT output will be forced to '0' in Standby Sleep mode. In Idle Sleep mode, the TRUTH table decoder will continue the operation and the LUT output will be refreshed accordingly, regardless of the RUNSTDBY bit.

If the Clock Source bit (CLKSRC) in the LUT n Control A register (CCL.LUTnCTRLA) is written to '1', the LUTn-TRUTHSEL[2] will always clock the filter, edge detector, and sequencer. The availability of the LUTn-TRUTHSEL[2] clock in sleep modes will depend on the sleep settings of the peripheral used.

## 31.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | RUNSTDBY | | | | | | ENABLE |
| 0x01 | SEQCTRL0 | 7:0 | | | | | SEQSEL0[3:0] | | | |
| 0x02 | SEQCTRL1 | 7:0 | | | | | SEQSEL1[3:0] | | | |
| 0x03 | SEQCTRL2 | 7:0 | | | | | SEQSEL2[3:0] | | | |
| 0x04 | Reserved | | | | | | | | | |
| 0x05 | INTCTRL0 | 7:0 | INTMODE3[1:0] | | INTMODE2[1:0] | | INTMODE1[1:0] | | INTMODE0[1:0] | |
| 0x06 | INTCTRL1 | 7:0 | | | | | INTMODE5[1:0] | | INTMODE4[1:0] | |
| 0x07 | INTFLAGS | 7:0 | | | INT5 | INT4 | INT3 | INT2 | INT1 | INT0 |
| 0x08 | LUT0CTRLA | 7:0 | EDGEDET | OUTEN | FILTSEL[1:0] | | CLKSRC[2:0] | | | ENABLE |
| 0x09 | LUT0CTRLB | 7:0 | INSEL1[3:0] | | | | INSEL0[3:0] | | | |
| 0x0A | LUT0CTRLC | 7:0 | | | | | INSEL2[3:0] | | | |
| 0x0B | TRUTH0 | 7:0 | TRUTH[7:0] | | | | | | | |
| 0x0C | LUT1CTRLA | 7:0 | EDGEDET | OUTEN | FILTSEL[1:0] | | CLKSRC[2:0] | | | ENABLE |
| 0x0D | LUT1CTRLB | 7:0 | INSEL1[3:0] | | | | INSEL0[3:0] | | | |
| 0x0E | LUT1CTRLC | 7:0 | | | | | INSEL2[3:0] | | | |
| 0x0F | TRUTH1 | 7:0 | TRUTH[7:0] | | | | | | | |
| 0x10 | LUT2CTRLA | 7:0 | EDGEDET | OUTEN | FILTSEL[1:0] | | CLKSRC[2:0] | | | ENABLE |
| 0x11 | LUT2CTRLB | 7:0 | INSEL1[3:0] | | | | INSEL0[3:0] | | | |
| 0x12 | LUT2CTRLC | 7:0 | | | | | INSEL2[3:0] | | | |
| 0x13 | TRUTH2 | 7:0 | TRUTH[7:0] | | | | | | | |
| 0x14 | LUT3CTRLA | 7:0 | EDGEDET | OUTEN | FILTSEL[1:0] | | CLKSRC[2:0] | | | ENABLE |
| 0x15 | LUT3CTRLB | 7:0 | INSEL1[3:0] | | | | INSEL0[3:0] | | | |
| 0x16 | LUT3CTRLC | 7:0 | | | | | INSEL2[3:0] | | | |
| 0x17 | TRUTH3 | 7:0 | TRUTH[7:0] | | | | | | | |
| 0x18 | LUT4CTRLA | 7:0 | EDGEDET | OUTEN | FILTSEL[1:0] | | CLKSRC[2:0] | | | ENABLE |
| 0x19 | LUT4CTRLB | 7:0 | INSEL1[3:0] | | | | INSEL0[3:0] | | | |
| 0x1A | LUT4CTRLC | 7:0 | | | | | INSEL2[3:0] | | | |
| 0x1B | TRUTH4 | 7:0 | TRUTH[7:0] | | | | | | | |
| 0x1C | LUT5CTRLA | 7:0 | EDGEDET | OUTEN | FILTSEL[1:0] | | CLKSRC[2:0] | | | ENABLE |
| 0x1D | LUT5CTRLB | 7:0 | INSEL1[3:0] | | | | INSEL0[3:0] | | | |
| 0x1E | LUT5CTRLC | 7:0 | | | | | INSEL2[3:0] | | | |
| 0x1F | TRUTH5 | 7:0 | TRUTH[7:0] | | | | | | | |

## 31.5 Register Description

### 31.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | RUNSTDBY | | | | | | ENABLE |
| Access | | R/W | | | | | | R/W |
| Reset | | 0 | | | | | | 0 |

**Bit 6 – RUNSTDBY** Run in Standby
Writing this bit to '1' will enable the peripheral to run in Standby Sleep mode.

| Value | Description |
|---|---|
| 0 | The CCL will not run in Standby Sleep mode |
| 1 | The CCL will run in Standby Sleep mode |

**Bit 0 – ENABLE** Enable

| Value | Description |
|---|---|
| 0 | The peripheral is disabled |
| 1 | The peripheral is enabled |

### 31.5.2 Sequencer Control 0

**Name:** SEQCTRL0
**Offset:** 0x01
**Reset:** 0x00
**Property:** Enable-Protected

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SEQSEL0[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – SEQSEL0[3:0]** Sequencer Selection
This bit group selects the sequencer configuration for LUT0 and LUT1.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLE | The sequencer is disabled |
| 0x1 | DFF | D flip-flop |
| 0x2 | JK | JK flip-flop |
| 0x3 | LATCH | D latch |
| 0x4 | RS | RS latch |
| Other | - | Reserved |

### 31.5.3 Sequencer Control 1

**Name:** SEQCTRL1
**Offset:** 0x02
**Reset:** 0x00
**Property:** Enable-Protected

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SEQSEL1[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – SEQSEL1[3:0]** Sequencer Selection
This bit group selects the sequencer configuration for LUT2 and LUT3.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLE | The sequencer is disabled |
| 0x1 | DFF | D flip-flop |
| 0x2 | JK | JK flip-flop |
| 0x3 | LATCH | D latch |
| 0x4 | RS | RS latch |
| Other | - | Reserved |

### 31.5.4 Sequencer Control 2

**Name:** SEQCTRL2
**Offset:** 0x03
**Reset:** 0x00
**Property:** Enable-Protected

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SEQSEL2[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – SEQSEL2[3:0]** Sequencer Selection
This bit group selects the sequencer configuration for LUT4 and LUT5.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLE | The sequencer is disabled |
| 0x1 | DFF | D flip-flop |
| 0x2 | JK | JK flip-flop |
| 0x3 | LATCH | D latch |
| 0x4 | RS | RS latch |
| Other | - | Reserved |

### 31.5.5 Interrupt Control 0

**Name:** INTCTRL0
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INTMODE3[1:0] | | INTMODE2[1:0] | | INTMODE1[1:0] | | INTMODE0[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 0:1, 2:3, 4:5, 6:7 – INTMODE**
The bits in INTMODEn select the interrupt sense configuration for LUTn-OUT.

| Value | Name | Description |
|---|---|---|
| 0x0 | INTDISABLE | Interrupt disabled |
| 0x1 | RISING | Sense rising edge |
| 0x2 | FALLING | Sense falling edge |
| 0x3 | BOTH | Sense both edges |

### 31.5.6 Interrupt Control 1

**Name:** INTCTRL1
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | INTMODE5[1:0] | | INTMODE4[1:0] | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 0:1, 2:3 – INTMODE**
The bits in INTMODEn select the interrupt sense configuration for LUTn-OUT.

| Value | Name | Description |
|---|---|---|
| 0x0 | INTDISABLE | Interrupt disabled |
| 0x1 | RISING | Sense rising edge |
| 0x2 | FALLING | Sense falling edge |
| 0x3 | BOTH | Sense both edges |

### 31.5.7 Interrupt Flag

**Name:** INTFLAGS
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | INT5 | INT4 | INT3 | INT2 | INT1 | INT0 |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5 – INT** Interrupt Flag
The INTn flag is set when the LUTn output change matches the Interrupt Sense mode as defined in CCL.INTCTRLn. Writing a '`1`' to this flag's bit location will clear the flag.

### 31.5.8 LUT n Control A

**Name:** LUTnCTRLA
**Offset:** 0x08 + n*0x04 [n=0..5]
**Reset:** 0x00
**Property:** Enable-Protected

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EDGEDET | OUTEN | FILTSEL[1:0] | | CLKSRC[2:0] | | | ENABLE |
| Access | R/W | 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 – EDGEDET** Edge Detection

| Value | Description |
|---|---|
| 0 | Edge detector is disabled |
| 1 | Edge detector is enabled |

**Bit 6 – OUTEN** Output Enable
This bit enables the LUT output to the LUTn OUT pin. When written to '1', the pin configuration of the PORT I/O-Controller is overridden.

| Value | Description |
|---|---|
| 0 | Output to pin disabled |
| 1 | Output to pin enabled |

**Bits 5:4 – FILTSEL[1:0]** Filter Selection
These bits select the LUT output filter options.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLE | Filter disabled |
| 0x1 | SYNCH | Synchronizer enabled |
| 0x2 | FILTER | Filter enabled |
| 0x3 | - | Reserved |

**Bits 3:1 – CLKSRC[2:0]** Clock Source Selection
This bit selects between various clock sources to be used as the clock (CLK_LUTn) for a LUT.
The CLK_LUTn of the even LUT is used for clocking the sequencer of a LUT pair.

| Value | Input Source | Description |
|---|---|---|
| 0x0 | CLKPER | CLK_PER is clocking the LUT |
| 0x1 | IN2 | IN2 is clocking the LUT |
| 0x2 | - | Reserved |
| 0x3 | - | Reserved |
| 0x4 | OSCHF | Internal high-frequency oscillator before prescaler is clocking LUT |
| 0x5 | OSC32K | Internal 32.786 kHz oscillator |
| 0x6 | OSC1K | Internal 32.768 kHz oscillator divided by 32 |
| 0x07 | - | Reserved |

**Bit 0 – ENABLE** LUT Enable

| Value | Description |
|---|---|
| 0 | The LUT is disabled |
| 1 | The LUT is enabled |

### 31.5.9 LUT n Control B

**Name:** LUTnCTRLB
**Offset:** 0x09 + n*0x04 [n=0..5]
**Reset:** 0x00
**Property:** Enable-Protected

**Notes:**
1. SPI connections to the CCL work in master SPI mode only.
2. USART connections to the CCL work only when the USART is in one of the following modes:
   – Asynchronous USART
   – Synchronous USART master

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | 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 |

**Bits 7:4 – INSEL1[3:0]** LUT n Input 1 Source Selection
These bits select the source for input 1 of LUT n.

| Value | Name | Description |
|---|---|---|
| 0x0 | MASK | Masked input |
| 0x1 | FEEDBACK | Feedback input |
| 0x2 | LINK | Output from LUT[n+1] as input source |
| 0x3 | EVENTA | Event A as input source |
| 0x4 | EVENTB | Event B as input source |
| 0x5 | IN1 | LUTn-IN1 as input source |
| 0x6 | AC1 | AC1 OUT as input source |
| 0x7 | ZCD1 | ZCD1 OUT as input source |
| 0x8 | USART1 | USART1 TXD as input source |
| 0x9 | SPI0 | SPI0 MOSI as input source |
| 0xA | TCA0 | TCA0 WO1 as input source |
| 0xB | TCA1 | TCA1 WO1 as input source |
| 0xC | TCB1 | TCB1 WO as input source |
| 0xD | TCD0 | TCD0 WOB as input source |
| Other | - | Reserved |

**Bits 3:0 – INSEL0[3:0]** LUT n Input 0 Source Selection
These bits select the source for input 0 of LUT n.

| Value | Name | Description |
|---|---|---|
| 0x0 | MASK | Masked input |
| 0x1 | FEEDBACK | Feedback input |
| 0x2 | LINK | Output from LUT[n+1] as input source |
| 0x3 | EVENTA | Event A as input source |
| 0x4 | EVENTB | Event B as input source |
| 0x5 | IN0 | LUTn-IN0 as input source |
| 0x6 | AC0 | AC0 OUT as input source |
| 0x7 | ZCD0 | ZCD0 OUT as input source |
| 0x8 | USART0 | USART0 TXD as input source |
| 0x9 | SPI0 | SPI0 MOSI as input source |
| 0xA | TCA0 | TCA0 WO0 as input source |
| 0xB | TCA1 | TCA1 WO0 as input source |
| 0xC | TCB0 | TCB0 WO as input source |
| 0xD | TCD0 | TCD0 WOA as input source |

| .....continued | | |
|---|---|---|
| **Value** | **Name** | **Description** |
| `Other` | - | Reserved |

### 31.5.10 LUT n Control C

**Name:** LUTnCTRLC
**Offset:** 0x0A + n*0x04 [n=0..5]
**Reset:** 0x00
**Property:** Enable-Protected

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | INSEL2[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – INSEL2[3:0]** LUT n Input 2 Source Selection
These bits select the source for input 2 of LUT n.

| Value | Name | Description |
|---|---|---|
| 0x0 | MASK | Masked input |
| 0x1 | FEEDBACK | Feedback input |
| 0x2 | LINK | Output from LUT[n+1] as input source |
| 0x3 | EVENTA | Event A as input source |
| 0x4 | EVENTB | Event B as input source |
| 0x5 | IN2 | LUTn-IN2 as input source |
| 0x6 | AC2 | AC2 OUT as input source |
| 0x7 | ZCD2 | ZCD2 OUT as input source |
| 0x8 | USART2 | USART2 TXD as input source |
| 0x9 | SPI0 | SPI0 SCK as input source |
| 0xA | TCA0 | TCA0 WO2 as input source |
| 0xB | TCA1 | TCA1 WO2 as input source |
| 0xC | TCB2 | TCB2 WO as input source |
| 0xD | TCD0 | TCD0 WOC as input source |
| Other | - | Reserved |

### 31.5.11 TRUTHn

**Name:** TRUTHn
**Offset:** 0x0B + n*0x04 [n=0..5]
**Reset:** 0x00
**Property:** Enable-Protected

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | 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 |

**Bits 7:0 – TRUTH[7:0]** Truth Table
These bits define the value of truth logic as a function of inputs LUTn-TRUTHSEL[2:0]. See also section *Truth Table Logic*.

# 32. AC - Analog Comparator

## 32.1 Features

- Selectable Response Time
- Selectable Hysteresis
- Analog Comparator Output Available on Pin
- Comparator Output Inversion Available
- Flexible Input Selection:
  - Four Positive pins
  - Three Negative pins
  - Internal reference voltage generator (DACREF)
- Interrupt Generation on:
  - Rising edge
  - Falling edge
  - Both edges
- Window Function Interrupt Generation on:
  - Signal above window
  - Signal inside window
  - Signal below window
  - Signal outside window
- Event Generation:
  - Comparator output
  - Window function

## 32.2 Overview

The analog comparator (AC) compares the voltage levels on two inputs and gives a digital output based on this comparison. The AC can be configured to generate interrupt requests and/or events based on several different combinations of input change.

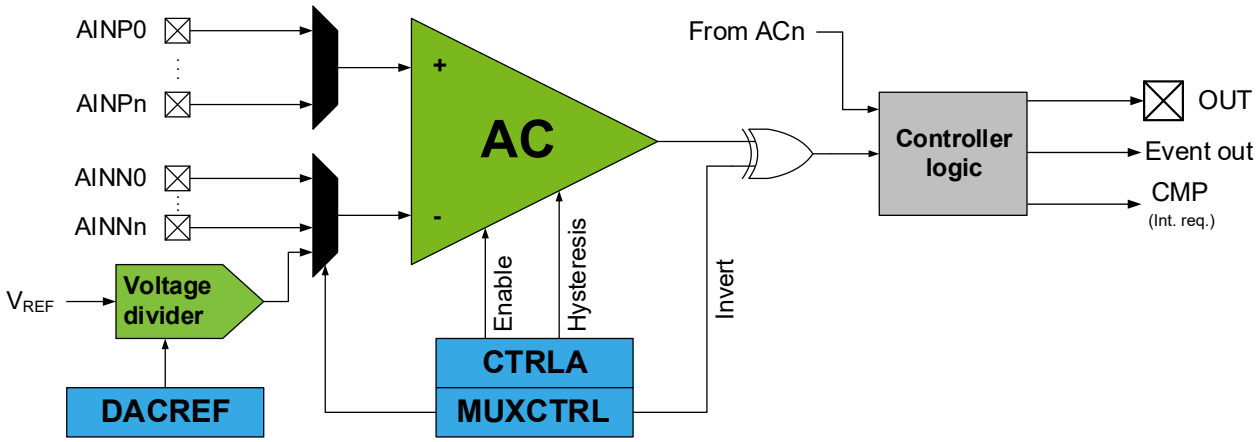
The input selection includes analog port pins and internally generated inputs. The AC digital output goes through controller logic, enabling customization of the signal for use internally with the Event System or externally on the pin.

The dynamic behavior of the AC can be adjusted by a hysteresis feature. The hysteresis can be customized to optimize the operation for each application.

The individual comparators can be used independently (Normal mode) or paired to form a window comparison (Window mode).

### 32.2.1 Block Diagram

**Figure 32-1. Analog Comparator**




### 32.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| AINNn | Negative input n | Analog |
| AINPn | Positive input n | Analog |
| OUT | Comparator output of AC | Digital |

## 32.3 Functional Description

### 32.3.1 Initialization

For basic operation, follow these steps:

1. Configure the desired input pins in the port peripheral as analog inputs.
2. Select the positive and negative input sources by writing to the Positive and Negative Input MUX Selection (MUXPOS and MUXNEG) bit fields in the MUX Control (ACn.MUXCTRL) register.
3. Optional: Enable the output to pin by writing a '1' to the Output Pad Enable (OUTEN) bit in the Control A (ACn.CTRLA) register.
4. Enable the AC by writing a '1' to the ENABLE bit in ACn.CTRLA.

During the start-up time after enabling the AC, the INITVAL bit in the CTRLB register can be used to set the AC output before the AC is ready. If $V_{REF}$ is used as a reference source, the respective start-up time of the reference source must be added. For details about the start-up time of the AC and VREF peripherals, refer to the *Electrical Characteristics* section.

To avoid the pin being tri-stated when the AC is disabled, the OUT pin must be configured as output.

### 32.3.2 Operation

#### 32.3.2.1 Input Hysteresis

Applying an input hysteresis helps to prevent constant toggling of the output when the noise-afflicted input signals are close to each other.

The input hysteresis can either be disabled or have one of three levels. The hysteresis is configured by writing to the Hysteresis Mode Select (HYSMODE) bit field in the Control A (ACn.CTRLA) register. For details about typical values of hysteresis levels, refer to the *Electrical Characteristics* section.

#### 32.3.2.2 Input and Reference Selection

The input selection to the ACn is controlled by the Positive and Negative Multiplexers (MUXPOS and MUXNEG) bit fields in the MUX Control (ACn.MUXCTRL) register. For positive input of ACn, an analog pin can be selected, while for negative input, the selection can be made between analog pins and internal DAC reference voltage (DACREF). For details about the possible selections, refer to the MUX Control (ACn.MUXCTRL) register description.

The generated voltage depends on the DACREF register value and the reference voltage selected in the VREF module, and is calculated as:

$$V_{\mathrm{DACREF}} = \frac{\mathrm{DACREF}}{256} \times V_{\mathrm{REF}}$$

The internal reference voltages ($V_{REF}$), except for $V_{REFA}$ and $V_{DD}$, are generated from an internal band gap reference.

After switching inputs to I/O pins or setting a new voltage reference, the ACn requires time to settle. Refer to the *Electrical Characteristics* section for more details.

#### 32.3.2.3 Normal Mode

The AC has one positive input and one negative input. The output of the comparator is '1' when the difference between the positive and the negative input voltage is positive, and '0' otherwise. This output is available on the output pin (OUT) through a logic XOR gate. This allows the inversion of the OUT pin when the INVERT bit in the MUX Control (ACn.MUXCTRL) register is '1'.

To avoid random output and set a specific level on the OUT pin during the ACn initialization, the INITVAL bit in the same register is used.

#### 32.3.2.4 Power Modes

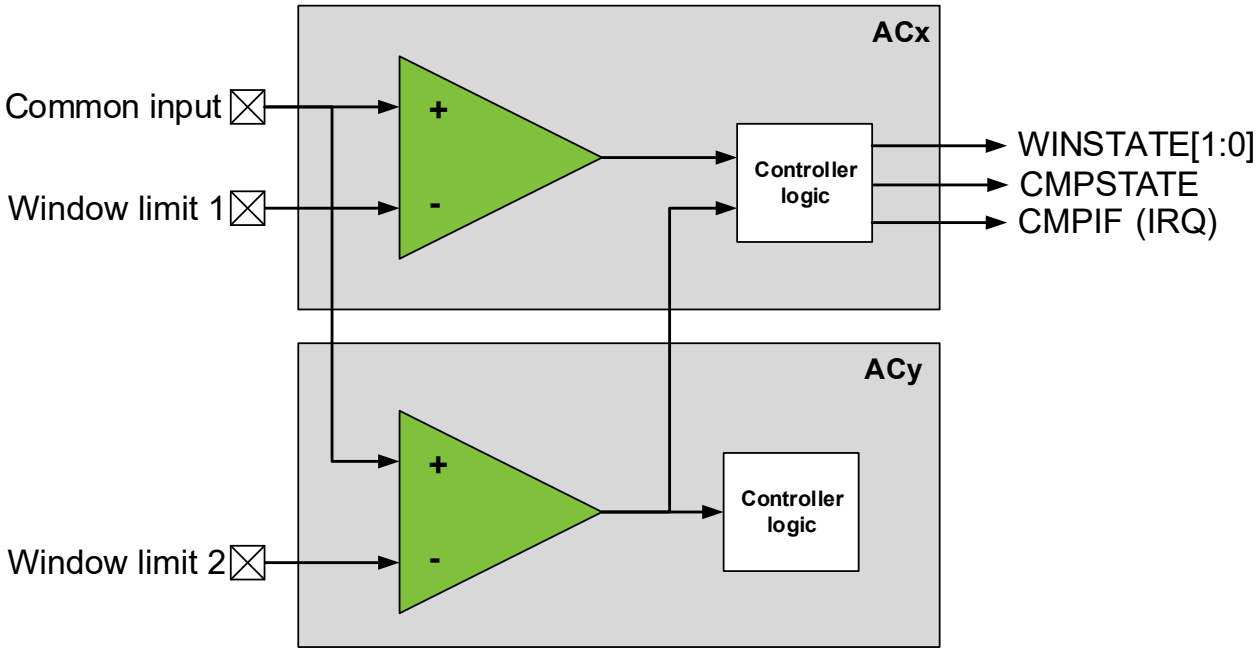
For power sensitive applications, the AC provides multiple power modes with balance power consumption and response time. A mode is selected by writing to the Power Profile (POWER) bit field in the Control A (ACn.CTRLA) register.

#### 32.3.2.5 Window Mode

Each AC (i.e., ACx) can be configured to work together with another comparator (i.e., ACy) in Window mode. In this mode, a voltage range (the window) is defined, and the selected comparator indicates whether an input signal is within this range or not.

The WINSEL bit field in the Control B (ACn.CTRLB) register selects which ACy instance is connected to the current comparator (ACx) to create the window comparator. The user is responsible for configuring the MUXPOS and MUXNEG bit fields in the MUX Control (ACn.MUXCTRL) register for ACx and ACy, so they match the setup in the figure below. Note that the MUXPOS bit field in the ACn.MUXCTRL register of both ACs must be configured to the same pin.

**Figure 32-2. Analog Comparators in Window Mode**




The status of the input signal is reported by the Window State (WINSTATE) flags in the Status (ACn.STATUS) register. The status can be:

- Above window - the input signal is above the upper limit.
- Inside window - the input signal is between the lower and upper limit.
- Below window - the input signal is below the lower limit.

Writing to the INTMODE bit field in the Interrupt Control (INTCTRL) register selects one of these window modes for triggering an event or requesting an interrupt:

- Above window - the interrupt/event is issued when the input signal is above the upper limit.
- Inside window - the interrupt/event is issued when the input signal is between the lower and upper limit.
- Below window - the interrupt/event is issued when the input signal is below the lower limit.
- Outside window - the interrupt/event is issued when the input signal is not between the lower and upper limit.

The CMPSTATE bit is '`1`' when the Window state matches the selected Interrupt Mode (INTMODE) bit field, and '`0`' otherwise.

The window interrupt is enabled by writing a '`1`' to the Analog Comparator Interrupt Enable (CMP) bit in the Interrupt Control (ACn.INTCTRL) register.

### 32.3.3 Events

The AC can generate the following events:

**Table 32-1. Event Generators in AC**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Module | Event | | | | |
| ACn | OUT | Comparator output level | Level | Asynchronous | Given by AC output level |

The AC has no event users.

Refer to the *Event System* (EVSYS) section for more details regarding event types and Event System configuration.

### 32.3.4 Interrupts

**Table 32-2. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| CMP | Analog comparator interrupt | AC output is toggling as configured by INTMODE in ACn.INTCTRL |

When an interrupt condition occurs, the corresponding interrupt flag is set in the Status (ACn.STATUS) register.

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control (ACn.INTCTRL) register.

The AC can generate a comparator interrupt, CMP, and can request this interrupt on either rising, falling, or both edges of the toggling comparator output. This is configured by writing to the Interrupt Mode (INTMODE) bit field in the Interrupt Control (ACn.INTCTRL) register. The interrupt is enabled by writing a '`1`' to the Analog Comparator Interrupt Enable (CMP) bit in the Interrupt Control (ACn.INTCTRL) register. The interrupt request remains active until the interrupt flag is cleared. Refer to the Status (ACn.STATUS) register description for details on how to clear the interrupt flags.

### 32.3.5 Sleep Mode Operation

In Idle Sleep mode the AC will continue to operate as normal.

In Standby Sleep mode the AC is disabled by default. If the Run in Standby Mode (RUNSTDBY) bit in the Control A (ACn.CTRLA) register is written to '`1`', the AC will continue to operate as normal with an event, interrupt and AC output on the pin even if the CLK_PER is not running in Standby Sleep mode.

In Power-Down Sleep mode the AC and the output to the pad are disabled.

## 32.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | RUNSTDBY | OUTEN | | POWER[1:0] | | HYSMODE[1:0] | | ENABLE |
| 0x01 | CTRLB | 7:0 | | | | | | | WINSEL[1:0] | |
| 0x02 | MUXCTRL | 7:0 | INVERT | INITVAL | MUXPOS[2:0] | | | MUXNEG[2:0] | | |
| 0x03<br>...<br>0x04 | Reserved | | | | | | | | | |
| 0x05 | DACREF | 7:0 | DACREF[7:0] | | | | | | | |
| 0x06 | INTCTRL | 7:0 | | | INTMODE[1:0] | | | | | CMP |
| 0x07 | STATUS | 7:0 | WINSTATE[1:0] | | | CMPSTATE | | | | CMPIF |

## 32.5 Register Description

### 32.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | OUTEN | | POWER[1:0] | | HYSMODE[1:0] | | ENABLE |
| 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 – RUNSTDBY** Run in Standby Mode
Writing this bit to '1' allows the AC to continue operation in Standby Sleep mode. Since the clock is stopped, interrupts and status flags are not updated.

| Value | Description |
|---|---|
| 0 | In Standby Sleep mode, the peripheral is halted |
| 1 | In Standby Sleep mode, the peripheral continues operation |

**Bit 6 – OUTEN** Output Pad Enable
Writing this bit to '1' makes the OUT signal available on the pin.

**Bits 4:3 – POWER[1:0]** Power Profile
This setting controls the current through the comparator, which allows the AC to trade power consumption for the response time. Refer to the *Electrical Characteristics* section for power consumption and response time.

| Value | Name | Description |
|---|---|---|
| 0x0 | PROFILE0 | Power profile 0. Shortest response time and highest consumption. |
| 0x1 | PROFILE1 | Power profile 1 |
| 0x2 | PROFILE2 | Power profile 2 |
| 0x3 | - | Reserved |

**Bits 2:1 – HYSMODE[1:0]** Hysteresis Mode Select
Writing to this bit field selects the Hysteresis mode for the AC input. For details about typical values of hysteresis levels, refer to the *Electrical Characteristics* section.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No hysteresis |
| 0x1 | SMALL | Small hysteresis |
| 0x2 | MEDIUM | Medium hysteresis |
| 0x3 | LARGE | Large hysteresis |

**Bit 0 – ENABLE** Enable AC
Writing this bit to '1' enables the AC.

### 32.5.2 Control B

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

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

**Bits 1:0 – WINSEL[1:0]** Window Selection Mode
This bit field selects the AC connected to the current comparator in Window mode.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLED | Window function disabled |
| 0x1 | UPSEL1 | Windows enabled, with ACn+1 connected |
| 0x2 | UPSEL2 | Windows enabled, with ACn+2 connected |
| 0x3 | - | Reserved |

### 32.5.3 MUX Control

**Name:** MUXCTRL
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INVERT | INITVAL | MUXPOS[2:0] | | | MUXNEG[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 – INVERT** Invert AC Output
Writing this bit to '1' enables inversion of the output of the AC. This inversion has to be taken into account when using the AC output signal as an input signal to other peripherals or parts of the system.

**Bit 6 – INITVAL** AC Output Initial Value
To avoid that the AC output toggles before the comparator is ready, the INITVAL can be used to set the initial state of the comparator output.

| Value | Name | Description |
|---|---|---|
| 0x0 | LOW | Output initialized to '0' |
| 0x1 | HIGH | Output initialized to '1' |

**Bits 5:3 – MUXPOS[2:0]** Positive Input MUX Selection
Writing to this bit field selects the input signal to the positive input of the AC.

| Value | Name | Description |
|---|---|---|
| 0x0 | AINP0 | Positive pin 0 |
| 0x1 | AINP1 | Positive pin 1 |
| 0x2 | AINP2 | Positive pin 2 |
| 0x3 | AINP3 | Positive pin 3 |
| Other | - | Reserved |

**Bits 2:0 – MUXNEG[2:0]** Negative Input MUX Selection
Writing to this bit field selects the input signal to the negative input of the AC.

| Value | Name | Description |
|---|---|---|
| 0x0 | AINN0 | Negative pin 0 |
| 0x1 | AINN1 | Negative pin 1 |
| 0x2 | AINN2 | Negative pin 2 |
| 0x3 | DACREF | DAC Reference |
| Other | - | Reserved |

### 32.5.4 DAC Voltage Reference

**Name:** DACREF
**Offset:** 0x05
**Reset:** 0xFF
**Property:** R/W

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

**Bits 7:0 – DACREF[7:0]** DACREF Data Value
This bit field defines the output voltage from the internal voltage divider. The DAC voltage reference depends on the DACREF value and the reference voltage selected in the VREF module, and is calculated as:

$$V_{\mathrm{DACREF}} = \frac{\mathrm{DACREF}[7:0]}{256} \times V_{\mathrm{REF}}$$

### 32.5.5 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x06
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | INTMODE[1:0] | | | | | CMP |
| Access | | | R/W | R/W | | | | R/W |
| Reset | | | 0 | 0 | | | | 0 |

**Bits 5:4 – INTMODE[1:0]** Interrupt Mode
Writing to this bit field selects which edge(s) of the AC output or when entering a window state triggers an interrupt request.

**Table 32-3. Interrupt Generation in Window Mode**

| Value | Name | Description |
|---|---|---|
| 0x0 | ABOVE | Enables Window mode above interrupt |
| 0x1 | INSIDE | Enables Window mode inside interrupt |
| 0x2 | BELOW | Enables Window mode below interrupt |
| 0x3 | OUTSIDE | Enables Window mode outside interrupt |

**Table 32-4. Interrupt Generation with Single Comparator**

| Value | Name | Description |
|---|---|---|
| 0x0 | BOTHEDGE | Positive and negative inputs crosses |
| 0x1 | - | Reserved |
| 0x2 | NEGEDGE | Positive input goes above negative input |
| 0x3 | POSEDGE | Positive input goes below negative input |

**Bit 0 – CMP** AC Interrupt Enable
This bit enables the AC interrupt. The enabled interrupt will be triggered when the CMPIF bit in the ACn.STATUS register is set.

### 32.5.6 Status

**Name:** STATUS
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WINSTATE[1:0] | | | CMPSTATE | | | | CMPIF |
| Access | R | R | | R | | | | R/W |
| Reset | 0 | 0 | | 0 | | | | 0 |

**Bits 7:6 – WINSTATE[1:0]** Window State
When the window function is enabled, these flags indicate the current status of the input signal with respect to the window.
Not valid when the Window mode is disabled.

**Table 32-5. Window State Settings**

| Value | Name | Description |
|---|---|---|
| 0x0 | ABOVE | Above window |
| 0x1 | INSIDE | Inside window |
| 0x2 | BELOW | Below window |
| Other | - | Reserved |

**Bit 4 – CMPSTATE** AC State
If this bit is '1', the OUT signal is high. If this bit is '0', the OUT signal is low. In Window mode, if this bit is '1', the Window state matches the selected Interrupt mode (INTMODE) bit field. If INTMODE is 'OUTSIDE', both 'ABOVE' and 'BELOW' are valid matches. It will have a synchronizer delay to get updated in the I/O register (three cycles).

**Bit 0 – CMPIF** AC Interrupt Flag
This bit is '1' when the OUT signal matches the Interrupt Mode (INTMODE) bit field as defined in the ACn.INTCTRL register. Writing a '1' to this flag bit location will clear the flag.

# 33. ADC - Analog-to-Digital Converter

## 33.1 Features

- 12-Bit Resolution
- Up to 130 ksps at 12-Bit Resolution
- Differential and Single-Ended Conversion
- Up to 22 Inputs
- Rail-to-Rail Input Voltage Range
- Free-Running and Single Conversion
- Accumulation of Up to 128 Samples per Conversion
- Multiple Voltage Reference Options
- Temperature Sensor Input Channel
- Programmable Input Sampling Duration
- Configurable Threshold and Window Comparator
- Event Triggered Conversion
- Interrupt and Event on Conversion Complete

## 33.2 Overview

The Analog-to-Digital Converter (ADC) is a 12-bit Successive Approximation Register (SAR) ADC, with a sampling rate up to 130 ksps at 12-bit resolution. The ADC is connected to an analog input multiplexer for selection between multiple single-ended or differential inputs. In single-ended conversions, the ADC measures the voltage between the selected input and 0V (GND). In differential conversions, the ADC measures the voltage between two selected input channels. The selected ADC input channels can either be internal (e.g., a voltage reference) or external analog input pins.

An ADC conversion can be started by software, or by using the Event System (EVSYS) to route an event from other peripherals. This makes it possible to do a periodic sampling of input signals, trigger an ADC conversion on a special condition or trigger an ADC conversion in Standby sleep mode.

A digital window compare feature is available for monitoring the input signal and can be configured only to trigger an interrupt if the sample is below or above a user-defined threshold, or inside or outside a user-defined window, with minimum software intervention required.

The ADC input signal is fed through a sample-and-hold circuit which ensures that the input voltage to the ADC is held at a constant level during sampling.

The ADC supports sampling in bursts where a configurable number of conversions are accumulated into a single ADC result (Sample Accumulation). Furthermore, a sample delay can be configured to tune the ADC burst sampling frequency away from any harmonic noise aliased from the sampled signal.

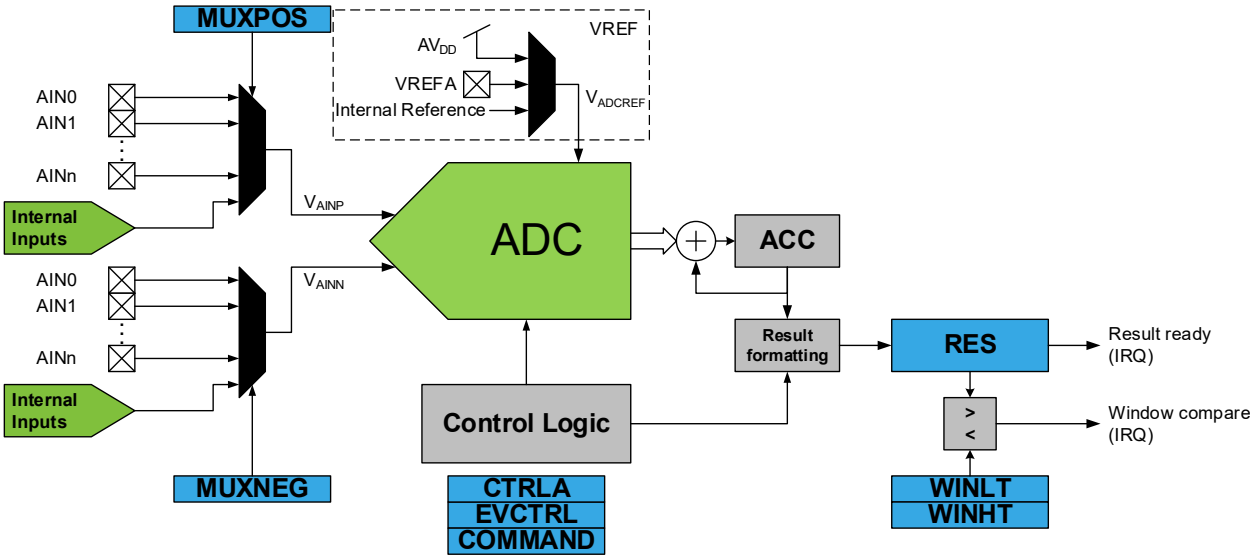
The ADC voltage reference is configured in the Voltage Reference (VREF) peripheral and can use one of the following sources as voltage reference:

- Multiple Internally Generated Voltages
- $AV_{DD}$ Supply Voltage
- External VREF Pin (VREFA)

This device has one instance of the ADC peripheral: ADC0.

### 33.2.1 Block Diagram

**Figure 33-1. Block Diagram**




### 33.2.2 Signal Description

| Pin Name | Type | Description |
|---|---|---|
| AIN[n:0] | Analog input | Analog input to be converted |
| VREFA | Analog input | External voltage reference pin |

## 33.3 Functional Description

### 33.3.1 Definitions

- Conversion: The operation in which analog values on the selected ADC inputs are transformed into a digital representation.
- Sample: The output of a single ADC conversion.
- Result: The value placed in the Result (ADCn.RES) register. Depending on the ADC configuration, this value is a single sample or the sum of multiple accumulated samples.

### 33.3.2 Initialization

The following steps are recommended to initialize ADC operation:

1. Configure the ADC voltage reference in the Voltage Reference (VREF) peripheral.
2. Optional: Select between Single-Ended or Differential mode by writing to the Conversion Mode (CONVMODE) bit in the Control A (ADCn.CTRLA) register.
3. Configure the resolution by writing to the Resolution Selection (RESSEL) bit field in the ADCn.CTRLA register.
4. Optional: Configure to left adjust by writing a '1' to the Left Adjust Result (LEFTADJ) bit in the ADCn.CTRLA register.
5. Optional: Select the Free-Running mode by writing a '1' to the Free-Running (FREERUN) bit in the ADCn.CTRLA register.
6. Optional: Configure the number of samples to be accumulated per conversion by writing to the Sample Accumulation Number Select (SAMPNUM) bit field in the Control B (ADCn.CTRLB) register.
7. Configure the ADC clock (CLK_ADC) by writing to the Prescaler (PRESC) bit field in the Control C (ADCn.CTRLC) register.

8. Select the positive ADC input by writing to the MUXPOS bit field in the ADCn.MUXPOS register.
9. Optional: Select the negative ADC input by writing to the MUXNEG bit field in the ADCn.MUXNEG register.
10. Optional: Enable Start Event input by writing a '1' to the Start Event Input (STARTEI) bit in the Event Control (ADCn.EVCTRL) register, and configure the Event System accordingly.
11. Enable the ADC by writing a '1' to the ADC Enable (ENABLE) bit in the ADCn.CTRLA register.

Following these steps will initialize the ADC for basic measurements.

For details about the start-up time of the VREF peripheral, refer to the *Electrical Characteristics* section.

The ADC does not consume power when the ENABLE bit is '0'. The ADC generates a 10- or 12-bit result which can be read from the Result (ADCn.RES) register.

**Notes:** Changing the following registers during a conversion will give unpredictable results:

- In ADCn.CTRLA:
  - Conversion Mode (CONVMODE) bit
  - Left Adjust Result (LEFTADJ) bit
  - Resolution Selection (RESSEL) bit field
- In ADCn.CTRLB:
  - Sample Accumulation Number Select (SAMPNUM) bit field
- In ADCn.CTRLC:
  - Prescaler (PRESC) bit field

### 33.3.3 Operation

#### 33.3.3.1 Operation Modes

The ADC supports differential and single-ended conversions. This is configured in the CONVMODE bit in the ADCn.CTRLA register.

The operation modes can be split into two groups:

- Single conversion of one sample per trigger
- Accumulated conversion of n conventions per trigger, the result is accumulated

The accumulated conversion utilizes 12-bit conversions and can be configured with or without truncation of the accumulated result. The accumulator is always reset to zero when a new accumulated conversion is started.

#### 33.3.3.2 Starting a Conversion

Once the initialization is finished, a conversion is started by writing a '1' to the ADC Start Conversion (STCONV) bit in the Command (ADCn.COMMAND) register. This bit is '1' as long as the conversion is in progress. The STCONV bit will be set during a conversion and cleared once the conversion is complete.

If a different input channel is selected while a conversion is in progress, the ADC will finish the current conversion before changing the channel.

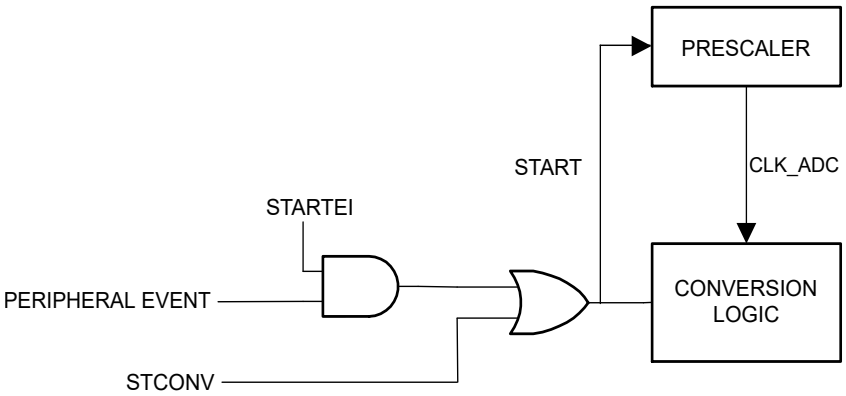
Depending on the accumulator setting, the conversion result is a single sample, or an accumulation of samples. Once the triggered operation is finished, the Result Ready (RESRDY) flag in the Interrupt Flags (ADCn.INTFLAGS) register is set. The corresponding interrupt vector is executed if the Result Ready Interrupt Enable (RESRDY) bit in the Interrupt Control (ADCn.INTCTRL) register is '1' and the Global Interrupt Enable bit is '1'.

The RESRDY interrupt flag in the ADCn.INTFLAGS register will be set even if the specific interrupt is disabled, allowing software to check for any finished conversion by polling the flag. A conversion can thus be triggered without causing an interrupt upon completion.

Alternatively, a conversion can be triggered by an event. This is enabled by writing a '1' to the Start Event Input (STARTEI) bit in the Event Control (ADCn.EVCTRL) register. Any incoming event routed to the ADC through the Event System (EVSYS) will trigger an ADC conversion. This provides a method to start conversions with predictable intervals or at specific conditions.

The ADC will trigger a conversion on the rising edge of an event signal. When an event occurs, the STCONV bit in the ADCn.COMMAND register is set and it will be cleared when the conversion is complete. Refer to Figure 33-2.

**Figure 33-2. ADC Event Trigger Logic**

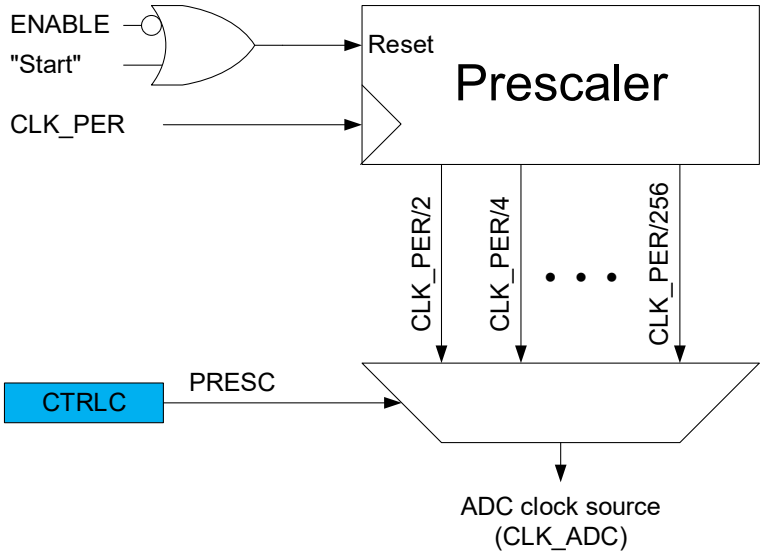



In Free-Running mode, the first conversion is started by writing a '1' to the STCONV bit in the ADCn.COMMAND register. A new conversion cycle is started immediately after the previous conversion cycle has completed. A completed conversion will set the RESRDY flag in the ADCn.INTFLAGS register.

### 33.3.3.3 Clock Generation

The ADC peripheral contains a prescaler which generates the ADC clock (CLK_ADC) from the peripheral clock (CLK_PER). The minimum ADC_CLK frequency is 125 kHz. The prescaling is selected by writing to the Prescaler (PRESC) bit field in the Control C (ADCn.CTRLC) register. The prescaler begins counting from the moment the ADC conversion starts and is reset for every new conversion. Refer to Figure 33-3.

**Figure 33-3. ADC Prescaler**




When initiating a conversion by writing a '1' to the Start Conversion (STCONV) bit in the ADCn.COMMAND register or from event, the conversion starts after one CLK_PER cycle. The prescaler is kept in Reset, as long as there is no ongoing conversion. This assures a fixed delay from the trigger to the actual start of a conversion of maximum 2 CLK_PER cycles.

### 33.3.3.4 Conversion Timing

A normal conversion takes place in the following order:

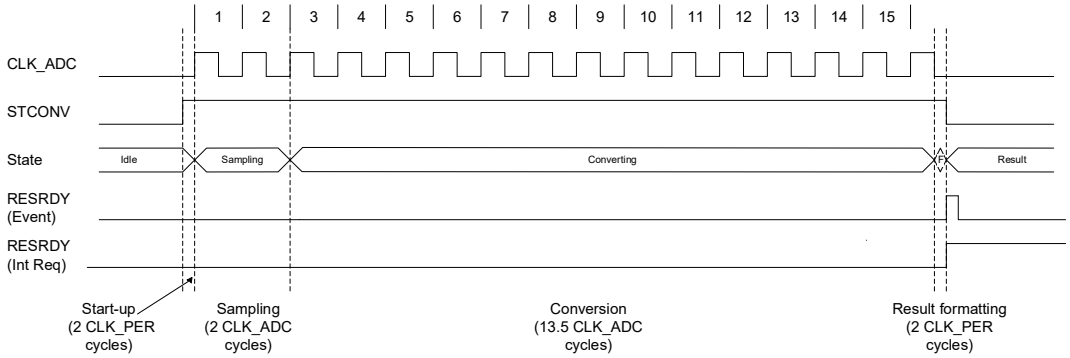
1. Write a '1' to the STCONV bit in the Command (ADCn.COMMAND) register.
2. Start-up for maximum 2 CLK_PER cycles.
3. Sample-and-hold for 2 CLK_ADC cycles.
4. Conversion for 13.5 CLK_ADC cycles.
5. Result formatting for 2 CLK_PER cycles.

When a conversion is complete, the result is available in the Result (ADCn.RES) register, and the Result Ready (RESRDY) interrupt flag is set in the Interrupt Flags (ADCn.INTFLAGS) register.

#### 33.3.3.4.1 Single Conversion

The figure below shows the timing diagram for a single 12-bit ADC conversion.

**Figure 33-4. Timing Diagram - Single Conversion**



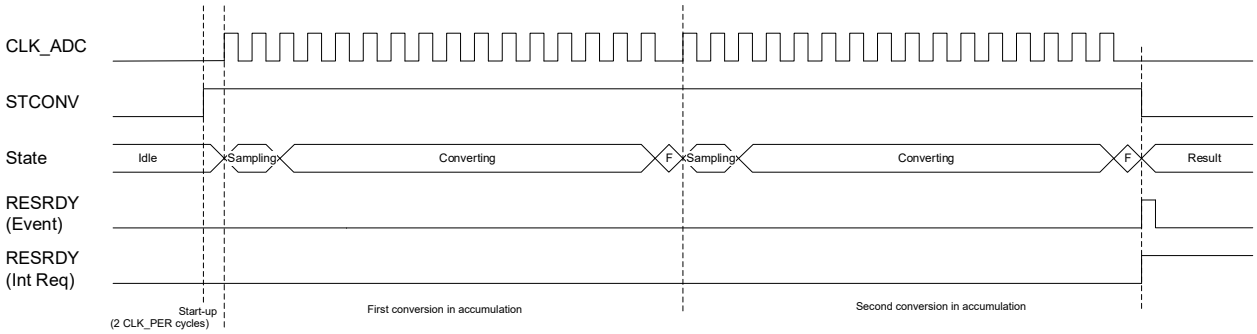

For a single conversion, the total conversion time is calculated by:

$$\text{Total Conversion Time (12-bit)} = \frac{13.5 + 2}{f_{\text{CLK_ADC}}} + \frac{4}{f_{\text{CLK_PER}}}$$

#### 33.3.3.4.2 Accumulated Conversion

The figure below shows the timing diagram for the ADC when accumulating two samples in Accumulation mode.

**Figure 33-5. Timing Diagram - Accumulated Conversion**




The number of samples to accumulate is configured with the Sample Number (SAMPNUM) bit field in the Control B (ADCn.CTRLB) register. The STCONV bit is set for the entire conversion. The total conversion time for n samples is given by:

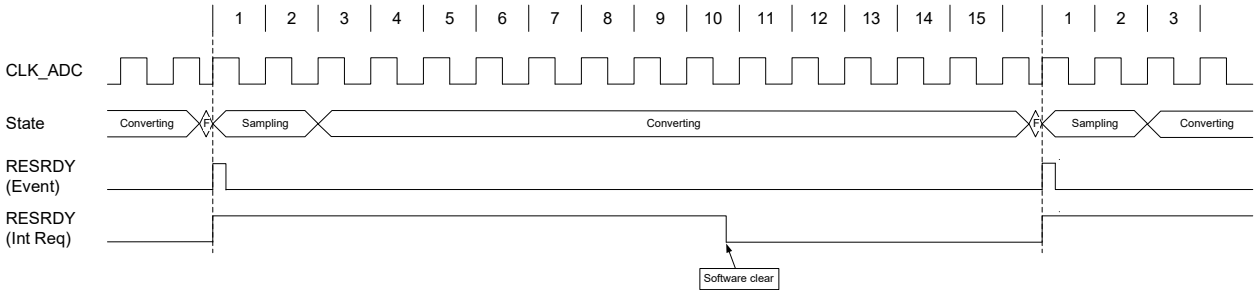
$$\text{Total Conversion Time (12-bit)} = \frac{2}{f_{\text{CLK_PER}}} + n\left(\frac{13.5 + 2}{f_{\text{CLK_ADC}}} + \frac{2}{f_{\text{CLK_PER}}}\right)$$

#### 33.3.3.4.3 Free-Running Conversion

In Free-Running mode, a new conversion is started as soon as the previous conversion has completed. This is signaled by the RESRDY bit in the Interrupt Flags (ADCn.INTFLAGS) register.

The figure below shows the timing diagram for the ADC in Free-Running mode with single conversion.

**Figure 33-6. Timing Diagram - Free-Running Conversion**




The Result Ready event and the interrupt flag are set after each conversion. It is possible to combine accumulated conversion and Free-Running mode.

To safely change any of these settings when using Free-Running mode, disable Free-Running mode, and wait for the conversion to complete before doing any changes. Enable Free-Running mode again before starting the next conversion.

#### 33.3.3.4.4 Adjusting Conversion Time

Both sampling time and sampling length can be adjusted using the Sampling Delay Selection (SAMPDLY) bit field in the Control D (ADCn.CTRLD) register and Sample Length (SAMPLEN) bit field in the Sample Control (ADCn.SAMPCTRL) register. Both of these control the ADC sampling time and sampling length in a number of CLK_ADC cycles. Increasing SAMPLEN allows sampling high-impedance sources without reducing CLK_ADC frequency. Adjusting SAMPDLY is intended for tuning the sampling frequency away from harmonic noise in the analog signal. Total sampling time is given by:

$$\text{SampleTime} = \frac{(2 + \text{SAMPDLY} + \text{SAMPLEN})}{f_{\text{CLK_ADC}}}$$

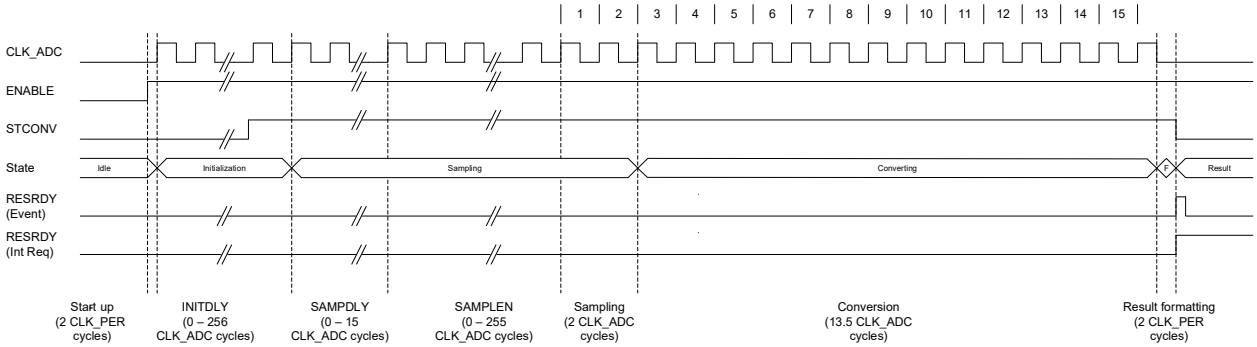
The equation above implies that the total conversion time for n samples is now:

$$\text{Total Conversion Time (12-bit)} = \frac{2}{f_{\text{CLK_PER}}} + n\left(\frac{13.5 + 2 + SAMPDLY + SAMPLEN}{f_{\text{CLK_ADC}}} + \frac{2}{f_{\text{CLK_PER}}}\right)$$

Some of the analog resources used by the ADC require time to initialize before a conversion can start. The Initialization Delay (INITDLY) bit field in the Control D (ADCn.CTRLD) register can be used to prevent starting a conversion prematurely by halting sampling for the configured delay duration.

The figure below shows the timing diagram for the ADC and the usage of the INITDLY, SAMPDLY and SAMPLEN bit fields:

**Figure 33-7. Timing Diagram - Conversion with Delays and Custom Sampling Length**




### 33.3.3.5 Conversion Result (Output Formats)

The result of an analog-to-digital conversion is written to the 16-bit Result (ADCn.RES) register and is given by the following equations:

Single-ended 12-bit conversion: $RES = \frac{V_{\text{AINP}}}{V_{\text{ADCREF}}} \times 4096 \in [0, 4095]$

Single-ended 10-bit conversion: $RES = \frac{V_{\mathrm{AINP}}}{V_{\mathrm{ADCREF}}} \times 1024 \in [0, 1023]$

Differential 12-bit conversion: $RES = \frac{V_{\mathrm{AINP}} - V_{\mathrm{AINN}}}{V_{\mathrm{ADCREF}}} \times 2048 \in [-2048, 2047]$

Differential 10-bit conversion: $RES = \frac{V_{\mathrm{AINP}} - V_{\mathrm{AINN}}}{V_{\mathrm{ADCREF}}} \times 512 \in [-512, 511]$

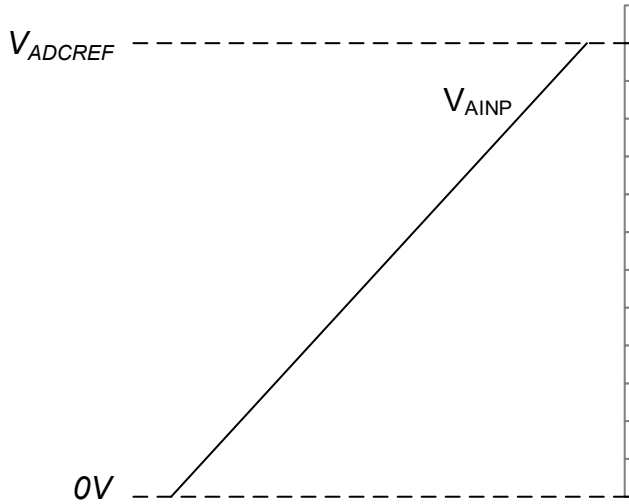
where $V_{AINP}$ and $V_{AINN}$ are the positive and negative ADC inputs and $V_{ADCREF}$ is the selected ADC voltage reference.

The data format used for single-ended conversions is unsigned one's complement, while two's complement with sign extension is used for differential conversions. Consequently, for differential conversions the sign bit is padded to the higher bits in the Result register, if needed.

By default, conversion results are stored in the Result register as right-adjusted 16-bit values. The eight Least Significant bits (LSbs) are then located in the low byte of the Result register. By writing a '`1`' to the Left Adjust Result (LEFTADJ) bit in the Control A (ADCn.CTRLA) register, the values will be left-adjusted by placing the eight Most Significant bits (MSbs) in the high byte of the Result register.

The two figures below illustrate the relationship between the analog input and the corresponding ADC output.

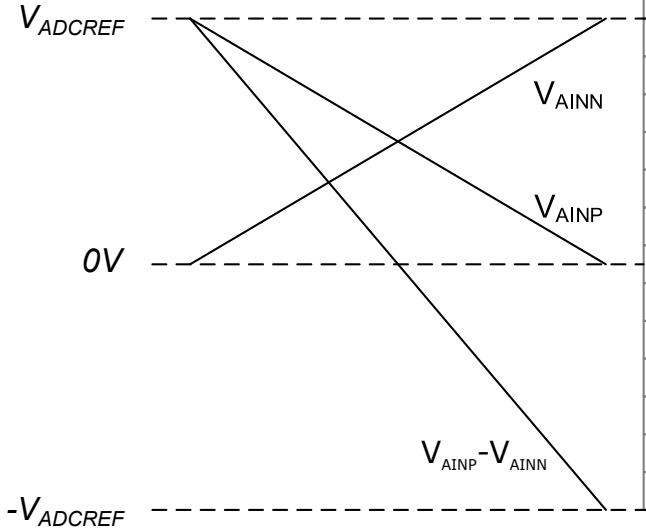
**Figure 33-8. Unsigned Single-Ended, Input Range, and Result Representation**




| Dec | Hex | 16-bit result |
|---|---|---|
| 4095 | FFF | 0x0FFF |
| 4094 | FFE | 0x0FFE |
| 4093 | FFD | 0x0FFD |
| ... | ... | ... |
| 2049 | 801 | 0x0801 |
| 2048 | 800 | 0x0800 |
| 2047 | 7FF | 0x07FF |
| 2046 | 7FE | 0x07FE |
| ... | ... | ... |
| 2 | 2 | 0x0002 |
| 1 | 1 | 0x0001 |
| 0 | 0 | 0x0000 |

Where $V_{AINP}$ is the single-ended or internal input.

**Figure 33-9. Signed Differential Input, Input Range, and Result Representation**




| Dec | Hex | 16-bit result |
|---|---|---|
| 2047 | 7FF | 0x07FF |
| 2046 | 7FE | 0x07FE |
| 2045 | 7FD | 0x07FD |
| ... | ... | ... |
| 2 | 2 | 0x0002 |
| 1 | 1 | 0x0001 |
| 0 | 0 | 0x0000 |
| -1 | FFF | 0xFFFF |
| -2 | FFE | 0xFFFE |
| ... | ... | ... |
| -2046 | 802 | 0xF802 |
| -2047 | 801 | 0xF801 |
| -2048 | 800 | 0xF800 |

If a single-ended analog input is above the ADC voltage reference level, the 12-bit ADC result will be `0xFFF` (decimal 4095). Likewise, if the input is below 0V, the ADC result will be `0x000`.

If the voltage difference between $V_{AINP}$ and $V_{AINN}$ for a 12-bit differential conversion is above the ADC voltage reference level, the ADC result will be `0x7FF` (decimal 2047). If the voltage difference is larger than the voltage reference level in the negative direction, the ADC result will be `0x800` (decimal -2048).

### 33.3.3.6 Accumulation

By default, conversion results are stored in the Result register as right-adjusted 16-bit values. The eight Least Significant bits (LSbs) are then located in the low byte of the Result register. By writing a '`1`' to the Left Adjust Result (LEFTADJ) bit in the Control A (ADCn.CTRLA) register, the values will be left-adjusted by placing the eight Most Significant bits (MSbs) in the high byte of the Result register.

The result from multiple consecutive conversions can be accumulated. The number of samples to be accumulated is specified by the Sample Accumulation Number Select (SAMPNUM) bit field in the Control B (ADCn.CTRLB) register. When accumulating more than 16 samples, the result might be too large to match the 16-bit Result register size. To avoid overflow, the LSbs of the result are truncated to fit within the available register size.

The two following tables show how the Result (ADCn.RES) register value is stored for single-ended and differential conversions.

**Table 33-1. Result Format in Single-Ended Mode**

| Accumulations | LEFTADJ | RES[15:8] | | | | | | | | | RES[7:0] | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | Conversion [11:0] | | | | | | | | | | | |
| | 1 | Conversion [11:0] | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | Accumulation [12:0] | | | | | | | | | | | | |
| | 1 | Accumulation [12:0] | | | | | | | | | | | | | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | Accumulation [13:0] | | | | | | | | | | | | | |
| | 1 | Accumulation [13:0] | | | | | | | | | | | | | | 0 | 0 |
| 8 | 0 | 0 | Accumulation [14:0] | | | | | | | | | | | | | | |
| | 1 | Accumulation [14:0] | | | | | | | | | | | | | | | 0 |

.....continued

| Accumulations | LEFTADJ | RES[15:8] | | | | | | | | | RES[7:0] | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 16 | 0 | Accumulation [15:0] | | | | | | | | | | | | | | | |
| | 1 | | | | | | | | | | | | | | | | |
| 32, 64, 128 | 0 | Truncated Accumulation [15:0] | | | | | | | | | | | | | | | |
| | 1 | | | | | | | | | | | | | | | | |

**Table 33-2. Result Format in Differential Mode**

| Accumulations | LEFTADJ | RES[15:8] | | | | | | | | | RES[7:0] | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 1 | 0 | Sign extension | | | | Signed conversion [11:0] | | | | | | | | | | | |
| | 1 | Signed conversion [11:0] | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| 2 | 0 | Sign extension | | | Signed accumulation [12:0] | | | | | | | | | | | | |
| | 1 | Signed accumulation [12:0] | | | | | | | | | | | | | 0 | 0 | 0 |
| 4 | 0 | Sign extension | | | | Signed accumulation [13:0] | | | | | | | | | | | |
| | 1 | Signed accumulation [13:0] | | | | | | | | | | | | | | 0 | 0 |
| 8 | 0 | Sign extension | | Signed accumulation [14:0] | | | | | | | | | | | | | |
| | 1 | Signed accumulation [14:0] | | | | | | | | | | | | | | | 0 |
| 16 | 0 | Signed accumulation [15:0] | | | | | | | | | | | | | | | |
| | 1 | | | | | | | | | | | | | | | | |
| 32, 64, 128 | 0 | Signed truncated accumulation [15:0] | | | | | | | | | | | | | | | |
| | 1 | | | | | | | | | | | | | | | | |

### 33.3.3.7 Channel Selection

The input selection for the ADC is controlled by the MUXPOS and MUXNEG bit fields in the ADCn.MUXPOS and ADCn.MUXNEG registers, respectively. If the ADC is running single-ended conversions, only MUXPOS is used, while both are used in differential conversions.

The MUXPOS bit field of the ADCn.MUXPOS register and the MUXNEG bit field of the ADCn.MUXNEG register are buffered through a temporary register. This ensures that the input selection only comes into effect at a safe point during the conversion. The channel selections are continuously updated until a conversion is started.

Once the conversion starts, the channel selections are locked to ensure sufficient sampling time for the ADC. The continuous updating of input channel selection resumes in the last CLK_ADC clock cycle before the conversion completes. The next conversion starts on the following rising CLK_ADC clock edge after the STCONV bit is written to '1'.

### 33.3.3.8 Temperature Measurement

An on-chip temperature sensor is available. Follow the steps below to do a temperature measurement. The resulting value will be right-adjusted.

1. In the Voltage Reference (VREF) peripheral, select the internal 2.048V reference as the ADC reference voltage.
2. Select the temperature sensor as input in the ADCn.MUXPOS register.

3. Acquire the temperature sensor output voltage by running a 12-bit, right-adjusted, single-ended conversion.
4. Process the measurement result as described below.

The measured voltage has an almost linear relationship with the temperature. Due to process variations, the temperature sensor output voltage varies between individual devices at the same temperature. The individual compensation factors determined during production test are stored in the Signature Row. These compensations factors are generated for the internal 2.048V reference.

- SIGROW.TEMPSENSE0 contains the slope of the temperature sensor characteristics
- SIGROW.TEMPSENSE1 contains the offset of the temperature sensor characteristics

In order to achieve more accurate results, the result of the temperature sensor measurement must be processed in the application software using compensation values from device production or user calibration. The temperature (in Kelvin) is calculated by the following equation:

$$T = \frac{(\text{Offset} - \text{ADC Result}) \times \text{Slope}}{4096}$$

It is recommended to follow these steps in the application code when using the compensation values from the Signature Row:

```
uint16_t sigrow_offset = SIGROW.TEMPSENSE1; // Read unsigned value from signature row
uint16_t sigrow_slope = SIGROW.TEMPSENSE0;  // Read unsigned value from signature row
uint16_t adc_reading = ADCn.RES;            // ADC conversion result

uint32_t temp = sigrow_offset - adc_reading;
temp *= sigrow_slope;   // Result will overflow 16-bit variable
temp += 0x0800;         // Add 4096/2 to get correct rounding on division below
temp >>= 12;            // Round off to nearest degree in Kelvin, by dividing with 2^12 (4096)
uint16_t temperature_in_K = temp;
```

To increase the precision of the measurement to less than 1 Kelvin it is possible to adjust the last two steps to round off to a fraction of one degree. Add 4096/4 and right shift by 11 for a precision of ½ Kelvin, or add 4096/8 and right shift by 10 for a ¼ Kelvin precision.

If accumulation is used to reduce noise in the temperature measurement, the ADC result needs to be adjusted to a 12-bit value before the calculation is performed.

If another reference ($V_{ADCREF}$) than 2.048V is required, the offset and slope values need to be adjusted according to the following equations:

$$\text{Slope} = \text{TEMPSENSE0} \times \frac{V_{ADCREF}}{2.048V}$$

$$\text{Offset} = \text{TEMPSENSE1} \times \frac{2.048V}{V_{ADCREF}}$$

#### 33.3.3.9 Window Comparator

The ADC can raise the Window Comparator Interrupt (WCMP) flag in the Interrupt Flags (ADCn.INTFLAGS) register and request an interrupt (WCMP) when the output of a conversion or accumulation is above and/or below certain thresholds. The available modes are:

- The result is below a threshold
- The result is above a threshold
- The result is inside a window (above the lower threshold and below the upper threshold)
- The result is outside a window (either under the lower threshold or above the upper threshold)

The thresholds are defined by writing to the Window Comparator Low and High Threshold (ADCn.WINLT and ADCn.WINHT) registers. Writing to the Window Comparator Mode (WINCM) bit field in the Control E (ADCn.CTRLE) register selects the Window mode to use.

When accumulating multiple samples, the comparison between the result and the threshold will happen after the last sample was acquired. Consequently, the flag is raised only once, after taking the last sample of the accumulation.

Assuming the ADC is already configured to run, follow these steps to use the Window Comparator:

1. Set the required threshold(s) by writing to the Window Comparator Low and High Threshold (ADCn.WINLT and ADCn.WINHT) registers.
2. Optional: Enable the interrupt request by writing a '1' to the Window Comparator Interrupt Enable (WCMP) bit in the Interrupt Control (ADCn.INTCTRL) register.
3. Enable the Window Comparator and select a mode by writing a valid non-zero value to the Window Comparator Mode (WINCM) bit field in the Control E (ADCn.CTRLE) register.

When accumulating samples, the window comparator thresholds are applied to the accumulated value and not to each sample. Using left adjustment of the result will make the comparator values independent of number of samples.

### 33.3.4 I/O Lines and Connections

The analog input pins and the VREF pin (AINx and VREFA) are configured in the I/O Pin Controller (PORT).

To reduce power consumption, the digital input buffer has to be disabled on the pins used as inputs for ADC. This is configured by the I/O Pin Controller (PORT).

### 33.3.5 Events

The ADC can generate the following events:

**Table 33-3. Event Generators in ADC**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| ADCn | RESRDY | Result ready | Pulse | CLK_PER | One clock period |

The conditions for generating an event are identical to those that will raise the corresponding flag in the Interrupt Flags (ADCn.INTFLAGS) register.

The ADC has one event user for detecting and acting upon input events. The table below describes the event user and the associated functionality.

**Table 33-4. Event Users and Available Event Actions in ADC**

| User Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| ADCn | START | ADC start conversion | Edge | Async |

The ADC can be configured to start a conversion on the rising edge of an event signal by writing a '1' to the STARTEI bit field in the Event Control (ADCn.EVCTRL) register. Refer to the *Event System (EVSYS)* chapter for more details regarding event types and Event System configuration.

When an input event trigger occurs, the positive edge will be detected, the Start Conversion (STCONV) bit in the Command (ADCn.COMMAND) register will be set, and the conversion will start. When the conversion is completed, the Result Ready (RESRDY) flag in the Interrupt Flags (ADCn.INTFLAGS) register is set and the STCONV bit in ADCn.COMMAND is cleared.

### 33.3.6 Interrupts

**Table 33-5. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| RESRDY | Result Ready interrupt | The conversion result is available in ADCn.RES. |
| WCMP | Window Comparator interrupt | As defined by WINCM in ADCn.CTRLE. |

When an interrupt condition occurs, the corresponding interrupt flag is set in the Interrupt Flags (ADCn.INTFLAGS) register.

An interrupt source is enabled or disabled by writing to the corresponding enable bit in the Interrupt Control (ADCn.INTCTRL) register.

An interrupt request is generated when the corresponding interrupt source is enabled and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. Refer to the ADCn.INTFLAGS register for details on how to clear interrupt flags.

### 33.3.7 Debug Operation

By default, halting the CPU in Debugging mode will halt the normal operation of the peripheral.

This peripheral can be forced to operate while the CPU is halted by writing a '1' to the Debug Run (DBGRUN) bit in the Debug Control (ADCn.DBGCTRL) register.

### 33.3.8 Sleep Mode Operation

By default, the ADC is disabled in Standby sleep mode.

The ADC can stay fully operational in Standby Sleep mode if the Run in Standby (RUNSTDBY) bit in the Control A (ADCn.CTRLA) register is written to '1'.

In this case, the ADC will stay active, any ongoing conversions will be completed, and interrupts will be executed as configured.

In Standby seep mode, an ADC conversion can be triggered only via the Event System (EVSYS), or the ADC must be in Free-Running mode with the first conversion triggered by software before entering sleep. The peripheral clock is requested if needed and is turned off after the conversion is completed.

The reference source and supply infrastructure need time to stabilize when activated in Standby sleep mode. Configure a delay for the start of the first conversion by writing a non-zero value to the Initialization Delay (INITDLY) bit field in the Control D (ADCn.CTRLD) register.

In Power-Down sleep mode, no conversions are possible. Any ongoing conversions are halted and will be resumed when going out of sleep. At the end of the conversion, the Result Ready (RESRDY) flag will be set, but the content of the Result (ADCn.RES) registers will be invalid since the ADC was halted during a conversion. It is recommended to make sure conversions have completed before entering Power-Down sleep mode.

### 33.3.9 Synchronization

Not applicable.

### 33.3.10 Configuration Change Protection

Not applicable.

## 33.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | RUNSTDBY | | CONVMODE | LEFTADJ | RESSEL[1:0] | | FREERUN | ENABLE |
| 0x01 | CTRLB | 7:0 | | | | | | SAMPNUM[2:0] | | |
| 0x02 | CTRLC | 7:0 | | | | | PRESC[3:0] | | | |
| 0x03 | CTRLD | 7:0 | INITDLY[2:0] | | | | SAMPDLY[3:0] | | | |
| 0x04 | CTRLE | 7:0 | | | | | | WINCM[2:0] | | |
| 0x05 | SAMPCTRL | 7:0 | SAMPLEN[7:0] | | | | | | | |
| 0x06<br>...<br>0x07 | Reserved | | | | | | | | | |
| 0x08 | MUXPOS | 7:0 | | MUXPOS[6:0] | | | | | | |
| 0x09 | MUXNEG | 7:0 | | MUXNEG[6:0] | | | | | | |
| 0x0A | COMMAND | 7:0 | | | | | | | SPCONV | STCONV |
| 0x0B | EVCTRL | 7:0 | | | | | | | | STARTEI |
| 0x0C | INTCTRL | 7:0 | | | | | | | WCMP | RESRDY |
| 0x0D | INTFLAGS | 7:0 | | | | | | | WCMP | RESRDY |
| 0x0E | DBGCTRL | 7:0 | | | | | | | | DBGRUN |
| 0x0F | TEMP | 7:0 | TEMP[7:0] | | | | | | | |
| 0x10 | RES | 7:0 | RES[7:0] | | | | | | | |
| | | 15:8 | RES[15:8] | | | | | | | |
| 0x12 | WINLT | 7:0 | WINLT[7:0] | | | | | | | |
| | | 15:8 | WINLT[15:8] | | | | | | | |
| 0x14 | WINHT | 7:0 | WINHT[7:0] | | | | | | | |
| | | 15:8 | WINHT[15:8] | | | | | | | |

## 33.5 Register Description

### 33.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | | CONVMODE | LEFTADJ | RESSEL[1:0] | | FREERUN | ENABLE |
| 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 – RUNSTDBY** Run in Standby
This bit determines whether the ADC still runs during Standby.

| Value | Description |
|---|---|
| 0 | ADC will not run in Standby sleep mode. An ongoing conversion will finish before the ADC enters sleep mode |
| 1 | ADC will run in Standby sleep mode |

**Bit 5 – CONVMODE** Conversion Mode
This bit defines if the ADC is working in Single-Ended or Differential mode.

| Value | Name | Description |
|---|---|---|
| 0x0 | SINGLEENDED | The ADC is operating in Single-Ended mode where only the positive input is used. The ADC result is presented as an unsigned value. |
| 0x1 | DIFF | The ADC is operating in Differential mode where both positive and negative inputs are used. The ADC result is presented as a signed value. |

**Bit 4 – LEFTADJ** Left Adjust Result
Writing a '1' to this bit will enable left adjustment of the ADC result.

**Bits 3:2 – RESSEL[1:0]** Resolution Selection
This bit field selects the ADC resolution. When changing the resolution from 12-bit to 10-bit, the conversion time is reduced from 13.5 CLK_ADC cycles to 11.5 CLK_ADC cycles.

| Value | Description |
|---|---|
| 0x00 | 12-bit resolution |
| 0x01 | 10-bit resolution |
| Other | Reserved |

**Bit 1 – FREERUN** Free-Running
Writing a '1' to this bit will enable the Free-Running mode for the ADC. The first conversion is started by writing a '1' to the Start Conversion (STCONV) bit in the Command (ADCn.COMMAND) register.

**Bit 0 – ENABLE** ADC Enable

| Value | Description |
|---|---|
| 0 | ADC is disabled |
| 1 | ADC is enabled |

### 33.5.2 Control B

**Name:** CTRLB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

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

**Bits 2:0 – SAMPNUM[2:0]** Sample Accumulation Number Select
This bit field selects how many consecutive ADC sampling results are accumulated automatically. When this bit field is written to a value greater than `0x0`, the according number of consecutive ADC sampling results are accumulated into the ADC Result (ADCn.RES) register.

| Value | Name | Description |
|---|---|---|
| `0x0` | NONE | No accumulation |
| `0x1` | ACC2 | 2 results accumulated |
| `0x2` | ACC4 | 4 results accumulated |
| `0x3` | ACC8 | 8 results accumulated |
| `0x4` | ACC16 | 16 results accumulated |
| `0x5` | ACC32 | 32 results accumulated |
| `0x6` | ACC64 | 64 results accumulated |
| `0x7` | ACC128 | 128 results accumulated |

### 33.5.3 Control C

**Name:** CTRLC
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

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

**Bits 3:0 – PRESC[3:0]** Prescaler
This bit field defines the division factor from the peripheral clock (CLK_PER) to the ADC clock (CLK_ADC).

| Value | Name | Description |
|---|---|---|
| 0x0 | DIV2 | CLK_PER divided by 2 |
| 0x1 | DIV4 | CLK_PER divided by 4 |
| 0x2 | DIV8 | CLK_PER divided by 8 |
| 0x3 | DIV12 | CLK_PER divided by 12 |
| 0x4 | DIV16 | CLK_PER divided by 16 |
| 0x5 | DIV20 | CLK_PER divided by 20 |
| 0x6 | DIV24 | CLK_PER divided by 24 |
| 0x7 | DIV28 | CLK_PER divided by 28 |
| 0x8 | DIV32 | CLK_PER divided by 32 |
| 0x9 | DIV48 | CLK_PER divided by 48 |
| 0xA | DIV64 | CLK_PER divided by 64 |
| 0xB | DIV96 | CLK_PER divided by 96 |
| 0xC | DIV128 | CLK_PER divided by 128 |
| 0xD | DIV256 | CLK_PER divided by 256 |
| Other | - | Reserved |

### 33.5.4 Control D

**Name:** CTRLD
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INITDLY[2:0] | | | | SAMPDLY[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 7:5 – INITDLY[2:0]** Initialization Delay
This bit field defines the initialization delay before the first sample when enabling the ADC or changing to an internal reference voltage. Setting this delay will ensure that the components of ADC are ready before starting the first conversion. The initialization delay will also be applied when waking up from deep Sleep to do a measurement.
The delay is expressed as a number of CLK_ADC cycles.

| Value | Name | Description |
|---|---|---|
| 0x0 | DLY0 | Delay 0 CLK_ADC cycles |
| 0x1 | DLY16 | Delay 16 CLK_ADC cycles |
| 0x2 | DLY32 | Delay 32 CLK_ADC cycles |
| 0x3 | DLY64 | Delay 64 CLK_ADC cycles |
| 0x4 | DLY128 | Delay 128 CLK_ADC cycles |
| 0x5 | DLY256 | Delay 256 CLK_ADC cycles |
| Other | - | Reserved |

**Bits 3:0 – SAMPDLY[3:0]** Sampling Delay
This bit field defines the delay between consecutive ADC samples. This allows modifying the sampling frequency used during hardware accumulation, to suppress periodic noise that may otherwise disturb the sampling. The delay is expressed as CLK_ADC cycles and is given directly by the bit field setting.

| Value | Name | Description |
|---|---|---|
| 0x0 | DLY0 | Delay 0 CLK_ADC cycles |
| 0x1 | DLY1 | Delay 1 CLK_ADC cycles |
| 0x2 | DLY2 | Delay 2 CLK_ADC cycles |
| ... | ... | |
| 0xF | DLY15 | Delay 15 CLK_ADC cycles |

### 33.5.5 Control E

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

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

**Bits 2:0 – WINCM[2:0]** Window Comparator Mode
This bit field enables the Window Comparator and defines when the Window Comparator Interrupt Flag (WCMP) in the Interrupt Flags (ADCn.INTFLAGS) register is set. In the table below, RESULT is the accumulated 16-bit result. WINLT and WINHT are the 16-bit lower threshold value and the 16-bit upper threshold value given by the ADCn.WINLT and ADCn.WINHT registers, respectively.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No Window Comparison (default) |
| 0x1 | BELOW | *RESULT < WINLT* |
| 0x2 | ABOVE | *RESULT > WINHT* |
| 0x3 | INSIDE | *WINLT ≤ RESULT ≤ WINHT* |
| 0x4 | OUTSIDE | *RESULT < WINLT or RESULT >WINHT* |
| Other | - | Reserved |

### 33.5.6 Sample Control

**Name:** SAMPCTRL
**Offset:** 0x05
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SAMPLEN[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 – SAMPLEN[7:0]** Sample Length
This bit field extends the ADC sampling time with the number of CLK_ADC cycles given by the bit field value. Increasing the sampling time allows sampling sources with higher impedance. By default, the sampling time is two CLK_ADC cycles. The total conversion time increases with the selected sampling length.

### 33.5.7 MUX Selection for Positive ADC Input

**Name:** MUXPOS
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | MUXPOS[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 6:0 – MUXPOS[6:0]** MUX Selection for Positive ADC Input
This bit field selects which analog input is connected to the positive input of the ADC. If this bit field is changed during a conversion, the change will not take effect until the conversion is complete.

| Value | Name | Description |
|---|---|---|
| 0x00-0x15 | AIN0-AIN21 | ADC input pin 0-21 |
| 0x16-0x3F | - | Reserved |
| 0x40 | GND | Ground |
| 0x41 | - | Reserved |
| 0x42 | TEMPSENSE | Temperature sensor |
| 0x43 | - | Reserved |
| 0x44 | VDDDIV10 | VDD divided by 10 |
| 0x45 | VDDIO2DIV10 | VDDIO2 divided by 10 |
| 0x46-0x47 | - | Reserved |
| 0x48 | DAC0 | DAC0 |
| 0x49 | DACREF0 | DACREF0 |
| 0x4A | DACREF1 | DACREF1 |
| 0x4B | DACREF2 | DACREF2 |
| Other | - | Reserved |

### 33.5.8 MUX Selection for Negative ADC Input

**Name:** MUXNEG
**Offset:** 0x09
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | MUXNEG[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 6:0 – MUXNEG[6:0]** MUX Selection for Negative ADC Input
This bit field selects which analog input is connected to the negative input of the ADC. If this bit field is changed during a conversion, the change will not take effect until the conversion is complete.

| Value | Name | Description |
|---|---|---|
| 0x00-0x0F | AIN0-AIN15 | ADC input pin 0-15 |
| 0x10-0x3F | - | Reserved |
| 0x40 | GND | Ground |
| 0x41-0x47 | - | Reserved |
| 0x48 | DAC0 | DAC0 |
| Other | - | Reserved |

### 33.5.9 Command

**Name:** COMMAND
**Offset:** 0x0A
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | SPCONV | STCONV |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – SPCONV** Stop Conversion
Writing a '1' to this bit will end the current measurement. This bit will take precedence over the Start Conversion (STCONV) bit. Writing a '0' to this bit has no effect.

**Bit 0 – STCONV** Start Conversion
Writing a '1' to this bit will start a conversion as soon as any ongoing conversions are completed. If in Free-Running mode, this will start the first conversion. STCONV will read as '1' as long as a conversion is in progress. When the conversion is complete, this bit is automatically cleared. Writing a '0' to this bit has no effect.

### 33.5.10 Event Control

**Name:** EVCTRL
**Offset:** 0x0B
**Reset:** 0x00
**Property:** -

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

**Bit 0 – STARTEI** Start Event Input
This bit enables the event input as trigger for starting a conversion. When a '1' is written to this bit, a rising event edge will trigger an ADC conversion.

### 33.5.11 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x0C
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | WCMP | RESRDY |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – WCMP** Window Comparator Interrupt Enable
Writing a '1' to this bit enables the window comparator interrupt.

**Bit 0 – RESRDY** Result Ready Interrupt Enable
Writing a '1' to this bit enables the Result Ready interrupt.

### 33.5.12 Interrupt Flags

**Name:** INTFLAGS
**Offset:** 0x0D
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | WCMP | RESRDY |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – WCMP** Window Comparator Interrupt Flag
This window comparator flag is set when the measurement is complete and if the result matches the selected Window Comparator mode defined by the WINCM bit field in the Control E (ADCn.CTRLE) register. The comparison is done at the end of the conversion. The flag is cleared by either writing a '1' to the bit position or by reading the Result (ADCn.RES) register. Writing a '0' to this bit has no effect.

**Bit 0 – RESRDY** Result Ready Interrupt Flag
The Result Ready interrupt flag is set when a measurement is complete and a new result is ready. The flag is cleared by either writing a '1' to the bit location or by reading the Result (ADCn.RES) register. Writing a '0' to this bit has no effect.

### 33.5.13 Debug Control

**Name:** DBGCTRL
**Offset:** 0x0E
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Run in Debug Mode
When written to '1', the peripheral will continue operating in Debug mode when the CPU is halted.

### 33.5.14 Temporary

**Name:** TEMP
**Offset:** 0x0F
**Reset:** 0x00
**Property:** -

The Temporary register is used by the CPU for 16-bit single-cycle access to the 16-bit registers of this peripheral. The register is common for all the 16-bit registers of this peripheral and can be read and written by software. For more details on reading and writing 16-bit registers, refer to *Accessing 16-Bit Registers* in the *Memories* section.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TEMP[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 – TEMP[7:0]** Temporary
Temporary register for read and write operations to and from 16-bit registers.

### 33.5.15 Result

**Name:** RES
**Offset:** 0x10
**Reset:** 0x00
**Property:** -

The ADCn.RESL and ADCn.RESH register pair represents the 16-bit value, ADCn.RES. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

Refer to the 33.3.3.5 Conversion Result (Output Formats) section for details on the output from this register.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | RES[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 |
|---|---|---|---|---|---|---|---|---|
| | RES[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:8 – RES[15:8]** Result High Byte
This bit field constitutes the high byte of the ADCn.RES register, where the MSb is RES[15].

**Bits 7:0 – RES[7:0]** Result Low Byte
This bit field constitutes the low byte of the ADCn.RES register.

### 33.5.16 Window Comparator Low Threshold

**Name:** WINLT
**Offset:** 0x12
**Reset:** 0x00
**Property:** -

This register is the 16-bit low threshold for the digital comparator monitoring the Result (ADCn.RES) register. The data format must be according to the Conversion mode and left/right adjustment setting.

The ADCn.WINLTH and ADCn.WINLTL register pair represents the 16-bit value, ADCn.WINLT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

| 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 15:8 – WINLT[15:8]** Window Comparator Low Threshold High Byte
This bit field holds the MSB of the 16-bit register.

**Bits 7:0 – WINLT[7:0]** Window Comparator Low Threshold Low Byte
This bit field holds the LSB of the 16-bit register.

### 33.5.17 Window Comparator High Threshold

**Name:** WINHT
**Offset:** 0x14
**Reset:** 0x00
**Property:** -

This register is the 16-bit high threshold for the digital comparator monitoring the Result (ADCn.RES) register. The data format must be according to the Conversion mode and left/right adjustment setting.

The ADCn.WINHTH and ADCn.WINHTL register pair represents the 16-bit value, ADCn.WINHT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + `0x01`.

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

**Bits 15:8 – WINHT[15:8]** Window Comparator High Threshold High Byte
This bit field holds the MSB of the 16-bit register.

**Bits 7:0 – WINHT[7:0]** Window Comparator High Threshold Low Byte
This bit field holds the LSB of the 16-bit register.

# 34. DAC - Digital-to-Analog Converter

## 34.1 Features

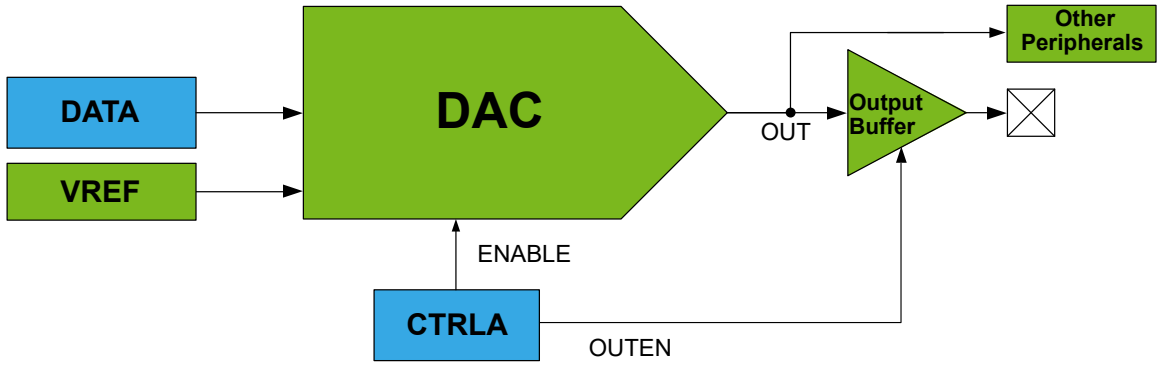
- 10-bit Resolution
- Up to 140 ksps Conversion Rate
- High Drive Capabilities
- The DAC Output Can Be Used as Input to the ADC Positive Input

## 34.2 Overview

The Digital-to-Analog Converter (DAC) converts a digital value written to the Data (DACn.DATA) register to an analog voltage. The conversion range is between GND and the selected voltage reference in the Voltage Reference (VREF) peripheral. The DAC has one continuous time output with high drive capabilities. The DAC conversion can be started from the application by writing to the Data (DACn.DATA) register.

### 34.2.1 Block Diagram

**Figure 34-1. DAC Block Diagram**




### 34.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| OUT | DAC output | Analog |

## 34.3 Functional Description

### 34.3.1 Initialization

To operate the DAC, the following steps are required:

1. Select the DAC reference voltage in the Voltage Reference (VREF) peripheral by writing the appropriate Reference Selection bits.
2. Configure the further usage of the DAC output:
   - Configure an internal peripheral to use the DAC output. Refer to the documentation of the respective peripherals.
   - Enable the output to a pin by writing a '1' to the Output Buffer Enable (OUTEN) bit. The input for the DAC pin must be disabled in the Port peripheral (ISC = INPUT_DISABLE in PORTx.PINCTRLn).
3. Write an initial digital value to the Data (DACn.DATA) register.

4. Enable the DAC by writing a '1' to the ENABLE bit in the Control A (DACn.CTRLA) register.

### 34.3.2 Operation

#### 34.3.2.1 Enabling, Disabling and Resetting

The DAC is enabled by writing a '1' to the ENABLE bit in the Control A (DACn.CTRLA) register, and disabled by writing a '0' to this bit.

#### 34.3.2.2 Starting a Conversion

When the ENABLE bit in the Control A (DACn.CTRLA) register is written to '1', a conversion starts as soon as the Data (DACn.DATA) register is written.

When the ENABLE bit in DACn.CTRLA is written to '0', writing to the Data register does not trigger a conversion. Instead, the conversion starts when the ENABLE bit in DACn.CTRLA is written to '1'.

#### 34.3.2.3 DAC as Source For Internal Peripherals

The analog output of the DAC can be internally connected to other peripherals when the ENABLE bit in the Control A (DACn.CTRLA) register is written to '1'. When the DAC analog output is only being used internally, the Output Buffer Enable (OUTEN) bit in DACn.CTRLA can be '0'.

#### 34.3.2.4 DAC Output on Pin

The analog output of the DAC can be connected to a pin by writing a '1' to the Output Buffer Enable (OUTEN) bit in the Control A (DACn.CTRLA) register. The pin used by the DAC must have the input disabled from the Port peripheral. There is an output buffer between the DAC output and the pin, which ensures the analog value does not depend on the load of the pin. The output buffer can only source current, it has very limited sinking capability.

### 34.3.3 Sleep Mode Operation

If the Run in Standby (RUNSTDBY) bit in the Control A (DACn.CTRLA) register is written to '1', the DAC will continue to operate in Standby sleep mode. If the RUNSTDBY bit is zero, the DAC will stop the conversion in Standby sleep mode.

If the conversion is stopped in Standby sleep mode, the DAC and the output buffer are disabled to reduce power consumption. When the device is exiting Standby sleep mode, the DAC and the output buffer (if the OUTEN bit in the Control A (DACn.CTRLA) register is written to '1') are enabled again. Therefore, a start-up time is required before a new conversion is initiated.

In Power-Down sleep mode, the DAC and the output buffer are disabled to reduce power consumption.

## 34.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | RUNSTDBY | OUTEN | | | | | | ENABLE |
| 0x01 | Reserved | | | | | | | | | |
| 0x02 | DATA | 7:0 | DATA[1:0] | | | | | | | |
| | | 15:8 | DATA[9:2] | | | | | | | |

## 34.5 Register Description

### 34.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | OUTEN | | | | | | ENABLE |
| Access | R/W | R/W | | | | | | R/W |
| Reset | 0 | 0 | | | | | | 0 |

**Bit 7 – RUNSTDBY** Run in Standby Mode
If this bit is written to '1', the DAC or the output buffer will not automatically be disabled when the device is entering Standby sleep mode.

**Bit 6 – OUTEN** Output Buffer Enable
Writing a '1' to this bit enables the output buffer and sends the OUT signal to a pin.

**Bit 0 – ENABLE** DAC Enable
Writing a '1' to this bit enables the DAC.

### 34.5.2 DATA

**Name:** DATA
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

The DACn.DATAL and DACn.DATAH register pair represents the 10-bit value, DACn.DATA. The two LSbs [1:0] are accessible at the original offset. The eight MSbs [9:2] can be accessed at offset + `0x01`.

The output will be updated after DACn.DATAH is written.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | DATA[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 |
|---|---|---|---|---|---|---|---|---|
| | DATA[1:0] | | | | | | | |
| Access | R/W | R/W | | | | | | |
| Reset | 0 | 0 | | | | | | |

**Bits 15:6 – DATA[9:0]**
These bits contain the digital data, which will be converted to an analog voltage.

# 35. OPAMP - Analog Signal Conditioning

## 35.1 Features

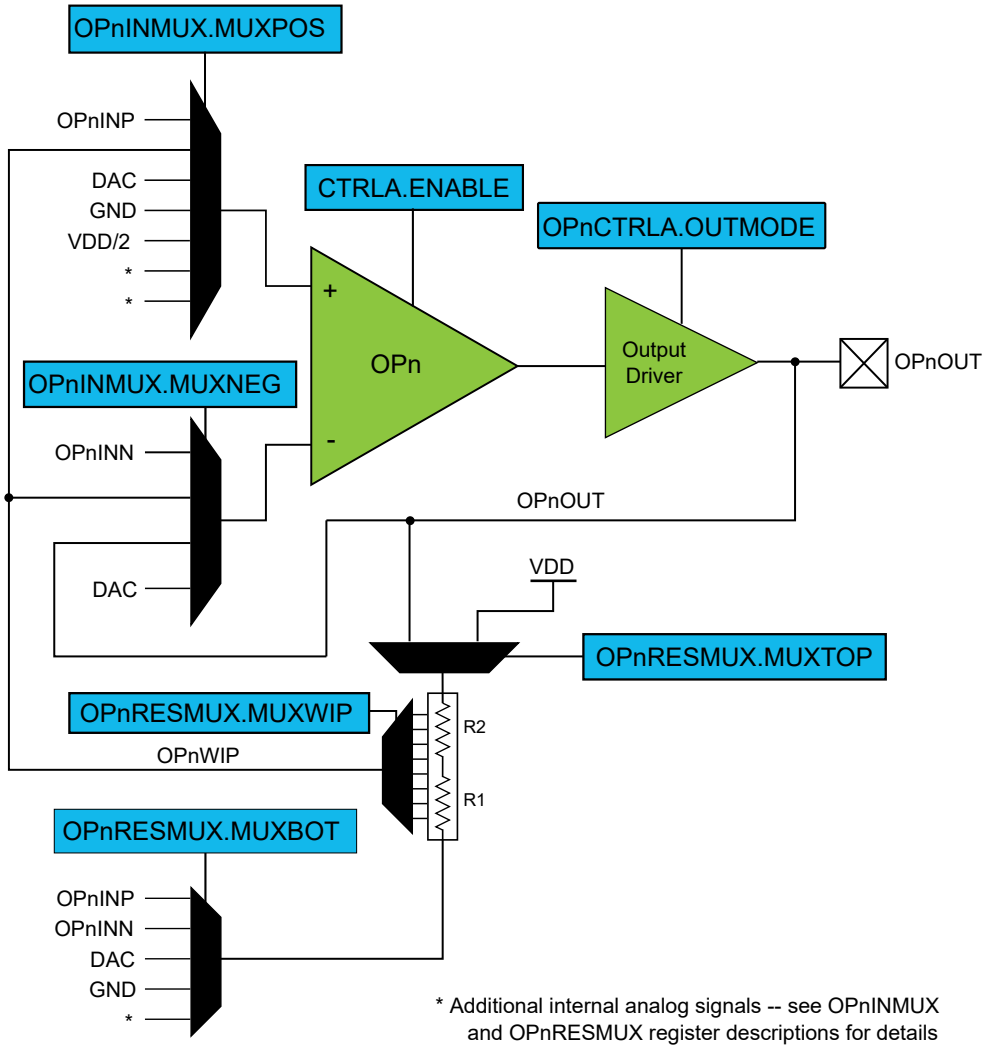
- Internal Resistor Ladder Facilitates Analog Signal Conditioning with Zero External Components
- Selectable Configurations:
  - Standalone general-purpose operational amplifier (op amp)
  - Unity-gain buffer
  - Non-inverting/inverting Programmable Gain Amplifier (PGA)
  - Cascaded PGAs
  - Instrumentation amplifier
- Input Selection:
  - I/O pins
  - DAC
  - Ground
  - $V_{DD}/2$ reference
  - Output from another op amp
  - Internal resistor ladder
- Output Selection:
  - On I/O pins
  - As input for ADC
  - As input for AC
  - As input for another op amp
- Internal Timer Generates READY Event When Settling is Complete
- Low-Power Support:
  - Optional event-triggered operation
- Event-System-Controlled Dump Mode to Support Signal Integration
- Offset and Gain Calibration Using the ADC

## 35.2 Overview

The Analog Signal Conditioning (OPAMP) peripheral features three operational amplifiers (op amps), designated OPn where n is zero, one or two. These op amps are implemented with a flexible connection scheme using analog multiplexers and resistor ladders. This allows a large number of analog signal conditioning configurations to be achieved, many of which require no external components. A multiplexer at the non-inverting (+) input of each op amp allows connection to either an external pin, a wiper position from a resistor ladder, a DAC output, ground, $V_{DD}/2$, or an output from another op amp. A second multiplexer at the inverting (-) input of each op amp allows connection to either an external pin, a wiper position from a resistor ladder, the output of the op amp, or DAC output. Three more multiplexers connected to each resistor ladder provide additional configuration flexibility. Two of these multiplexers select the top and bottom connections to the resistor ladder, and the third controls the wiper position.

### 35.2.1 Block Diagram

**Figure 35-1. Op Amp n (OPn) Block Diagram**




### 35.2.2 Signal Description

| Signal Name | Type | Description |
|---|---|---|
| OPnINP | Analog input | Non-inverting (+) input pin for OPn |
| OPnINN | Analog input | Inverting (-) input pin for OPn |
| OPnOUT | Analog output | Output from OPn |

## 35.3 Functional Description

### 35.3.1 Initialization

To initialize the OPAMP peripheral for basic, always-on operation, the following steps are recommended:

1. Configure the timebase of the OPAMP peripheral by writing to the Timebase (TIMEBASE) bit field in the Timebase (OPAMP.TIMEBASE) register.
2. For each op amp that will be used:
   2.1. Configure the Op Amp n Input Multiplexer (OPAMP.OPnINMUX):
      - Select the non-inverting (+) input of the op amp by writing to the Multiplexer for Positive Input (MUXPOS) bit field
      - Select the inverting (-) input of the op amp by writing to the Multiplexer for Negative Input (MUXNEG) bit field

   2.2. Configure the Op Amp n Resistor Ladder Multiplexer (OPAMP.OPnRESMUX):
      - Select the connection to the top resistor in the resistor ladder by writing to the Multiplexer for Top (MUXTOP) bit field
      - Select the connection to the bottom resistor in the resistor ladder by writing to the Multiplexer for Bottom (MUXBOT) bit field
      - Select the wiper position in the resistor ladder by writing to the Multiplexer for Wiper (MUXWIP) bit field

   2.3. Configure the Op Amp n Control A (OPAMP.OPnCTRLA) register:
      - Configure the op amp to be always on by writing '1' to the Always On (ALWAYSON) bit
      - Disable events for the op amp by writing '0' to the Event Enable (EVENTEN) bit
      - Select the normal output mode by writing the NORMAL setting to the Output Mode (OUTMODE) bit field

   2.4. *Optional:* Configure the allowed settling time by writing to the Settle Timer (SETTLE) bit field in the Op Amp n Settle Timer (OPAMP.OPnSETTLE) register.
3. Enable the OPAMP peripheral by writing a '1' to the OPAMP Enable (ENABLE) bit in the Control A (OPAMP.CTRLA) register.
4. For each op amp whose settling time was configured in step 2.4 above, wait for the SETTLED bit in the Op Amp n Status (OPAMP.OPnSTATUS) register to become '1'. This indicates that the op amp start-up and settling have completed, and the OPAMP peripheral is ready for use.

### 35.3.2 Operation

#### 35.3.2.1 MUXPOS - Non-Inverting (+) Input Selection

As shown in Figure 35-1, the non-inverting (+) input of an op amp is connected to an analog multiplexer that allows one input source to be selected out of a variety of input sources. The source selection is configured using the MUXPOS bit field of the Op Amp n Input Multiplexer (OPAMP.OPnINMUX) register. If the OPnINMUX register is changed while the OPn output is enabled, a glitch will occur on the output signal due to the opening and closing of analog switches, and some time will elapse before the output settles to the new value.

The following non-inverting input options are available for OPn:

- INP (OPnINP) - Positive input pin for OPn
- WIP (OPnWIP) - Wiper from OPn's resistor ladder
- DAC - DAC output (DAC and DAC output buffer must be enabled)
- GND - Ground
- VDDDIV2 - $V_{DD}/2$

OP1 has the following additional input available:

- LINKOUT (OP0OUT) - OP0 output

OP2 has the following additional inputs available:

- LINKOUT (OP1OUT) - OP1 output
- LINKWIP (OP0WIP) - Wiper from OP0's resistor ladder

#### 35.3.2.2 MUXNEG - Inverting (-) Input Selection

As shown in Figure 35-1, the inverting (-) input of an op amp is also connected to an analog multiplexer that allows one of a variety of input sources to be chosen. The selection is configured using the MUXNEG bit field of the Op Amp n Input Multiplexer (OPAMP.OPnINMUX) register. If the OPnINMUX register is changed while the OPn output is

enabled, a glitch will occur on the output signal due to the opening and closing of analog switches, and some time will elapse before the output settles to the new value.

The following inverting input options are available for OPn:

- INN (OPnINN) - Negative input pin for OPn
- WIP (OPnWIP) - Wiper from OPn's resistor ladder
- OUT (OPnOUT) - OPn output
- DAC - DAC output (DAC and DAC output buffer must be enabled)

#### 35.3.2.3 MUXTOP, MUXBOT, MUXWIP - Resistor Ladder Configuration

A resistor ladder and three connected multiplexers are associated with each op amp, as shown in Figure 35-1. These multiplexers are configured using the MUXTOP, MUXBOT and MUXWIP bit fields of the Op Amp n Resistor Ladder Multiplexer (OPAMP.OPnRESMUX) register. If the OPnRESMUX register is changed while the OPn output is enabled, a glitch will occur on the output signal due to the opening and closing of analog switches, and some time will elapse before the output settles to the new value.

The top of the resistor ladder is connected to a multiplexer that is controlled by the MUXTOP bit field. It can connect to the following signals:

- OFF - Multiplexer off, no signal connected
- OUT (OPnOUT) - OPn output
- VDD - $V_{DD}$

The bottom of the resistor ladder is also connected to a multiplexer. This multiplexer is controlled by the MUXBOT bit field, and it can connect to the following signals:

- OFF - Multiplexer off, no signal connected
- INP (OPnINP) - Positive input pin for OPn
- INN (OPnINN) - Negative input pin for OPn
- DAC - DAC output (DAC and DAC output buffer must be enabled)
- GND - Ground

OP0 has the following additional connection available via the MUXBOT bit field:

- LINKOUT (OP2OUT) - OP2 output

OP1 has the following additional connection available via the MUXBOT bit field:

- LINKOUT (OP0OUT) - OP0 output

OP2 has the following additional connection available via the MUXBOT bit field:

- LINKOUT (OP1OUT) - OP1 output

A third multiplexer is connected to eight different wiper positions on the resistor ladder. This multiplexer is controlled by the MUXWIP bit field and allows the OPnWIP signal to be connected to any of eight different ratios of the upper (R2) and lower (R1) resistors:

- WIP0 - R2/R1 = 1/15
- WIP1 - R2/R1 = 1/7
- WIP2 - R2/R1 = 1/3
- WIP3 - R2/R1 = 1
- WIP4 - R2/R1 = 5/3
- WIP5 - R2/R1 = 3
- WIP6 - R2/R1 = 7
- WIP7 - R2/R1 = 15

The tolerances of the resistor ratios, as well as the absolute values and tolerances of the resistors, are specified in the *Electrical Characteristics* section.

#### 35.3.2.4 Output Modes

As shown in Figure 35-1, an op amp has an output driver that is directly connected to an OPnOUT pin. When the output driver for OPn is enabled, the OPnOUT pin cannot be driven by other peripherals. The output driver can

operate in different modes controlled by the OUTMODE bit field of the OPnCTRLA register. The following output modes are available:

- OFF - The output driver is disabled, but can be overridden by a DRIVEn event
- NORMAL - The output driver is enabled and operates normally

#### 35.3.2.5 Input Voltage Range

In applications where a rail-to-rail input voltage range is not needed, the OPAMP peripheral may be configured to save power. Writing '1' to the Input Range Select (IRSEL) bit in the Power Control (PWRCTRL) register reduces the voltage range of the op amp inputs, which also reduces power consumption. See the *Electrical Characteristics* section for more information about the expected voltage range and power consumption.

#### 35.3.2.6 Internal Timer

When an op amp is enabled, it takes some time for the op amp to start up and begin functioning correctly. There are two components to this start-up time, *warmup* time and *settling* time:

1. The warmup time is the time required to elapse before the internal circuitry in the op amp stabilizes. The warmup time is documented in the *Electrical Characteristics* section. It does not depend on external circuitry.
2. The settling time is the additional time allowed for the op amp output to settle and become stable after the warmup time is completed. It depends on a variety of factors, including the strength of the output driver and the load on the op amp output.

Each op amp has an internal timer that is used to determine the delay between op amp enable and the indication that it is ready to use. For the internal timer to function correctly, two bit fields must be configured:

1. The timebase of the OPAMP peripheral must be configured by writing to the Timebase (TIMEBASE) bit field in the Timebase (OPAMP.TIMEBASE) register. Determine how many cycles of the peripheral clock (CLK_PER) are equal to 1 µs. If that number is an integer, subtract one from it and write it to the TIMEBASE bit field. If that number is not an integer, round it down to an integer and write it to the TIMEBASE bit field. For example, if the peripheral clock period is 260 ns, then 3.85 cycles are equal to 1 µs. Since 3.85 is not an integer, it should be rounded down to 3 and then written to the TIMEBASE bit field.
2. The settling time of OPn must be configured by writing to the Settle Timer (SETTLE) bit field in the Op Amp n Settle Timer (OPAMP.OPnSETTLE) register. Since the settling time depends on a variety of factors, including the load on the op amp, it may not be known until the later stages of design and development. If the settling time is unknown, the maximum value of '0x7F' (127 µs) should be written to the SETTLE bit field.

After the user-programmed settling time has elapsed, the SETTLED bit is changed from '0' to '1' in OPnSTATUS. If the op amp is in EVENT_ENABLED mode, the READYn event is also issued.

If the OPnINMUX or OPnRESMUX register is changed while the OPn output is enabled, a glitch will occur on the output signal due to the opening and closing of analog switches, and some time will elapse before the output settles to the new value. For this reason, the settle timer restarts whenever there is a write to the OPnINMUX or OPnRESMUX registers.

The settle timer also restarts whenever there is a write to the OPnCTRLA register.

#### 35.3.2.7 Enable and Disable

An op amp can be configured to be in one of three possible Enable/Disable modes:

- Enabled/disabled by software with all events disabled (SW_ENABLED_WITHOUT_EVENTS mode)
- Enabled/disabled by the event system (EVENT_ENABLED mode)
- Enabled/disabled by software with other events enabled (SW_ENABLED_WITH_EVENTS mode)

To select SW_ENABLED_WITHOUT_EVENTS mode, a '0' must be written to the EVENTEN bit in OPnCTRLA. In SW_ENABLED_WITHOUT_EVENTS mode, an op amp is enabled by writing '1' to the ALWAYSON bit in OPnCTRLA. The op amp stays enabled as long as the ALWAYSON bit is '1'. Writing a '0' to the ALWAYSON bit in OPnCTRLA will disable the op amp. The op amp will not respond to incoming events and will not generate events.

To select EVENT_ENABLED mode, the ALWAYSON bit in OPnCTRLA must be written to '0', and the EVENTEN bit must be written to '1'. In this mode, the op amp is enabled by the ENABLEn event, and it is disabled by the DISABLEn event. In EVENT_ENABLED mode, the op amp will issue a READYn event when settling is complete. The op amp will also respond to incoming DUMPn and DRIVEn events. Special cases are handled as follows:

- If an ENABLEn event is received while the op amp is in the process of starting up as a result of a prior ENABLEn event, the most recent ENABLEn event will be ignored, and the READYn event will be issued when the settling time from the first ENABLEn event has elapsed
- If an ENABLEn event is received while the op amp is already enabled and has completed settling, the op amp will stay enabled with no change in state. The READYn event will also be generated immediately after the ENABLEn event in this case.
- If a DISABLEn event is received while the op amp is already disabled, the DISABLEn event is ignored, and the op amp remains disabled
- If both an ENABLEn and DISABLEn event are received simultaneously, the DISABLEn event is ignored, and the ENABLEn event is handled as described in the cases above

To select SW_ENABLED_WITH_EVENTS mode, a '1' must be written to the EVENTEN bit in OPnCTRLA. In SW_ENABLED_WITH_EVENTS mode, an op amp is enabled by writing '1' to the ALWAYSON bit in OPnCTRLA. The op amp stays enabled as long as the ALWAYSON bit is '1'. While the op amp is enabled, it will respond to incoming DUMPn and DRIVEn events but will be unaffected by incoming ENABLEn and DISABLEn events. Writing a '0' to the ALWAYSON bit in OPnCTRLA will disable the op amp.

#### 35.3.2.8 Offset Calibration

The input offset voltage of an op amp can be calibrated by using the DAC or an external constant-voltage source in conjunction with the ADC.

To calibrate an op amp, perform the following steps:

- Use the DAC or the external I/O pin connected to the op amp non-inverting (+) input as a constant-voltage reference. If the DAC is used, it and its output buffer must be enabled.
- Configure the op amp with:
  - MUXPOS in the OPnINMUX register set to the selected calibration source
  - MUXNEG in the OPnINMUX register set to OPnOUT, so the op amp will function as a voltage follower
- Use the ADC to measure the voltage of the selected calibration source. This value becomes the calibration target.
- Configure the ADC input multiplexer to select the output from OPn, then use the ADC to measure this voltage
- Use the difference between the two ADC measurements to determine how much the calibration register (OPnCAL) should be adjusted. The calibration step size is found in the *Electrical Characteristics* section.

### 35.3.3 Events

An op amp must be in EVENT_ENABLED or SW_ENABLED_WITH_EVENTS mode to generate events and respond to incoming events. To select EVENT_ENABLED mode for OPn, the EVENTEN bit in OPnCTRLA must be written to '1', and the ALWAYSON bit must be written to '0'. To select SW_ENABLED_WITH_EVENTS mode for OPn, the EVENTEN bit in OPnCTRLA must be written to '1', and the ALWAYSON bit must be written to '1'.

OPn can generate the event as described in the table below.

**Table 35-1. Event Generators in OPAMP**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| OPAMP | READYn | OPn is ready | Pulse | CLK_PER | One CLK_PER period |

The table below describes the event users for OPn and their associated functionality.

**Table 35-2. Event Users in OPAMP**

| User Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| OPAMP | ENABLEn | Enable OPn | Edge | Async |
| OPAMP | DISABLEn | Disable OPn | Edge | Sync |

.....continued

| User Name | | Description | Input Detection | Async/Sync |
|---|---|---|---|---|
| Peripheral | Input | | | |
| OPAMP | DUMPn | Dump OPn $V_{OUT}$ to $V_{INN}$ | Level | Sync |
| OPAMP | DRIVEn | Enable OPn output driver in NORMAL mode | Level | Sync |

Refer to the *Event System (EVSYS)* section for more details regarding event types and Event System configuration.

### 35.3.4 Interrupts

The OPAMP peripheral does not generate interrupts.

### 35.3.5 Sleep Mode Operation

If the RUNSTDBY bit is '0' in OPnCTRLA, OPn is shut off in Standby Sleep mode, and its output driver is disabled. The op amp will release its override of the I/O pin, and the behavior of the I/O pin can be controlled using standard GPIO control.

If the RUNSTDBY bit is '1' in OPnCTRLA, OPn remains operational in Standby Sleep mode with its enable/disable behavior determined by the ALWAYSON and EVENTEN bits in OPnCTRLA.

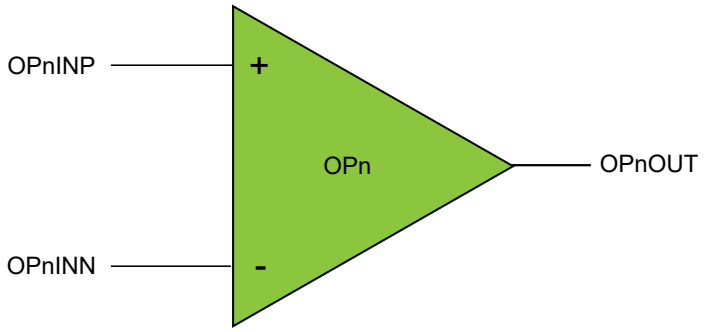
### 35.3.6 Debug Operation

When the CPU is halted in Debug mode, the analog portions of the OPAMP peripheral will continue to operate as before the CPU halt. If the DBGRUN bit is '1' in the DBGCTRL register, the digital interface of the OPAMP peripheral will also continue operating normally.

### 35.3.7 Application Usage

The OPAMP peripheral is highly flexible and can be used in various analog signal conditioning applications. This section describes a broad range of configurations and the multiplexer settings required to achieve them. Many of these configurations require no external components.

**Figure 35-2. Op Amp Connected Directly to Pins**

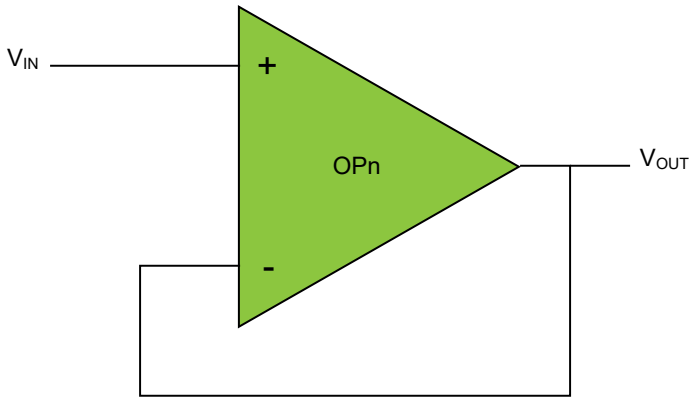



The figure above displays an op amp connected directly to the pins of the device without being connected to any of the internal resistors. This is useful for situations where the user desires to make all connections to other components externally. The multiplexer settings required to achieve this configuration are:

**Table 35-3. Op Amp Connected Directly to Pins**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OPn | INP | INN | OFF | WIP0 | OFF |

**Figure 35-3. Voltage Follower**

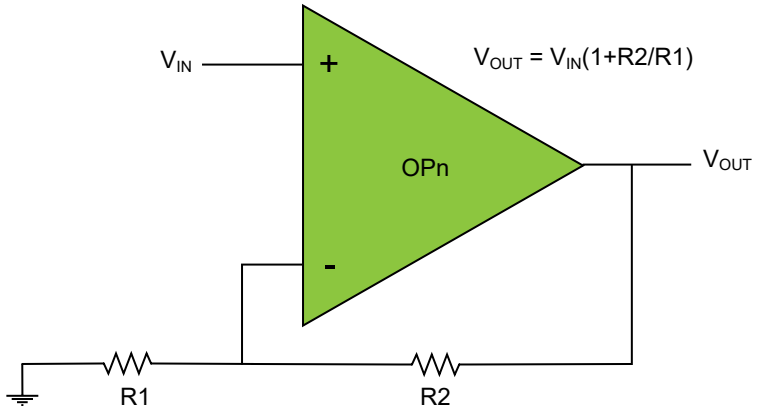



The figure above displays a voltage follower, also known as a unity-gain buffer. The non-inverting (+) input is connected to a pin, and the output is connected to the inverting (-) input. The multiplexer settings required for this configuration are:

**Table 35-4. Voltage Follower**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OPn | INP | OUT | OFF | WIP0 | OFF |

**Figure 35-4. Non-Inverting PGA**

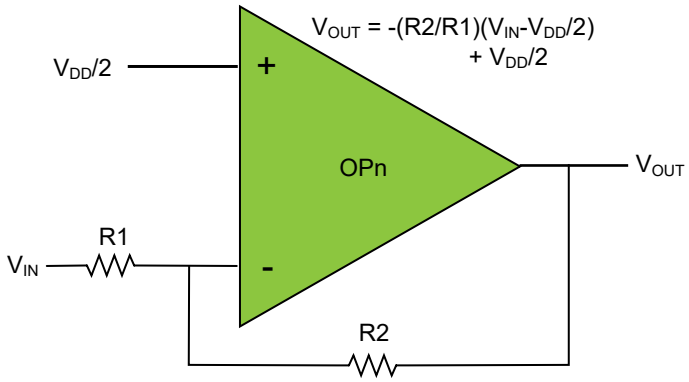



The figure above displays a non-inverting Programmable Gain Amplifier. The required multiplexer settings are as follows:

**Table 35-5. Non-Inverting PGA**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OPn | INP | WIP | GND | Setting determines gain | OUT |

**Figure 35-5. Inverting PGA**

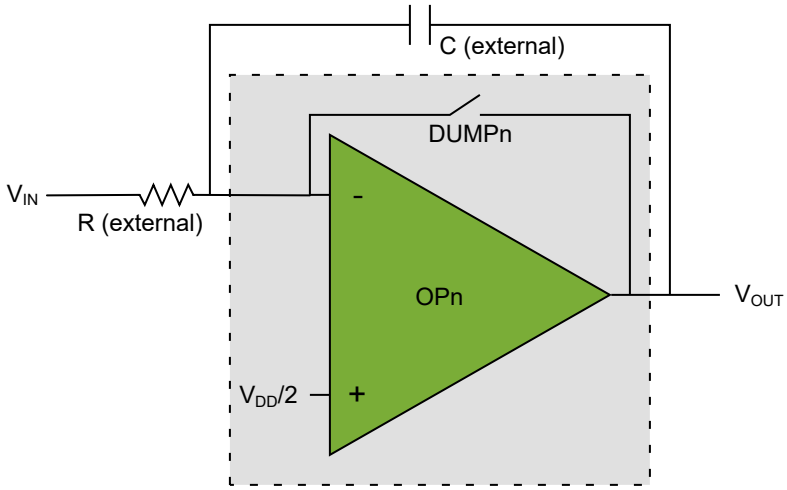



The figure above shows the inverting PGA configuration. The multiplexer settings required are:

**Table 35-6. Inverting PGA**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OPn | VDDDIV2 | WIP | INN | Setting determines gain | OUT |

**Figure 35-6. Integrator**

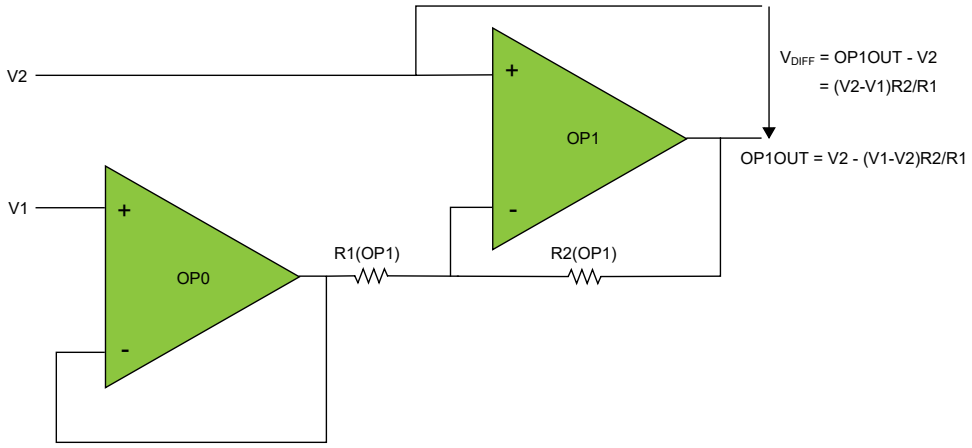



The figure above shows how an op amp can be configured as an integrator. An external capacitor and external resistor are required. The DUMPn event closes a switch to discharge the capacitor and reset the integrator. The multiplexer settings required are:

**Table 35-7. Integrator**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OPn | VDDDIV2 | INN | OFF | WIP0 | OFF |

**Figure 35-7. Differential Amplifier using Two Op Amps**

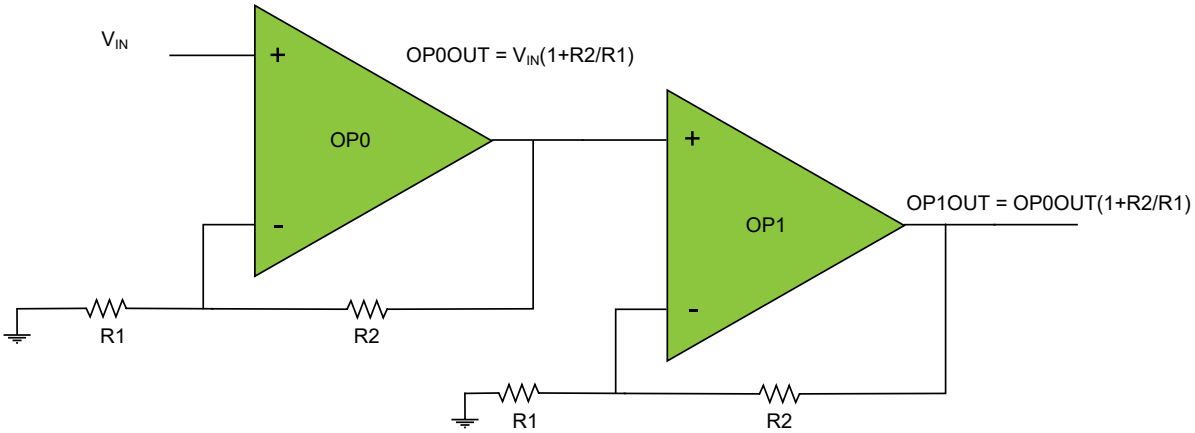



The figure above shows how a differential amplifier can be constructed from two op amps. The differential amplifier accepts two input signals, V1 and V2. The output also consists of two signals, OP1OUT and V2. The difference of the output signals, $V_{DIFF}$, is proportional to the difference (V2-V1) of the two input signals. The multiplexer settings required to implement this configuration are as follows:

**Table 35-8. Differential Amplifier using Two Op Amps**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OP0 | INP | OUT | OFF | WIP0 | OFF |
| OP1 | INP | WIP | LINKOUT (OP0OUT) | Setting determines gain | OUT |

**Figure 35-8. Cascaded (Two) Non-Inverting PGA**



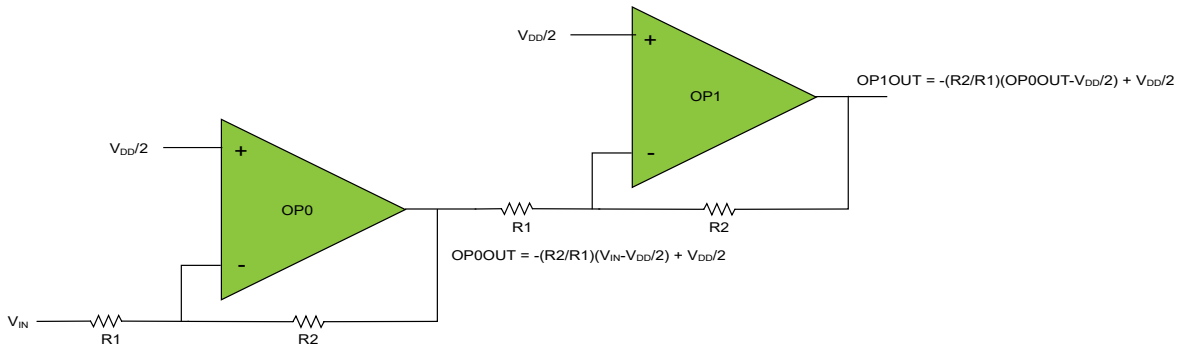

The figure above shows a cascaded non-inverting PGA constructed from two op amps. The following multiplexer settings are required:

**Table 35-9. Cascaded (Two) Non-Inverting PGA**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OP0 | INP | WIP | GND | Determines gain | OUT |

| .....continued | | | | | |
|---|---|---|---|---|---|
| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
| OP1 | LINKOUT (OP0OUT) | WIP | GND | Determines gain | OUT |

**Figure 35-9. Cascaded (Two) Inverting PGA**

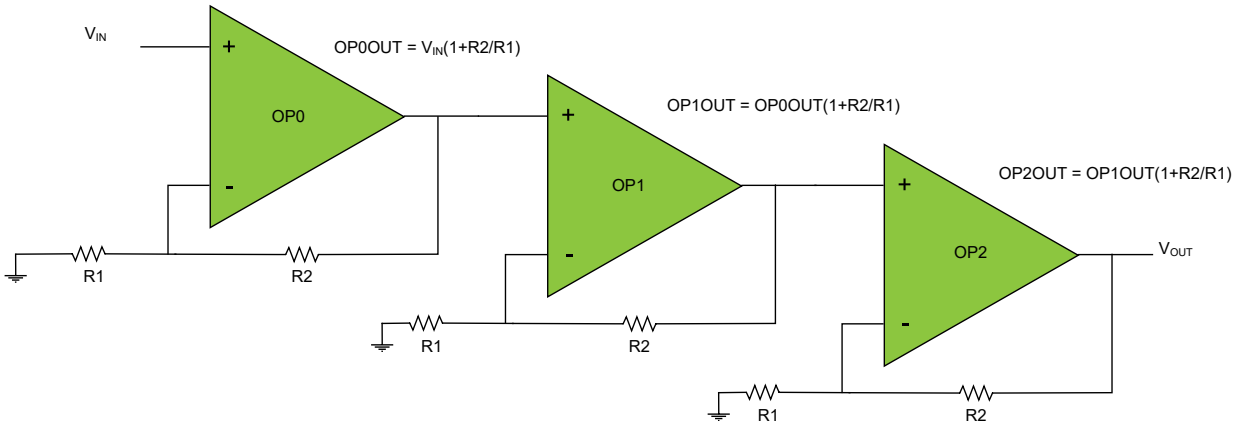



The figure above shows a cascaded inverting PGA constructed from two op amps. The required multiplexer settings are:

**Table 35-10. Cascaded (Two) Inverting PGA**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OP0 | VDDDIV2 | WIP | INN | Determines gain | OUT |
| OP1 | VDDDIV2 | WIP | LINKOUT (OP0OUT) | Determines gain | OUT |

**Figure 35-10. Cascaded (Three) Non-Inverting PGA**




The figure above shows a cascaded non-inverting PGA constructed from three op amps. The following multiplexer settings are required:

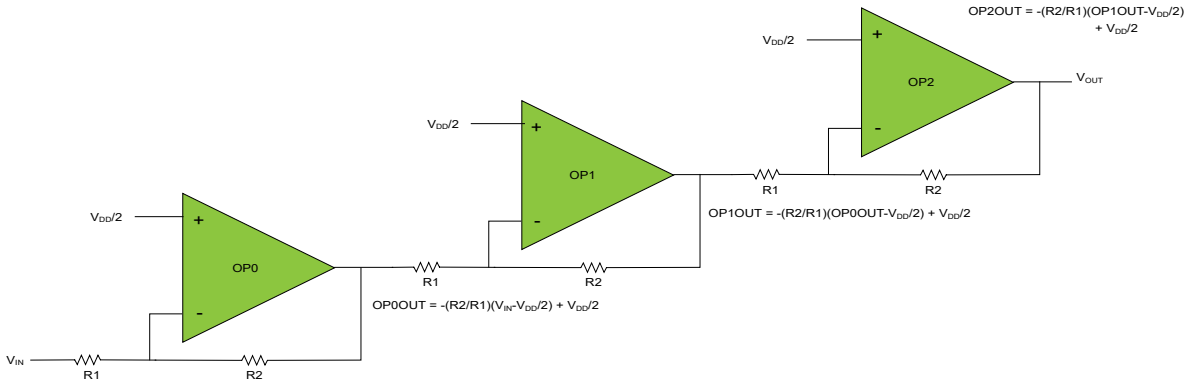
**Table 35-11. Cascaded (Three) Non-Inverting PGA**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OP0 | INP | WIP | GND | Determines gain | OUT |

.....continued

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OP1 | LINKOUT (OP0OUT) | WIP | GND | Determines gain | OUT |
| OP2 | LINKOUT (OP1OUT) | WIP | GND | Determines gain | OUT |

**Figure 35-11. Cascaded (Three) Inverting PGA**

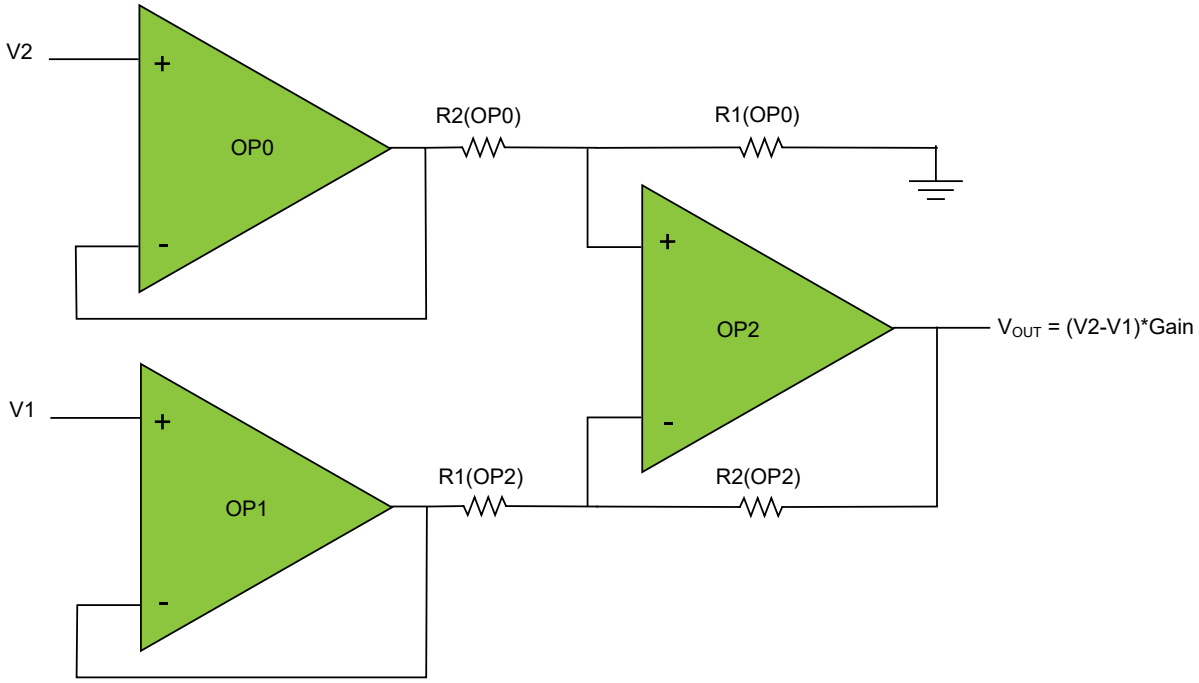



The figure above shows a cascaded inverting PGA constructed from three op amps. The required multiplexer settings are:

**Table 35-12. Cascaded (Three) Inverting PGA**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OP0 | VDDDIV2 | WIP | INN | Determines gain | OUT |
| OP1 | VDDDIV2 | WIP | LINKOUT (OP0OUT) | Determines gain | OUT |
| OP2 | VDDDIV2 | WIP | LINKOUT (OP1OUT) | Determines gain | OUT |

**Figure 35-12. Instrumentation Amplifier**




The figure above displays three op amps configured as an instrumentation amplifier. The following multiplexer settings are required:

**Table 35-13. Instrumentation Amplifier**

| | MUXPOS | MUXNEG | MUXBOT | MUXWIP | MUXTOP |
|---|---|---|---|---|---|
| OP0 | INP | OUT | GND | See the table below | OUT |
| OP1 | INP | OUT | OFF | WIP0 | OFF |
| OP2 | LINKWIP (OP0WIP) | WIP | LINKOUT (OP1OUT) | See the table below | OUT |

The resistor ladders associated with OP0 and OP2 must be configured as follows to select the desired gain:

**Table 35-14. Gain Selection for Instrumentation Amplifier**

| GAIN | OP0RESMUX.MUXWIP | OP2RESMUX.MUXWIP |
|---|---|---|
| 1/15 | `0x7` | `0x0` |
| 1/7 | `0x6` | `0x1` |
| 1/3 | `0x5` | `0x2` |
| 1 | `0x3` | `0x3` |
| 3 | `0x2` | `0x5` |
| 7 | `0x1` | `0x6` |
| 15 | `0x0` | `0x7` |

## 35.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | | | | | | | | ENABLE |
| 0x01 | DBGCTRL | 7:0 | | | | | | | | DBGRUN |
| 0x02 | TIMEBASE | 7:0 | | TIMEBASE[6:0] | | | | | | |
| 0x03 ... 0x0E | Reserved | | | | | | | | | |
| 0x0F | PWRCTRL | 7:0 | | | | | | | | IRSEL |
| 0x10 | OP0CTRLA | 7:0 | RUNSTDBY | | | | OUTMODE[1:0] | | EVENTEN | ALWAYSON |
| 0x11 | OP0STATUS | 7:0 | | | | | | | | SETTLED |
| 0x12 | OP0RESMUX | 7:0 | MUXWIP[2:0] | | | MUXBOT[2:0] | | | MUXTOP[1:0] | |
| 0x13 | OP0INMUX | 7:0 | | MUXNEG[2:0] | | | | MUXPOS[2:0] | | |
| 0x14 | OP0SETTLE | 7:0 | | SETTLE[6:0] | | | | | | |
| 0x15 | OP0CAL | 7:0 | CAL[7:0] | | | | | | | |
| 0x16 ... 0x17 | Reserved | | | | | | | | | |
| 0x18 | OP1CTRLA | 7:0 | RUNSTDBY | | | | OUTMODE[1:0] | | EVENTEN | ALWAYSON |
| 0x19 | OP1STATUS | 7:0 | | | | | | | | SETTLED |
| 0x1A | OP1RESMUX | 7:0 | MUXWIP[2:0] | | | MUXBOT[2:0] | | | MUXTOP[1:0] | |
| 0x1B | OP1INMUX | 7:0 | | MUXNEG[2:0] | | | | MUXPOS[2:0] | | |
| 0x1C | OP1SETTLE | 7:0 | | SETTLE[6:0] | | | | | | |
| 0x1D | OP1CAL | 7:0 | CAL[7:0] | | | | | | | |
| 0x1E ... 0x1F | Reserved | | | | | | | | | |
| 0x20 | OP2CTRLA | 7:0 | RUNSTDBY | | | | OUTMODE[1:0] | | EVENTEN | ALWAYSON |
| 0x21 | OP2STATUS | 7:0 | | | | | | | | SETTLED |
| 0x22 | OP2RESMUX | 7:0 | MUXWIP[2:0] | | | MUXBOT[2:0] | | | MUXTOP[1:0] | |
| 0x23 | OP2INMUX | 7:0 | | MUXNEG[2:0] | | | | MUXPOS[2:0] | | |
| 0x24 | OP2SETTLE | 7:0 | | SETTLE[6:0] | | | | | | |
| 0x25 | OP2CAL | 7:0 | CAL[7:0] | | | | | | | |

## 35.5 Register Description

### 35.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

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

**Bit 0 – ENABLE** Enable OPAMP Peripheral
This bit controls whether the OPAMP peripheral is enabled or not.

| Value | Description |
|---|---|
| 0 | The OPAMP peripheral is disabled |
| 1 | The OPAMP peripheral is enabled |

### 35.5.2 Debug Control

**Name:** DBGCTRL
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

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

**Bit 0 – DBGRUN** Run in Debug Mode
This bit controls whether or not the digital interface of the OPAMP peripheral will continue operation in Debug mode when the CPU is halted. The analog portions of the OPAMP peripheral will continue to operate.

| Value | Description |
|---|---|
| 0 | The digital interface of the OPAMP peripheral will not continue operating in Debug mode when the CPU is halted |
| 1 | The digital interface of the OPAMP peripheral will continue operating in Debug mode when the CPU is halted |

### 35.5.3 Timebase

**Name:** TIMEBASE
**Offset:** 0x02
**Reset:** 0x01
**Property:** -

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

**Bits 6:0 – TIMEBASE[6:0]** Timebase
This bit field controls the maximum value of a counter that counts CLK_PER cycles to achieve a time interval equal to or larger than 1 µs. It should be written with one less than the number of CLK_PER cycles that are equal to or larger than 1 µs. This is used for the internal timing of the warmup and settling times.

### 35.5.4 Power Control

**Name:** PWRCTRL
**Offset:** 0x0F
**Reset:** 0x00
**Property:** -

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

**Bit 0 – IRSEL** Input Range Select
This bit selects the op amp input voltage range.

| Value | Description |
|---|---|
| 0 | The op amp input voltage range is rail-to-rail |
| 1 | The op amp input voltage range and power consumption are reduced. See the *Electrical Characteristics* section for more information. |

### 35.5.5 Op Amp n Control A

**Name:** OPnCTRLA
**Offset:** 0x10 + n*0x08 [n=0..2]
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | | | | OUTMODE[1:0] | | EVENTEN | ALWAYSON |
| Access | R/W | | | | R/W | R/W | R/W | R/W |
| Reset | 0 | | | | 0 | 0 | 0 | 0 |

**Bit 7 – RUNSTDBY** Run in Standby Mode
This bit controls whether or not the op amp functions in Standby sleep mode.

| Value | Description |
|---|---|
| 0 | OPn is disabled when in Standby sleep mode, and its output driver is disabled |
| 1 | OPn will continue operating as configured in Standby sleep mode |

**Bits 3:2 – OUTMODE[1:0]** Output Mode
This bit field selects the output mode of the output driver.

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | The output driver for OPn is disabled, but this can be overridden by the DRIVEn event |
| 0x1 | NORMAL | The output driver for OPn is enabled in Normal mode |
| 0x2 – 0x3 | - | Reserved |

**Bit 1 – EVENTEN** Event Enable
This bit enables event reception and generation.

| Value | Description |
|---|---|
| 0 | No events are enabled for OPn |
| 1 | All events are enabled for OPn |

**Bit 0 – ALWAYSON** Always On
This bit controls whether the op amp is always on or not.

| Value | Description |
|---|---|
| 0 | OPn is not always on but can be enabled by the ENABLEn event and disabled by the DISABLEn event |
| 1 | OPn is always on |

### 35.5.6 Op Amp n Status

**Name:** OPnSTATUS
**Offset:** 0x11 + n*0x08 [n=0..2]
**Reset:** 0x00
**Property:** -

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

**Bit 0 – SETTLED** Op Amp has Settled
This bit is cleared when the op amp is waiting for settling related to enabling or configuration changes.
This bit is set when the allowed settling time is finished.

### 35.5.7 Op Amp n Resistor Ladder Multiplexer

**Name:** OPnRESMUX
**Offset:** 0x12 + n*0x08 [n=0..2]
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MUXWIP[2:0] | | | MUXBOT[2:0] | | | MUXTOP[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 7:5 – MUXWIP[2:0]** Multiplexer for Wiper
This bit field selects the resistor ladder wiper (potentiometer) position.

| Value | Name | Description |
|---|---|---|
| 0x0 | WIP0 | R1 = 15R, R2 = 1R |
| 0x1 | WIP1 | R1 = 14R, R2 = 2R |
| 0x2 | WIP2 | R1 = 12R, R2 = 4R |
| 0x3 | WIP3 | R1 = 8R, R2 = 8R |
| 0x4 | WIP4 | R1 = 6R, R2 = 10R |
| 0x5 | WIP5 | R1 = 4R, R2 = 12R |
| 0x6 | WIP6 | R1 = 2R, R2 = 14R |
| 0x7 | WIP7 | R1 = 1R, R2 = 15R |

**Bits 4:2 – MUXBOT[2:0]** Multiplexer for Bottom
This bit field selects the analog signal connected to the bottom resistor in the resistor ladder.

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | Multiplexer off |
| 0x1 | INP | Positive input pin for OPn |
| 0x2 | INN | Negative input pin for OPn |
| 0x3 | DAC | DAC output (DAC and DAC output buffer must be enabled) |
| 0x4 | LINKOUT | OP[n-1] output(1) |
| 0x5 | GND | Ground |
| Other | - | Reserved |

**Note:** When selecting LINKOUT for OP0, MUXBOT is connected to the output of OP2.

**Bits 1:0 – MUXTOP[1:0]** Multiplexer for Top
This bit field selects the analog signal connected to the top resistor in the resistor ladder.

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | Multiplexer off |
| 0x1 | OUT | OPn output |
| 0x2 | VDD | $V_{DD}$ |
| Other | - | Reserved |

### 35.5.8 Op Amp n Input Multiplexer

**Name:** OPnINMUX
**Offset:** 0x13 + n*0x08 [n=0..2]
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | MUXNEG[2:0] | | | | MUXPOS[2:0] | | |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bits 6:4 – MUXNEG[2:0]** Multiplexer for Negative Input
This bit field selects which analog signal is connected to the inverting (-) input of OPn.

| Value | Name | Description |
|---|---|---|
| 0x0 | INN | Negative input pin for OPn |
| 0x1 | WIP | Wiper from OPn's resistor ladder |
| 0x2 | OUT | OPn output (unity gain) |
| 0x3 | DAC | DAC output (DAC and DAC output buffer must be enabled) |
| Other | - | Reserved |

**Bits 2:0 – MUXPOS[2:0]** Multiplexer for Positive Input
This bit field selects which analog signal is connected to the non-inverting (+) input of OPn.

| Value | Name | Description |
|---|---|---|
| 0x0 | INP | Positive input pin for OPn |
| 0x1 | WIP | Wiper from OPn's resistor ladder |
| 0x2 | DAC | DAC output (DAC and DAC output buffer must be enabled) |
| 0x3 | GND | Ground |
| 0x4 | VDDDIV2 | $V_{DD/2}$ |
| 0x5 | LINKOUT | OP[n-1] output (Setting only available for OP1 and OP2) |
| 0x6 | LINKWIP | Wiper from OP0's resistor ladder (Setting only available for OP2) |
| Other | - | Reserved |

### 35.5.9 Op Amp n Settle Timer

**Name:** OPnSETTLE
**Offset:** 0x14 + n*0x08 [n=0..2]
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | SETTLE[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 6:0 – SETTLE[6:0]** Settle Timer
This bit field specifies the number of microseconds allowed for the op amp output to settle. This value, together with the value in the TIMEBASE register, is used by an internal timer to determine when to generate the READYn event and set the SETTLED flag in the OPnSTATUS register.

### 35.5.10 Op Amp n Calibration

**Name:** OPnCAL
**Offset:** 0x15 + n*0x08 [n=0..2]
**Reset:** Fuse readout
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CAL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – CAL[7:0]** Calibration of Input Offset Voltage
This bit field is a calibration value that adjusts the input offset voltage of the op amp. '`0x00`' provides the most negative value of offset adjustment, '`0x80`' provides no offset adjustment, and '`0xFF`' provides the most positive value of offset adjustment.

# 36. ZCD - Zero-Cross Detector

## 36.1 Features

- Detect Zero-Crossings on High-Voltage Alternating Signals
- Only One External Resistor Required
- The Detector Output is Available on a Pin
- The Polarity of the Detector Output can be Inverted
- Interrupt Generation on:
  - Rising edge
  - Falling edge
  - Both edges
- Event Generation:
  - Detector output

## 36.2 Overview

The Zero-Cross Detector (ZCD) detects when an alternating voltage crosses through a threshold voltage near the ground potential. The threshold is the zero-cross reference voltage, $Z_{CPINV}$, and the typical value can be found in the *Electrical Specifications* section of the peripheral.

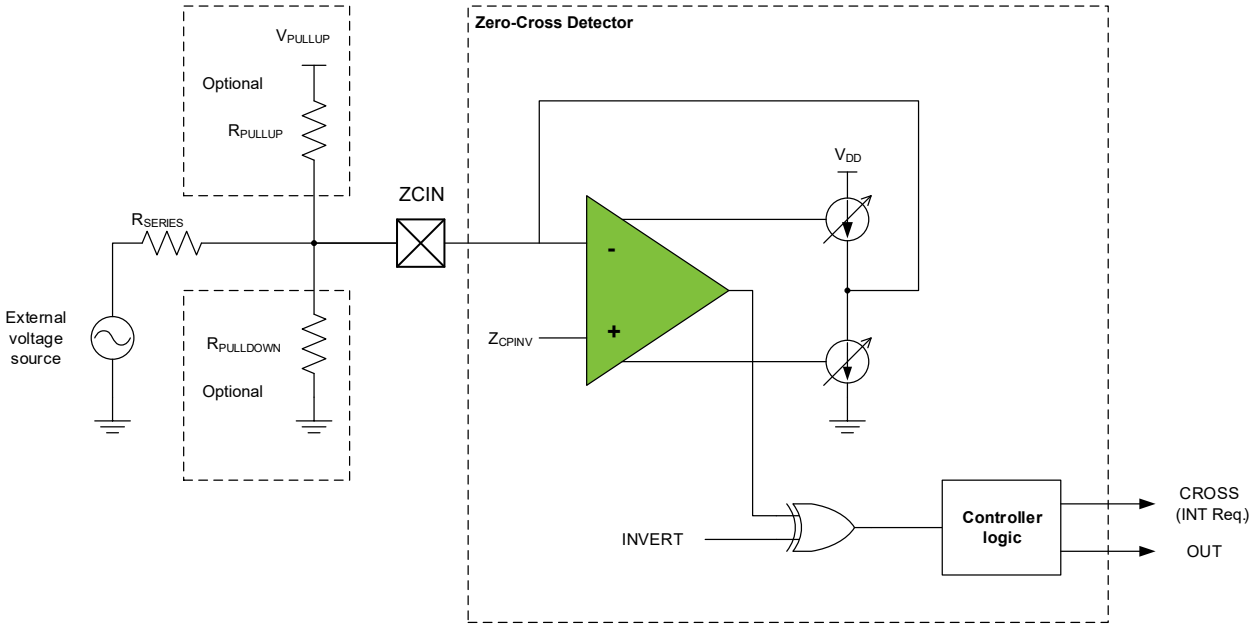
The connection from the ZCD input pin (ZCIN) to the alternating voltage must be made through a series current-limiting resistor ($R_{SERIES}$). The ZCD applies either a current source or sink to the ZCD input pin to maintain a constant voltage on the pin, thereby preventing the pin voltage from forward biasing the ESD protection diodes in the device. When the applied voltage is greater than the reference voltage, the ZCD sinks current. When the applied voltage is less than the reference voltage, the ZCD sources current.

The ZCD can be used when monitoring an alternating waveform for, but not limited to, the following purposes:

- Period Measurement
- Accurate Long-Term Time Measurement
- Dimmer Phase-Delayed Drive
- Low-EMI Cycle Switching

### 36.2.1 Block Diagram

**Figure 36-1. Zero-Cross Detector**




### 36.2.2 Signal Description

| Signal | Description | Type |
|---|---|---|
| ZCIN | Input | Analog |
| OUT | Output | Digital |

## 36.3 Functional Description

### 36.3.1 Initialization

For basic operation, follow these steps:

- Configure the desired input pin in the PORT peripheral as an analog pin with digital input buffer disabled. Internal pull-up and pull-down resistors must also be disabled.
- Optional: Enable the output pin by writing a '1' to the Output Enable (OUTEN) bit in the Control A (ZCDn.CTRLA) register.
- Enable the ZCD by writing a '1' to the ENABLE bit in ZCDn.CTRLA

After the ZCD is enabled, there is a start-up time during which the output of the ZCD may be invalid. The start-up time can be determined by referring to the ZCD electrical characteristics for the device.

### 36.3.2 Operation

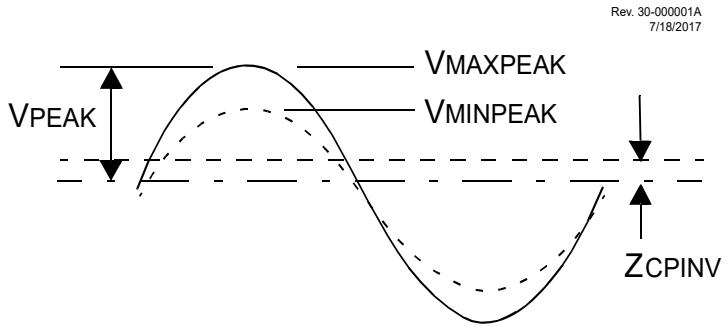
#### 36.3.2.1 External Resistor Selection

The ZCD requires a current-limiting resistor in series ($R_{SERIES}$) with the external voltage source. If the peak amplitude ($V_{PEAK}$) of the external voltage source is expected to be stable, the resistor value must be chosen such that an $I_{ZCD_MAX}/2$ resistor current results in a voltage drop equal to the expected peak voltage. The power rating of the resistor should be at least the mean square voltage divided by the resistor value. (How to handle a peak voltage that varies between a minimum ($V_{MINPEAK}$) and maximum ($V_{MAXPEAK}$) value is described in the section below on **Handling $V_{PEAK}$ Variations**.)

**Equation 36-1. External Resistor**

$$R_{SERIES} = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

**Figure 36-2. External Voltage Source**




#### 36.3.2.2 ZCD Logic Output

The STATE flag in the ZCDn.STATUS register indicates whether the input signal is above or below the reference voltage, $Z_{CPINV}$. By default, the STATE flag is '1' when the input signal is above the reference voltage and '0' when the input signal is below the reference voltage. The polarity of the STATE flag can be reversed by writing the INVERT bit to '1' in the ZCDn.CTRLA register. The INVERT bit will also affect ZCD interrupt polarity.

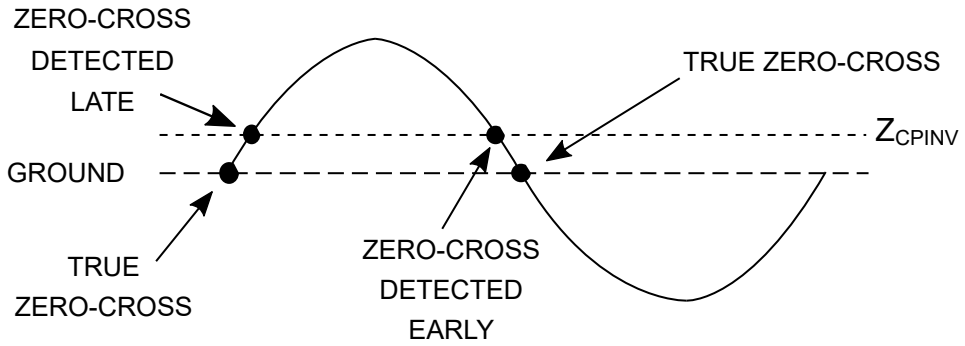
#### 36.3.2.3 Correction for $Z_{CPINV}$ Offset

The actual voltage at which the ZCD switches is the zero-cross reference voltage. Because this reference voltage is slightly offset from the ground, the zero-cross event generated by the ZCD will occur either early or late for the true zero-crossing.

##### 36.3.2.3.1 Correction By Offset Current

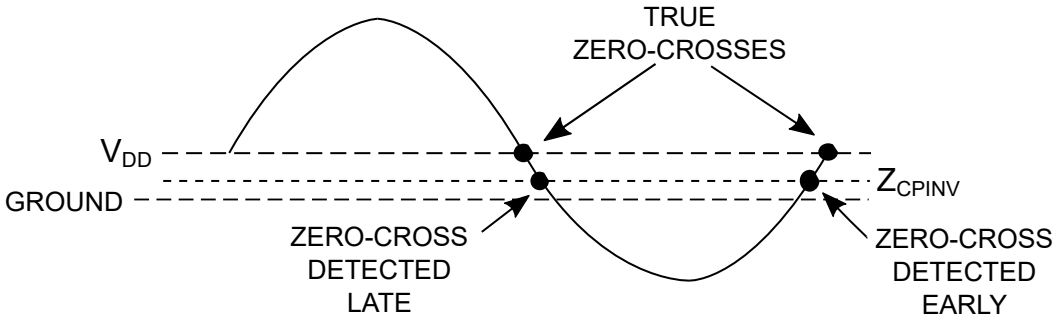
When the alternating waveform is referenced to the ground, as shown in the figure below, the zero-cross is detected too late as the waveform rises and too early as the waveform falls.

**Figure 36-3. Sine Wave Referenced to Ground**




When the waveform is referenced to $V_{DD}$, as shown in the figure below, the zero-cross is detected too late as the waveform falls and too early as the waveform rises.

**Figure 36-4. Sine Wave Referenced to $V_{DD}$**




The actual offset time can be determined for sinusoidal waveforms of a known frequency $f$ using the equations shown below.

**Equation 36-2. ZCD Event Offset**

When the External Voltage source is referenced to ground

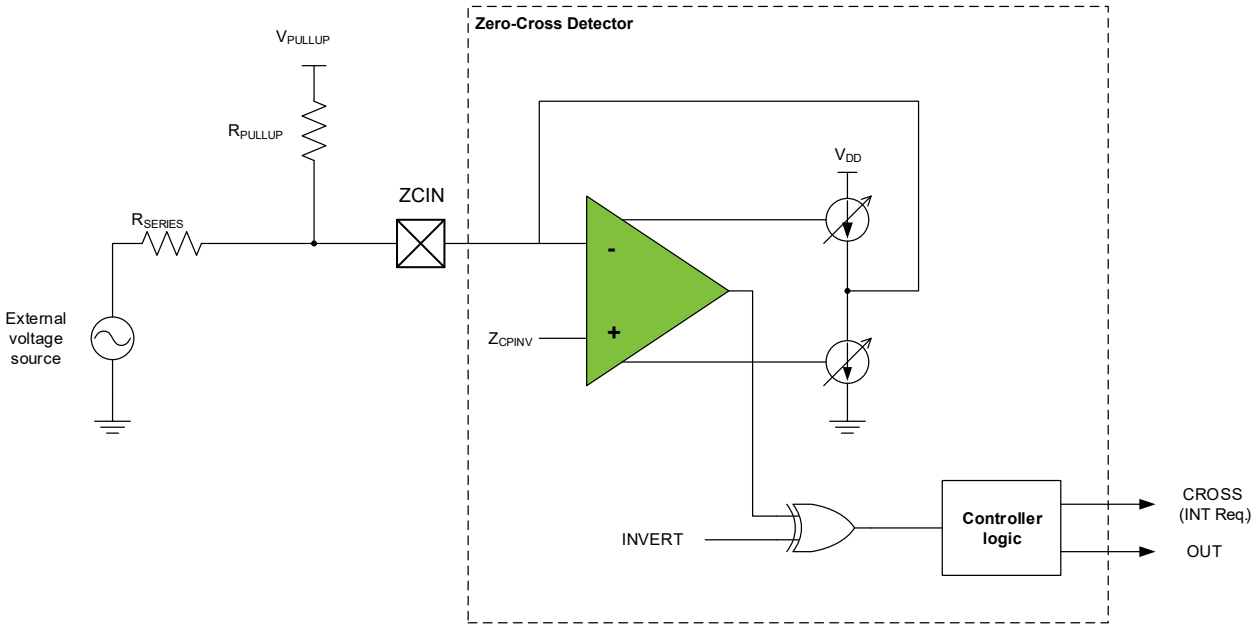
$$T_{offset} = \frac{\sin^{-1}\left(\frac{Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

When the External Voltage source is referenced to $V_{DD}$

$$T_{offset} = \frac{\sin^{-1}\left(\frac{V_{DD} - Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

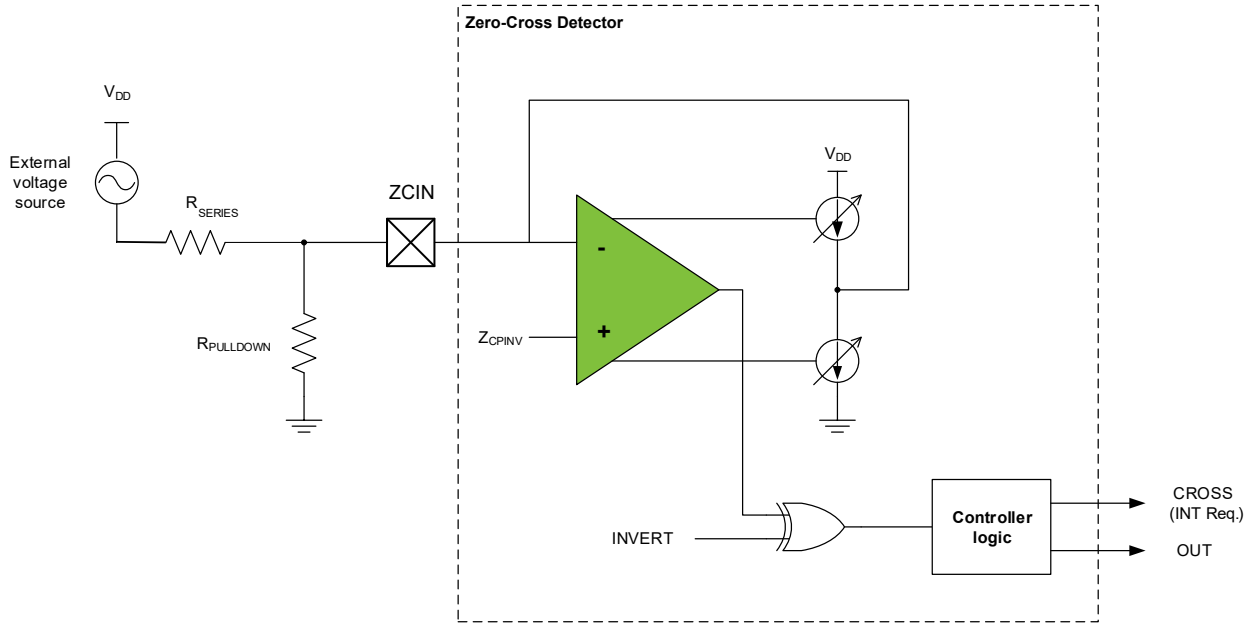
This offset time can be compensated by adding a pull-up or pull-down biasing resistor to the ZCD input pin. A pull-up resistor is used when the external voltage source is referenced to ground, as shown in the figure below.

**Figure 36-5. External Voltage Source Referenced to Ground**




A pull-down resistor is used when the voltage is referenced to $V_{DD}$, as shown in the figure below.

**Figure 36-6. External Voltage Source Referenced to $V_{DD}$**




The resistor adds a bias to the ZCD input pin so that the external voltage source must go to zero to pull the pin voltage to the $Z_{CPINV}$ switching voltage. The pull-up or pull-down value can be determined with the equations shown below.

**Equation 36-3. ZCD Pull-up/Pull-down Resistor**

When the External Voltage source is referenced to ground

$$R_{pullup} = \frac{R_{SERIES}\left(V_{pullup} - Z_{CPINV}\right)}{Z_{CPINV}}$$

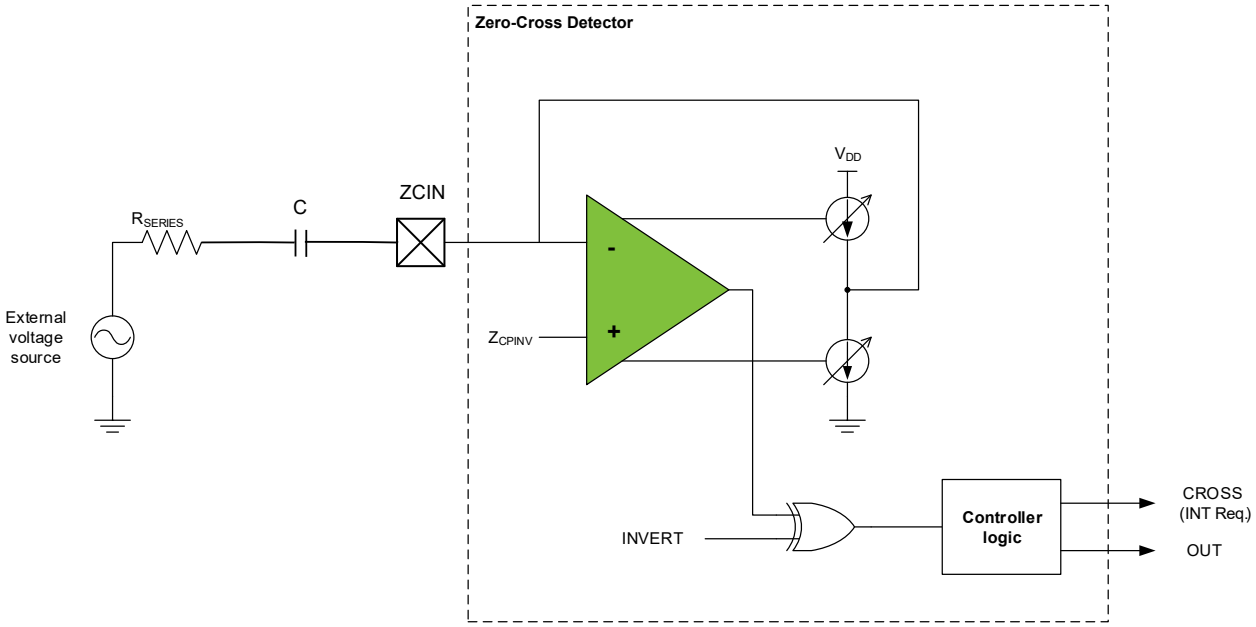
When the External Voltage source is referenced to $V_{DD}$

$$R_{pulldown} = \frac{R_{SERIES}(Z_{CPINV})}{(V_{DD} - Z_{CPINV})}$$

#### 36.3.2.3.2 Correction by AC Coupling

When the external voltage source is sinusoidal, the effects of the $Z_{CPINV}$ offset can be eliminated by isolating the external voltage source from the ZCD input pin with a capacitor in series with the current-limiting resistor, as shown in the figure below.

**Figure 36-7. AC Coupling the ZCD**




The phase shift resulting from the capacitor will cause the ZCD output to switch in advance of the actual zero-crossing event. The phase shift will be the same for both rising and falling zero-crossings, which can be compensated for by either delaying the CPU response to the ZCD switch by a timer or other means or selecting a capacitor value large enough that the phase shift is negligible.

To determine the series resistor and capacitor values for this configuration, start by computing the impedance, Z, to obtain a peak current of $I_{ZCD_MAX}/2$. Next, select a suitably large non-polarized capacitor and compute its reactance, $X_C$, at the external voltage source frequency. Finally, compute the series resistor ($R_{SERIES}$), capacitor peak voltage, and phase shift by using the formulas shown below.

When this technique is used, and the input signal is not present, the ZCD may oscillate. Oscillation can be prevented by connecting the ZCD input pin to ground with a high-value resistor such as 200 kΩ, but this resistor will introduce an offset in the detection of the zero-cross event.

**Equation 36-4. R-C Equations**

$V_{PEAK}$ = External voltage source peak voltage

f = External voltage source frequency

C = Series capacitor

R = Series resistor

$V_C$ = Peak capacitor voltage

Φ = Capacitor-induced zero-crossing phase advance in radians

$T_\Phi$ = Time zero-cross event occurs before actual zero-crossing

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

$$X_C = \frac{1}{2\pi f C}$$

$$R = \sqrt{Z^2 - {X_C}^2}$$

$$V_C = X_C\left(3 \times 10^{-4}\right)$$

$$\Phi = \tan^{-1}\theta\left(\frac{X_C}{R}\right)$$

$$T_\Phi = \frac{\Phi}{2\pi f}$$

**Equation 36-5. R-C Calculation Example**

$$V_{rms} = 120$$

$$V_{PEAK} = V_{rms} \times \sqrt{2} = 169.7$$

$$f = 60\ Hz$$

$$C = 0.1\ \mu F$$

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}} = \frac{169.7}{3 \times 10^{-4}} = 565.7\ k\Omega$$

$$X_C = \frac{1}{2\pi f C} = \frac{1}{2\pi \times 60 \times 10^{-7}} = 26.53\ k\Omega$$

$$R = \sqrt{Z^2 - {X_C}^2} = 565.1\ k\Omega\ (computed)$$

$$R_a = 560\ k\Omega\ (used)$$

$$Z_R = \sqrt{{R_a}^2 + {X_C}^2} = 560.6\ k\Omega$$

$$I_{PEAK} = \frac{V_{PEAK}}{Z_R} = 302.7 \times 10^{-6} A$$

$$V_C = X_C \times I_{PEAK} = 8.0\ V$$

$$\Phi = \tan^{-1}\theta\left(\frac{X_C}{R}\right) = 0.047 radians$$

$$T_\Phi = \frac{\Phi}{2\pi f} = 125.6 \mu s$$

#### 36.3.2.4 Handling $V_{PEAK}$ Variations

If the peak amplitude of the external voltage is expected to vary, the series resistor ($R_{SERIES}$) must be selected to keep the ZCD source and sink currents below the absolute maximum rating of $\pm I_{ZCD_MAX}$ and above a reasonable minimum range. A general rule of thumb for the ZCD is that the maximum peak voltage should be no more than six times the minimum peak voltage. To ensure that the maximum current does not exceed $\pm I_{ZCD_MAX}$ and the minimum is at least $I_{ZCD_MAX}/6$, compute the series resistance, as shown in the equation below. The compensating pull-up or pull-down for this series resistance can be determined using the **ZCD Pull-up/Pull-down Resistor** equations shown earlier, as the pull-up/pull-down resistor value is independent of the peak voltage.

**Equation 36-6. Series Resistor for External Voltage Range**

$$R_{SERIES} = \frac{V_{MAXPEAK} + V_{MINPEAK}}{7 \times 10^{-4}}$$

### 36.3.3 Events

The ZCD can generate the following events:

**Table 36-1. ZCD Event Generator**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Peripheral | Event | | | | |
| ZCDn | OUT | ZCD output level | Level | Asynchronous | Determined by the ZCD output level |

The ZCD has no event inputs. Refer to the Event System (EVSYS) section for more details regarding event types and Event System configuration.

### 36.3.4 Interrupts

**Table 36-2. Available Interrupt Vectors and Sources**

| Name | Vector Description | Conditions |
|---|---|---|
| CROSS | ZCD interrupt | Zero-cross detection as configured by INTMODE in ZCDn.INTCTRL and INVERT in ZCDn.CTRLA |

When a ZCD interrupt condition occurs, the CROSSIF flag is set in the Status (ZCDn.STATUS) register.

ZCD interrupts are enabled or disabled by writing to the INTMODE field in the Interrupt Control (ZCDn.INTCTRL) register.

A ZCD interrupt request is generated when the interrupt source is enabled, and the CROSSIF flag is set. The interrupt request remains active until the interrupt flag is cleared. See the ZCDn.STATUS register description for details on how to clear interrupt flags.

### 36.3.5 Sleep Mode Operation

In Idle sleep mode, the ZCD will continue to operate as normal.

In Standby sleep mode, the ZCD is disabled by default. If the Run in Standby (RUNSTDBY) bit in the Control A (ZCDn.CTRLA) register is written to '1', the ZCD will continue to operate as normal with interrupt generation, event generation, and ZCD output on pin even if CLK_PER is not running in Standby sleep mode.

In Power Down sleep mode, the ZCD is disabled, including its output to pin.

## 36.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CTRLA | 7:0 | RUNSTDBY | OUTEN | | | INVERT | | | ENABLE |
| 0x01 | Reserved | | | | | | | | | |
| 0x02 | INTCTRL | 7:0 | | | | | | | INTMODE[1:0] | |
| 0x03 | STATUS | 7:0 | | | | STATE | | | | CROSSIF |

## 36.5 Register Description

### 36.5.1 Control A

**Name:** CTRLA
**Offset:** 0x00
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNSTDBY | OUTEN | | | INVERT | | | ENABLE |
| Access | R/W | R/W | | | R/W | | | R/W |
| Reset | 0 | 0 | | | 0 | | | 0 |

**Bit 7 – RUNSTDBY** Run in Standby
Writing this bit to '1' will cause the ZCD to remain active when the device enters Standby sleep mode.

**Bit 6 – OUTEN** Output Pin Enable
Writing this bit to '1' connects the OUT signal to a supported pin.

**Bit 3 – INVERT** Invert Enable
Writing this bit to '1' inverts the ZCD output.

**Bit 0 – ENABLE** ZCD Enable
Writing this bit to '1' enables the ZCD.

### 36.5.2 Interrupt Control

**Name:** INTCTRL
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

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

**Bits 1:0 – INTMODE[1:0]** Interrupt Mode
Writing to these bits selects which edge(s) of the ZCD OUT signal will trigger the ZCD interrupt request.

| Value | Name | Description |
|---|---|---|
| 0x0 | NONE | No interrupt |
| 0x1 | RISING | Interrupt on rising OUT signal |
| 0x2 | FALLING | Interrupt on falling OUT signal |
| 0x3 | BOTH | Interrupt on both rising and falling OUT signal |

### 36.5.3 Status

**Name:** STATUS
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | STATE | | | | CROSSIF |
| Access | | | | R | | | | R/W |
| Reset | | | | 0 | | | | 0 |

**Bit 4 – STATE** ZCD State
This bit indicates the current status of the OUT signal from the ZCD. This includes a three-cycle synchronizer delay.

**Bit 0 – CROSSIF** Cross Interrupt Flag
This is the zero-cross interrupt flag. Writing this bit to '1' will clear the interrupt flag. Writing this bit to '0' will have no effect.

# 37. UPDI - Unified Program and Debug Interface

## 37.1 Features

- UPDI One-Wire Interface for External Programming and On-Chip-Debugging (OCD)
  - Uses a dedicated pin of the device for programming
  - No GPIO pins occupied during the operation
  - Asynchronous half-duplex UART protocol towards the programmer
- Programming:
  - Built-in error detection and error signature generation
  - Override of response generation for faster programming
- Debugging:
  - Memory-mapped access to device address space (NVM, RAM, I/O)
  - No limitation on the device clock frequency
  - Unlimited number of user program breakpoints
  - Two hardware breakpoints
  - Support for advanced OCD features
    - Run-time readout of the CPU Program Counter (PC), Stack Pointer (SP) and Status Register (SREG) for code profiling
    - Detection and signalization of the Break/Stop condition in the CPU
    - Program flow control for Run, Stop and Reset debug instructions
  - Nonintrusive run-time chip monitoring without accessing the system registers
  - Interface for reading the result of the CRC check of the Flash on a locked device

## 37.2 Overview

The Unified Program and Debug Interface (UPDI) is a proprietary interface for external programming and OCD of a device.

The UPDI supports programming of Nonvolatile Memory (NVM) space, Flash, EEPROM, fuses, lock bits, and the user row. Some memory-mapped registers are accessible only with the correct access privilege enabled (key, lock bits) and only in the OCD Stopped mode or certain Programming modes. These modes are unlocked by sending the correct key to the UPDI. See the *NVMCTRL - Nonvolatile Memory Controller* section for programming via the NVM controller and executing NVM controller commands.

The UPDI is partitioned into three separate protocol layers: the UPDI Physical (PHY) Layer, the UPDI Data Link (DL) Layer and the UPDI Access (ACC) Layer. The default PHY layer handles bidirectional UART communication over the UPDI pin line towards a connected programmer/debugger and provides data recovery and clock recovery on an incoming data frame in the One-Wire Communication mode. Received instructions and corresponding data are handled by the DL layer, which sets up the communication with the ACC layer based on the decoded instruction. Access to the system bus and memory-mapped registers is granted through the ACC layer.

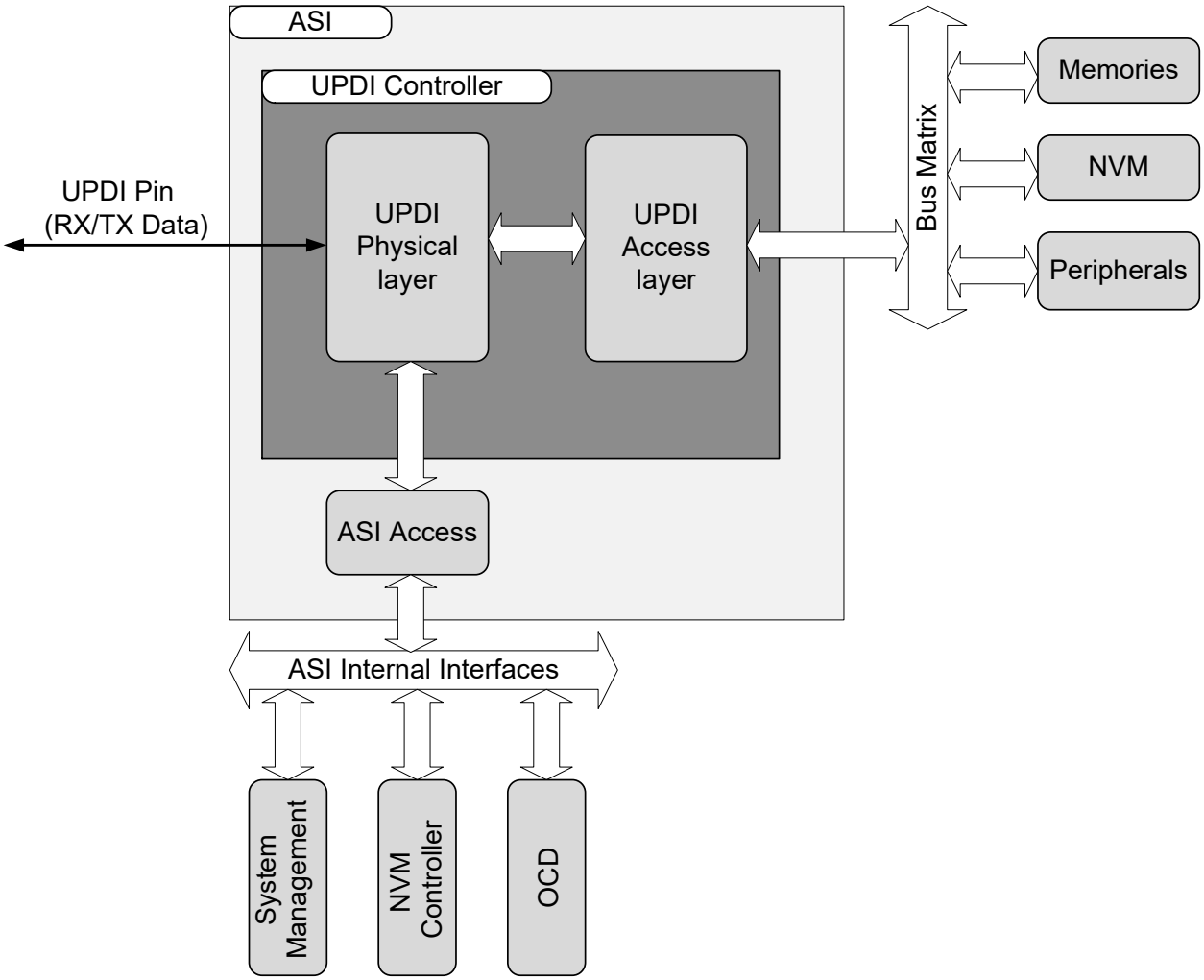
Programming and debugging are done through the PHY layer, which is a one-wire UART based on a half-duplex interface using a dedicated pin for data reception and transmission. The clocking of the PHY layer is done by a dedicated internal oscillator.

The ACC layer is the interface between the UPDI and the connected bus matrix. This layer grants access via the UPDI interface to the bus matrix with memory-mapped access to system blocks such as memories, NVM, and peripherals.

The Asynchronous System Interface (ASI) provides direct interface access to select features in the OCD, NVM, and System Management systems. This gives the debugger direct access to system information without requesting bus access.

### 37.2.1 Block Diagram

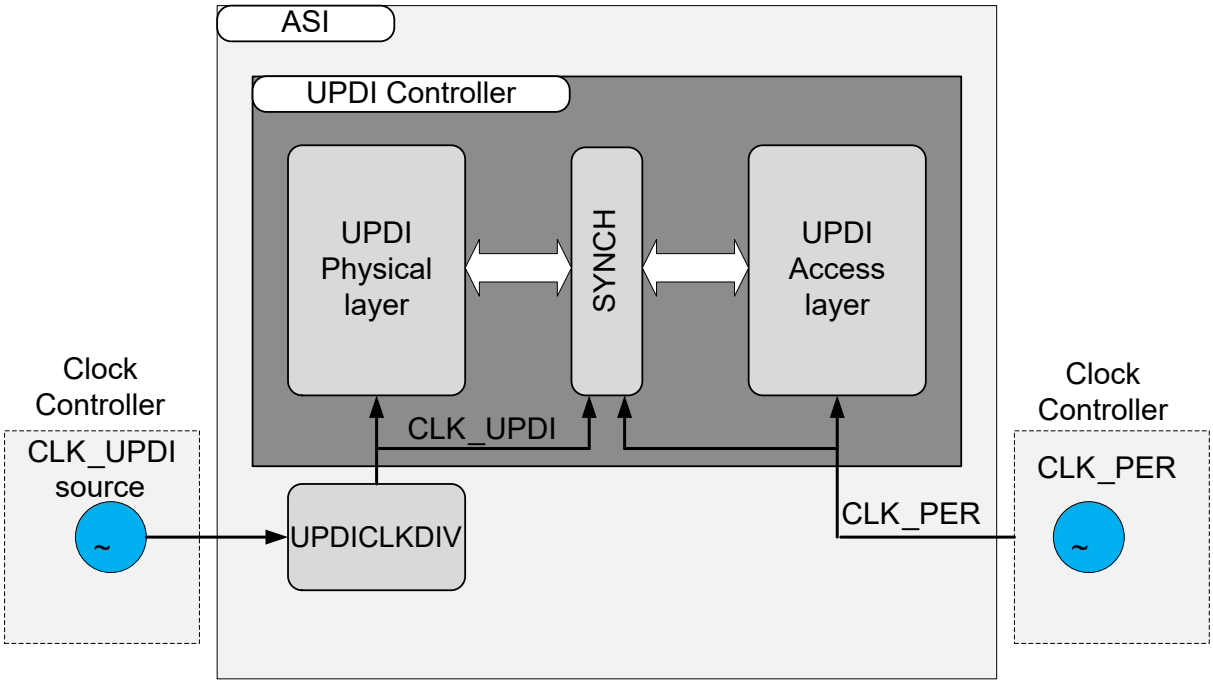
**Figure 37-1. UPDI Block Diagram**




### 37.2.2 Clocks

The PHY layer and the ACC layer can operate on different clock domains. The PHY layer clock is derived from the dedicated internal oscillator, and the ACC layer clock is the same as the peripheral clock. There is a synchronization boundary between the PHY and the ACC layer, which ensures correct operation between the clock domains. The UPDI clock output frequency is selected through the ASI, and the default UPDI clock start-up frequency is 4 MHz after enabling or resetting the UPDI. The UPDI clock frequency can be changed by writing to the UPDI Clock Divider Select (UPDICLKDIV) bit field in the ASI Control A (UPDI.ASI_CTRLA) register.

**Figure 37-2. UPDI Clock Domains**




### 37.2.3 Physical Layer

The PHY layer is the communication interface between a connected programmer/debugger and the device. The main features of the PHY layer can be summarized as follows:

- Dedicated pin on the device with no other function
- Support for UPDI One-Wire Asynchronous mode, using half-duplex UART communication on the UPDI pin
- Internal baud detection, clock and data recovery on the UART frame
- Error detection (parity, clock recovery, frame, system errors)
- Transmission response generation (ACK)
- Generation of error signatures during operation
- Guard time control

### 37.2.4 Pinout Description

The following table shows the functionality of the pin used by the UPDI. See the *I/O Multiplexing* section in the Device Data Sheet for more information about the UPDI physical pin.

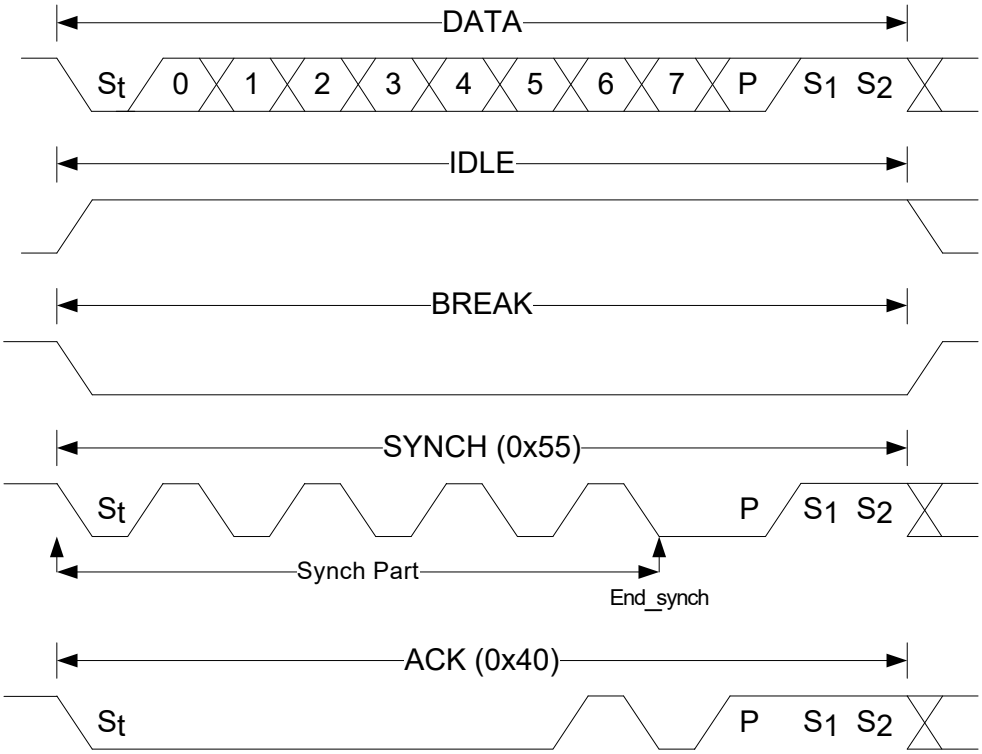
| Function | Pin Name |
|---|---|
| UPDI | UPDI |

## 37.3 Functional Description

### 37.3.1 Principle of Operation

The communication through the UPDI is based on standard UART communication, using a fixed frame format, and automatic baud rate detection for clock and data recovery. In addition to the data frame, several control frames are important to the communication: DATA, IDLE, BREAK, SYNCH, ACK.

**Figure 37-3. Supported UPDI Frame Formats**




| Frame | Description |
|---|---|
| DATA | A DATA frame consists of one Start (St) bit which is always low, eight Data bits, one Parity (P) bit for even parity and two Stop (S1 and S2) bits which are always high. If the Parity bit or Stop bits have an incorrect value, an error will be detected and signalized by the UPDI. The parity bit-check in the UPDI can be disabled by writing to the Parity Disable (PARD) bit in the Control A (UPDI.CTRLA) register, in which case the parity generation from the debugger is ignored. |
| IDLE | This is a special frame that consists of 12 high bits. This is the same as keeping the transmission line in an Idle state. |
| BREAK | This is a special frame that consists of 12 low bits. It is used to reset the UPDI back to its default state and is typically used for error recovery. |
| SYNCH | The SYNCH frame is used by the Baud Rate Generator to set the baud rate for the coming transmission. A SYNCH character is always expected by the UPDI in front of every new instruction, and after a successful BREAK has been transmitted. |
| ACK | The ACK frame is transmitted from the UPDI whenever an `ST` or an `STS` instruction has successfully crossed the synchronization boundary and gained bus access. When an ACK is received by the debugger, the next transmission can start. |

#### 37.3.1.1 UPDI UART

The communication is initiated from the master debugger/programmer side, and every transmission must start with a SYNCH character, which the UPDI can use to recover the transmission baud rate and store this setting for the incoming data. The baud rate set by the SYNCH character will be used for both reception and transmission of the subsequent instruction and data bytes. See the 37.3.3 UPDI Instruction Set section for details on when the next SYNCH character is expected in the instruction stream.

There is no writable Baud Rate register in the UPDI, so the baud rate sampled from the SYNCH character is used for data recovery when sampling the data byte.

The transmission baud rate of the PHY layer is related to the selected UPDI clock, which can be adjusted by writing to the UPDI Clock Divider Select (UPDICLKDIV) bit field in the ASI Control A (UPDI.ASI_CTRLA) register. The receive and transmit baud rates are always the same within the accuracy of the auto-baud.

**Table 37-1. Recommended UART Baud Rate Based on UPDICLKDIV Setting**

| UPDICLKDIV[1:0] | Max. Recommended Baud Rate | Min. Recommended Baud Rate |
|---|---|---|
| `0x0` (32 MHz) | 1.6 Mbps | 0.600 kbps |
| `0x1` (16 MHz) | 0.9 Mbps | 0.300 kbps |
| `0x2` (8 MHz) | 450 kbps | 0.150 kbps |
| `0x3` (4 MHz) - Default | 225 kbps | 0.075 kbps |

The UPDI Baud Rate Generator utilizes fractional baud counting to minimize the transmission error. With the fixed frame format used by the UPDI, the maximum and recommended receiver transmission error limits can be seen in the following table:

**Table 37-2. Receiver Baud Rate Error**

| Data + Parity Bits | $R_{slow}$ | $R_{fast}$ | Max. Total Error [%] | Recommended Max. RX Error [%] |
|---|---|---|---|---|
| 9 | 96.39 | 104.76 | +4.76/-3.61 | +1.5/-1.5 |

### 37.3.1.2 BREAK Character

The BREAK character is used to reset the internal state of the UPDI to the default setting. This is useful if the UPDI enters an Error state due to a communication error or when the synchronization between the debugger and the UPDI is lost.

To ensure that a BREAK is successfully received by the UPDI in all cases, the debugger must send two consecutive BREAK characters. The first BREAK will be detected if the UPDI is in Idle state and will not be detected if it is sent while the UPDI is receiving or transmitting (at a very low baud rate). However, this will cause a frame error for the reception (RX) or a contention error for the transmission (TX), and abort the ongoing operation. The UPDI will then detect the next BREAK successfully.

Upon receiving a BREAK, the UPDI oscillator setting in the ASI Control A (UPDI.ASI_CTRLA) register is reset to the 4 MHz default UPDI clock selection. This changes the baud rate range of the UPDI, according to Table 37-1.

#### 37.3.1.2.1 BREAK in One-Wire Mode

In One-Wire mode, the programmer/debugger and UPDI can be totally out of synch, requiring a worst-case length for the BREAK character to be sure that the UPDI can detect it. Assuming the slowest UPDI clock speed of 4 MHz (250 ns), the maximum length of the 8-bit SYNCH pattern value that can be contained in 16 bits is $65535 \times 250\ \mathrm{ns} = 16.4\ \mathrm{ms/byte} = 16.4\ \mathrm{ms}/8\ \mathrm{bits} = 2.05\ \mathrm{ms/bit}$.

This gives a worst-case BREAK frame duration of 2.05 ms*12 bits ≈ 24.6 ms for the slowest prescaler setting. When the prescaler setting is known, the time of the BREAK frame can be relaxed according to the values from the next table:

**Table 37-3. Recommended BREAK Character Duration**

| UPDICLKDIV[1:0] | Recommended BREAK Character Duration |
|---|---|
| `0x0` (32 MHz) | 3.075 ms |
| `0x1` (16 MHz) | 6.15 ms |
| `0x2` (8 MHz) | 12.30 ms |
| `0x3` (4 MHz) | 24.60 ms |

### 37.3.1.3 SYNCH Character

The SYNCH character has eight bits and follows the regular UPDI frame format. It has a fixed data bit value of '`0x55`'. The SYNCH character has two main purposes:

1. It acts as the enabling character for the UPDI after a disable.
2. It is used by the Baud Rate Generator to set the baud rate for the subsequent transmission. If an invalid SYNCH character is sent, the next transmission will not be sampled correctly.

#### 37.3.1.3.1 SYNCH in One-Wire Mode

The SYNCH character is used before each new instruction. When using the `REPEAT` instruction, the SYNCH character is expected only before the first instruction after `REPEAT`.

The SYNCH is a known character which, through its property of toggling for each bit, allows the UPDI to measure how many UPDI clock cycles are needed to sample the 8-bit SYNCH pattern. The information obtained through the sampling is used to provide Asynchronous Clock Recovery and Asynchronous Data Recovery on reception, and to keep the baud rate of the connected programmer when doing transmit operations.

### 37.3.2 Operation

The UPDI must be enabled before the UART communication can start.

#### 37.3.2.1 UPDI Enabling

The devices have a dedicated UPDI pin with no other function. The enable sequence for the UPDI is device independent and is described in the following paragraphs.

#### 37.3.2.1.1 One-Wire Enable

The UPDI pin has a constant pull-up enable, and by driving the UPDI pin low for more than 200 ns, a connected programmer will initiate the start-up sequence.

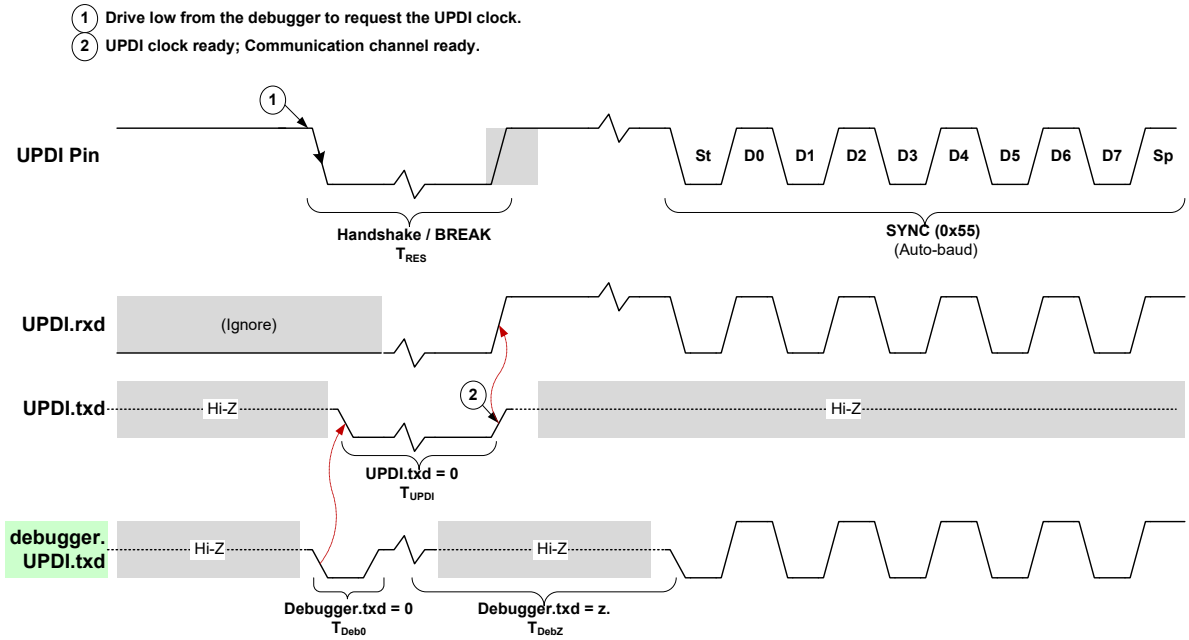
The negative edge transition will cause an edge detector (located in the high-voltage domain if it is in a Multi-Voltage System) to start driving the UPDI pin low, so when the programmer releases the line, it will stay low until the requested UPDI oscillator is ready. The expected arrival time for the clock will depend on the oscillator implementation regarding the accuracy, overshoot and readout of the oscillator calibration. For a Multi-Voltage System, the line will be driven low until the regulator is at the correct level, and the system is powered up with the selected oscillator ready and stable. The programmer must poll the UPDI pin after releasing it the first time to detect when the pin transitions to high again. This transition means that the edge detector has released the pin (pull-up), and the UPDI can receive a SYNCH character. Upon successful detection of the SYNCH character, the UPDI is enabled and will prepare for the reception of the first instruction.

The enable transmission sequence is shown in the next figure, where the active driving periods for the programmer and edge detector are included. The “UPDI pin” waveform shows the pin value at any given time.

The delay given for the edge detector active drive period is a typical start-up time waiting for 256 cycles on a 32 MHz oscillator + the calibration readout. Refer to the *Electrical Characteristics* section for details on the expected start-up times.

**Note:** The first instruction issued after the initial enable SYNCH does not need an extra SYNCH to be sent because the enable sequence SYNCH sets up the Baud Rate Generator for the first instruction.

**Figure 37-4. UPDI Enable Sequence**




To avoid the UPDI from staying enabled if an accidental trigger of the edge detector happens, the UPDI will automatically disable itself and lower its clock request. See the *Disable During Start-up* section for more details.

### 37.3.2.2 UPDI Disabling

#### 37.3.2.2.1 Disable During Start-up

During the enable sequence, the UPDI can disable itself in case of an invalid enable sequence. There are two mechanisms implemented to reset any requests the UPDI has given to the Power Management and set the UPDI to the disabled state. A new enable sequence must then be initiated to enable the UPDI.

***Time-Out Disable***

When the start-up negative edge detector releases the pin after the UPDI has received its clock, or when the regulator is stable and the system has power in a Multi-Voltage system, the default pull-up drives the UPDI pin high. If the programmer does not detect that the pin is high, and does not initiate a transmission of the SYNCH character within 16.4 ms at 4 MHz UPDI clock after the UPDI has released the pin, the UPDI will disable itself.

**Note:** Start-up oscillator frequency is device-dependent. The UPDI will count for 65536 cycles on the UPDI clock before issuing the time-out.

***Incorrect SYNCH pattern***

An incorrect SYNCH pattern is detected if the length of the SYNCH character is longer than the number of samples that can be contained in the UPDI Baud Rate register (overflow), or shorter than the minimum fractional count that can be handled for the sampling length of each bit. If any of these errors are detected, the UPDI will disable itself.

#### 37.3.2.2.2 UPDI Regular Disable

Any programming or debugging session that does not require any specific operation from the UPDI after disconnecting the programmer has to be terminated by writing the UPDI Disable (UPDIDIS) bit in the Control B (UPDI.CTRLB) register, upon which the UPDI will issue a System Reset and disable itself. The Reset will restore the CPU to the Run state, independent of the previous state. It will also lower the UPDI clock request to the system, and reset any UPDI KEYs and settings.

If the disable operation is not performed, the UPDI and the oscillator's request will remain enabled. This causes increased power consumption for the application.

#### 37.3.2.3 UPDI Communication Error Handling

The UPDI contains a comprehensive error detection system that provides information to the debugger when recovering from an error scenario. The error detection consists of detecting physical transmission errors like parity error, contention error, and frame error, to more high-level errors like access time-out error. See the UPDI Error Signature (PESIG) bit field in the Status B (UPDI.STATUSB) register for an overview of the available error signatures.

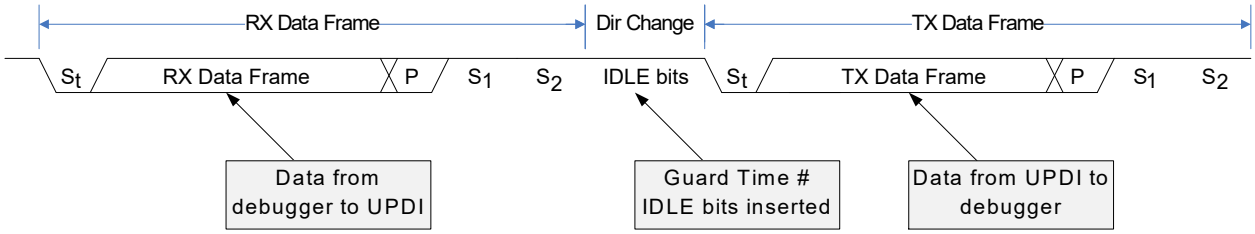
Whenever the UPDI detects an error, it will immediately enter an internal Error state to avoid unwanted system communication. In the Error state, the UPDI will ignore all incoming data requests, except when a BREAK character is received. The following procedure must always be applied when recovering from an Error condition.

1. Send a BREAK character. See the 37.3.1.2 BREAK Character section for recommended BREAK character handling.
2. Send a SYNCH character at the desired baud rate for the next data transfer.
3. Execute a Load Control Status (`LDCS`) instruction to read the UPDI Error Signature (PESIG) bit field in the Status B (UPDI.STATUSB) register and get the information about the occurred error.
4. The UPDI has now recovered from the Error state and is ready to receive the next SYNCH character and instruction.

#### 37.3.2.4 Direction Change

To ensure correct timing for a half-duplex UART operation, the UPDI has a built-in guard time mechanism to relax the timing when changing direction from RX to TX mode. The guard time is represented by Idle bits inserted before the next Start bit of the first response byte is transmitted. The number of Idle bits can be configured through the Guard Time Value (GTVAL) bit field in the Control A (UPDI.CTRLA) register. The duration of each Idle bit is given by the baud rate used by the current transmission.

**Figure 37-5. UPDI Direction Change by Inserting Idle Bits**



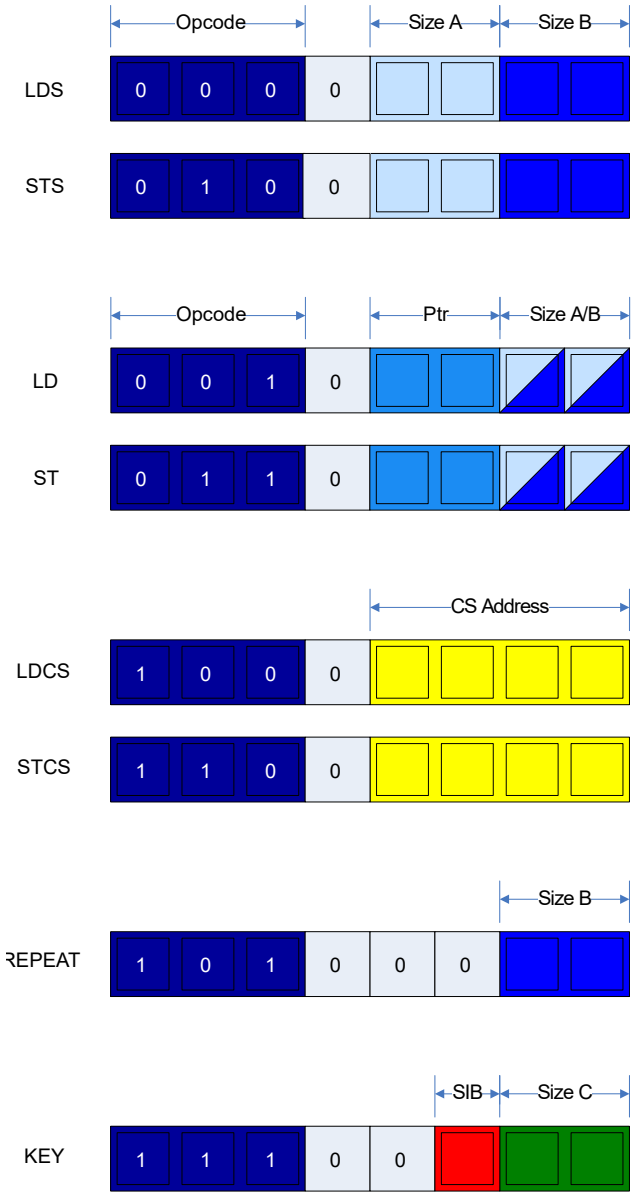

The UPDI guard time is the minimum Idle time that the connected debugger will experience when waiting for data from the UPDI. The maximum Idle time is the same as time-out. The Idle time before a transmission will be more than the expected guard time when the synchronization time plus the data bus accessing time is longer than the guard time.

It is recommended to always use the insertion of minimum two Guard Time bits on the UPDI side, and one guard time cycle insertion from the debugger side.

### 37.3.3 UPDI Instruction Set

The communication through the UPDI is based on a small instruction set. These instructions are part of the UPDI Data Link (DL) layer. The instructions are used to access the UPDI registers, since they are mapped into an internal memory space called “ASI Control and Status (CS) space”, as well as the memory-mapped system space. All instructions are byte instructions and must be preceded by a SYNCH character to determine the baud rate for the communication. See 37.3.1.1 UPDI UART for information about setting the baud rate for the transmission. The following figure gives an overview of the UPDI instruction set.

**Figure 37-6. UPDI Instruction Set Overview**




| OPCODE | | | |
|---|---|---|---|
| 0 | 0 | 0 | LDS |
| 0 | 0 | 1 | LD |
| 0 | 1 | 0 | STS |
| 0 | 1 | 1 | ST |
| 1 | 0 | 0 | LDCS (LDS Control/Status) |
| 1 | 0 | 1 | REPEAT |
| 1 | 1 | 0 | STCS (STS Control/Status) |
| 1 | 1 | 1 | KEY |

| Size A - Address Size | | |
|---|---|---|
| 0 | 0 | Byte - can address 0-255 B |
| 0 | 1 | Word (2 Bytes) - for memories up to 64 KB in size |
| 1 | 0 | 3 Bytes - for memories above 64 KB in size |
| 1 | 1 | Reserved |

| Ptr - Pointer Access | | |
|---|---|---|
| 0 | 0 | *(ptr) |
| 0 | 1 | *(ptr++) |
| 1 | 0 | ptr |
| 1 | 1 | Reserved |

| Size B - Data Size | | |
|---|---|---|
| 0 | 0 | Byte |
| 0 | 1 | Word (2 Bytes) |
| 1 | 0 | Reserved |
| 1 | 1 | Reserved |

CS Address (CS - Control/Status reg.)

| Size C - Key Size | | |
|---|---|---|
| 0 | 0 | 64 bits (8 Bytes) |
| 0 | 1 | 128 bits (16 Bytes) |
| 1 | 0 | Reserved |
| 1 | 1 | Reserved |

| SIB - System Information Block Sel. | |
|---|---|
| 0 | Receive KEY |
| 1 | Send SIB |

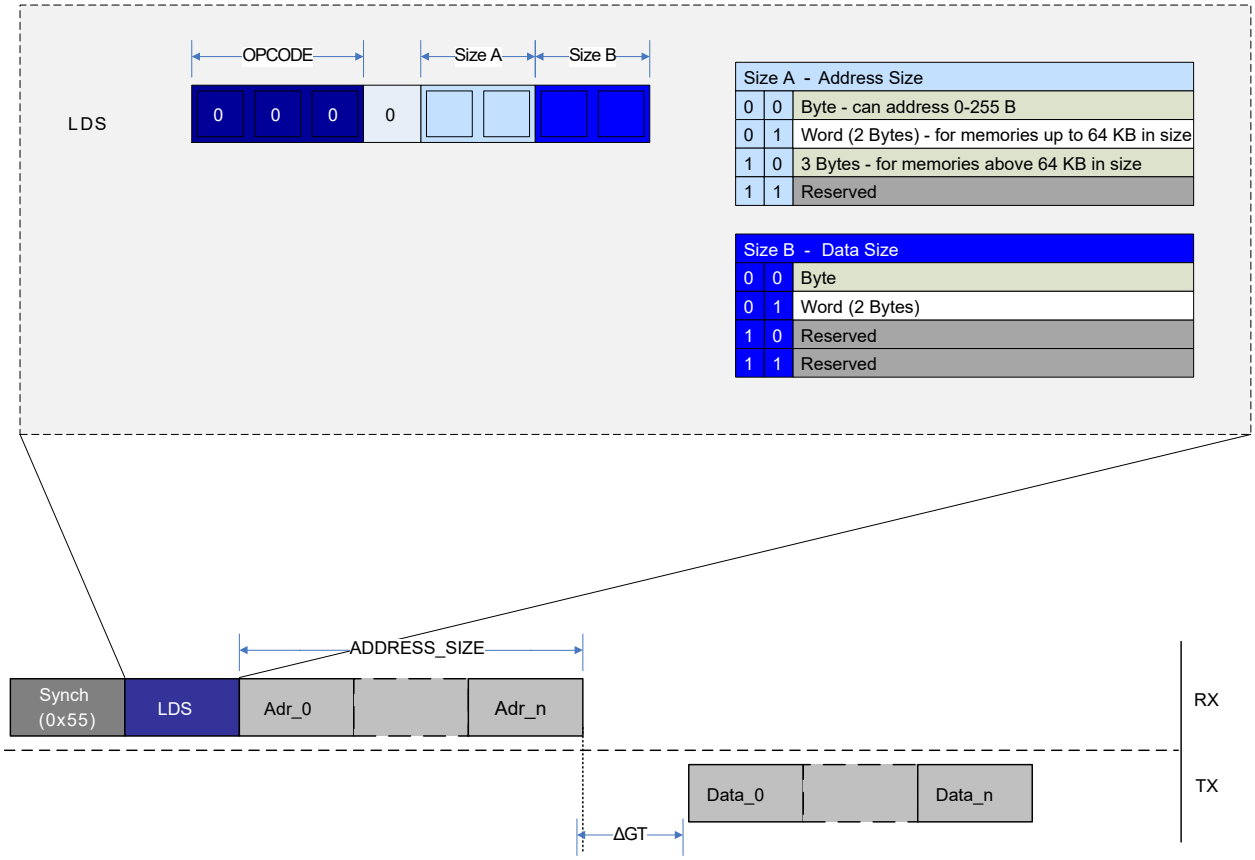
### 37.3.3.1 `LDS` - Load Data from Data Space Using Direct Addressing

The `LDS` instruction is used to load data from the system bus into the PHY layer shift register for serial readout. The `LDS` instruction is based on direct addressing, and the address must be given as an operand to the instruction for the

data transfer to start. The maximum supported size for the address and data is 32 bits. The `LDS` instruction supports repeated memory access when combined with the `REPEAT` instruction.

After issuing the `LDS` instruction, the number of desired address bytes, as indicated by the Size A field followed by the output data size, which is selected by the Size B field, must be transmitted. The output data is issued after the specified Guard Time (GT). When combined with the `REPEAT` instruction, the address must be sent in for each iteration of the repeat, meaning after each time the output data sampling is done. There is no automatic address increment when using `REPEAT` with `LDS`, as it uses a direct addressing protocol.

**Figure 37-7. `LDS` Instruction Operation**



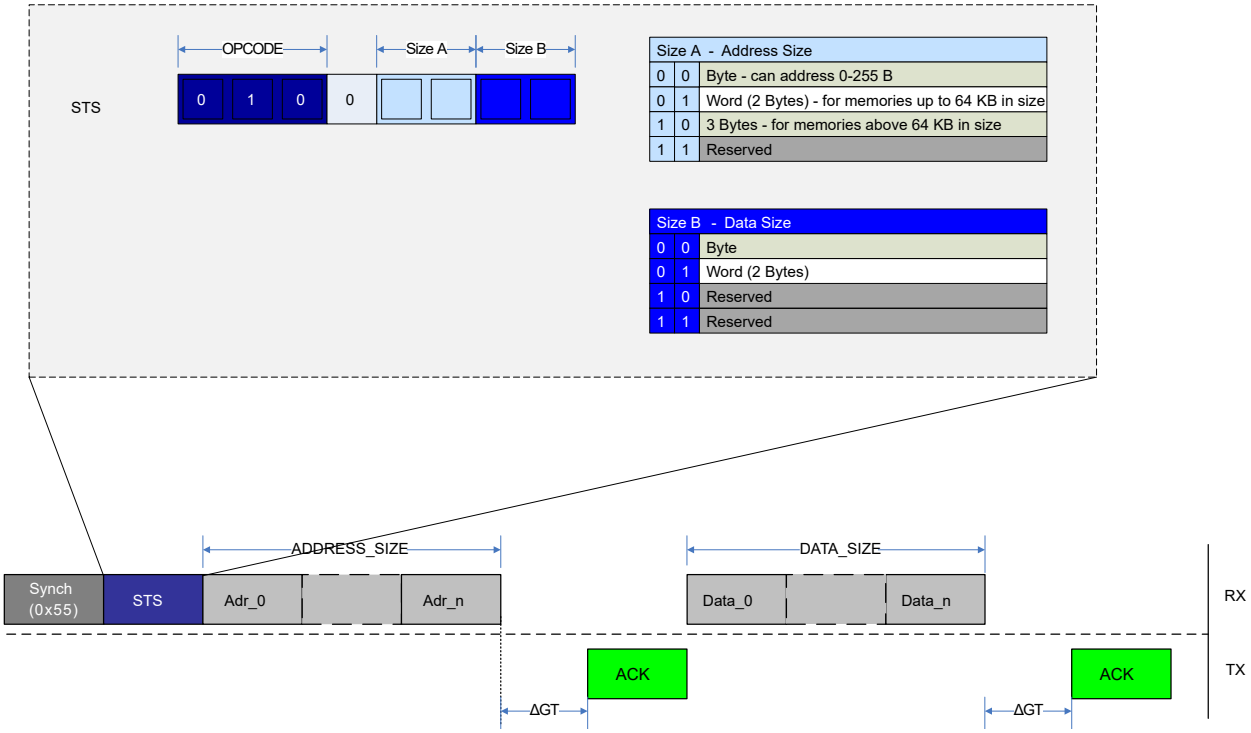

When the instruction is decoded, and the address byte(s) are received as dictated by the decoded instruction, the DL layer will synchronize all required information to the ACC layer, which will handle the bus request and synchronize data buffered from the bus back again to the DL layer. This will create a synchronization delay that must be taken into consideration upon receiving the data from the UPDI.

### 37.3.3.2 `STS` - Store Data to Data Space Using Direct Addressing

The `STS` instruction is used to store data that are shifted serially into the PHY layer shift register to the system bus address space. The `STS` instruction is based on direct addressing, and the address must be given as an operand to the instruction for the data transfer to start. The address is the first set of operands, and data are the second set. The size of the address and data operands are given by the size fields presented in the figure below. The maximum size for both address and data is 32 bits.

The `STS` supports repeated memory access when combined with the `REPEAT` instruction.

**Figure 37-8. STS Instruction Operation**



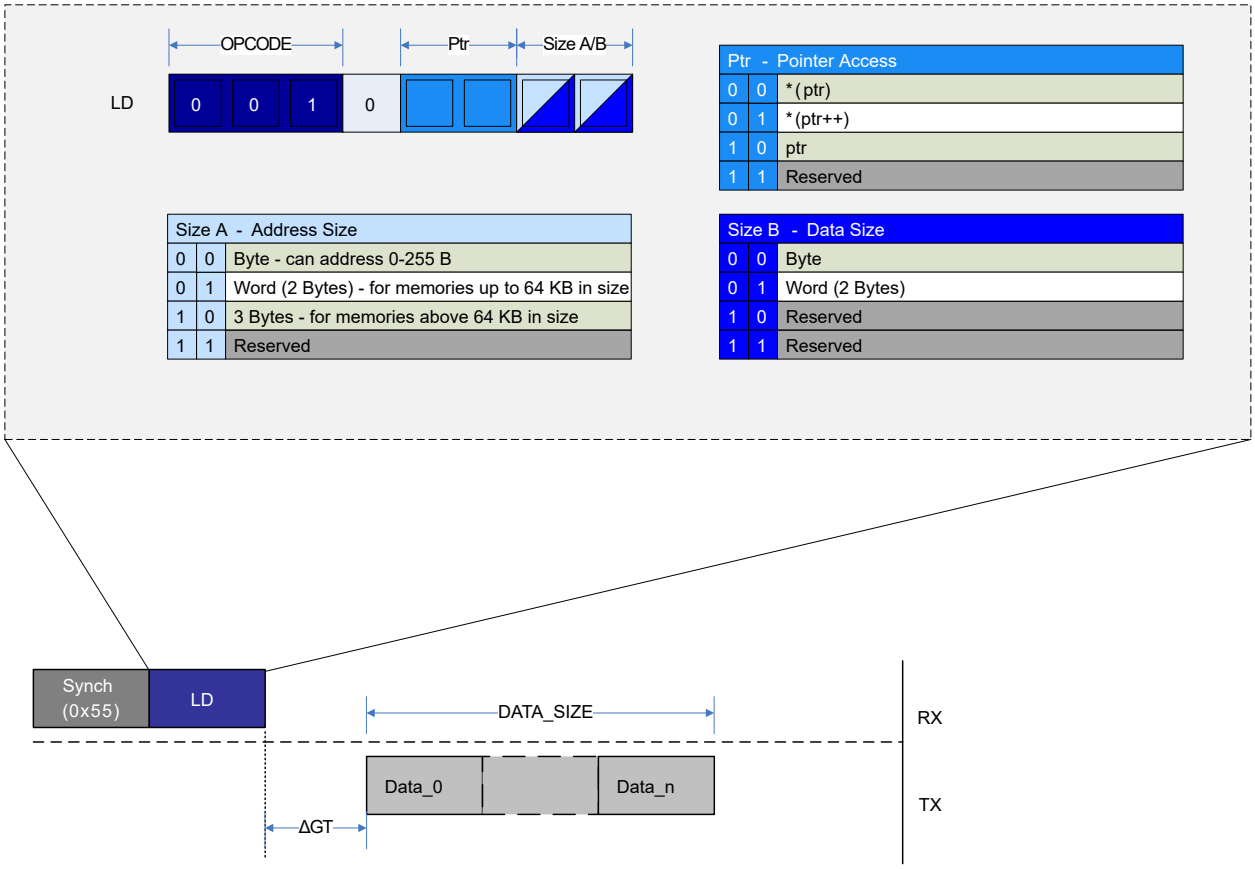

The transfer protocol for an STS instruction is depicted in the above figure, following this sequence:

1. The address is sent.
2. An Acknowledge (ACK) is sent back from the UPDI if the transfer was successful.
3. The number of bytes, as specified in the STS instruction, is sent.
4. A new ACK is received after the data have been successfully transferred.

### 37.3.3.3 LD - Load Data from Data Space Using Indirect Addressing

The LD instruction is used to load data from the data space and into the PHY layer shift register for serial readout. The LD instruction is based on indirect addressing, which means that the Address Pointer in the UPDI needs to be written before the data space read access. Automatic pointer post-increment operation is supported and is useful when the LD instruction is utilized with the REPEAT instruction. It is also possible to do an LD from the UPDI Pointer register. The maximum supported size for address and data load is 32 bits.

**Figure 37-9. LD Instruction Operation**



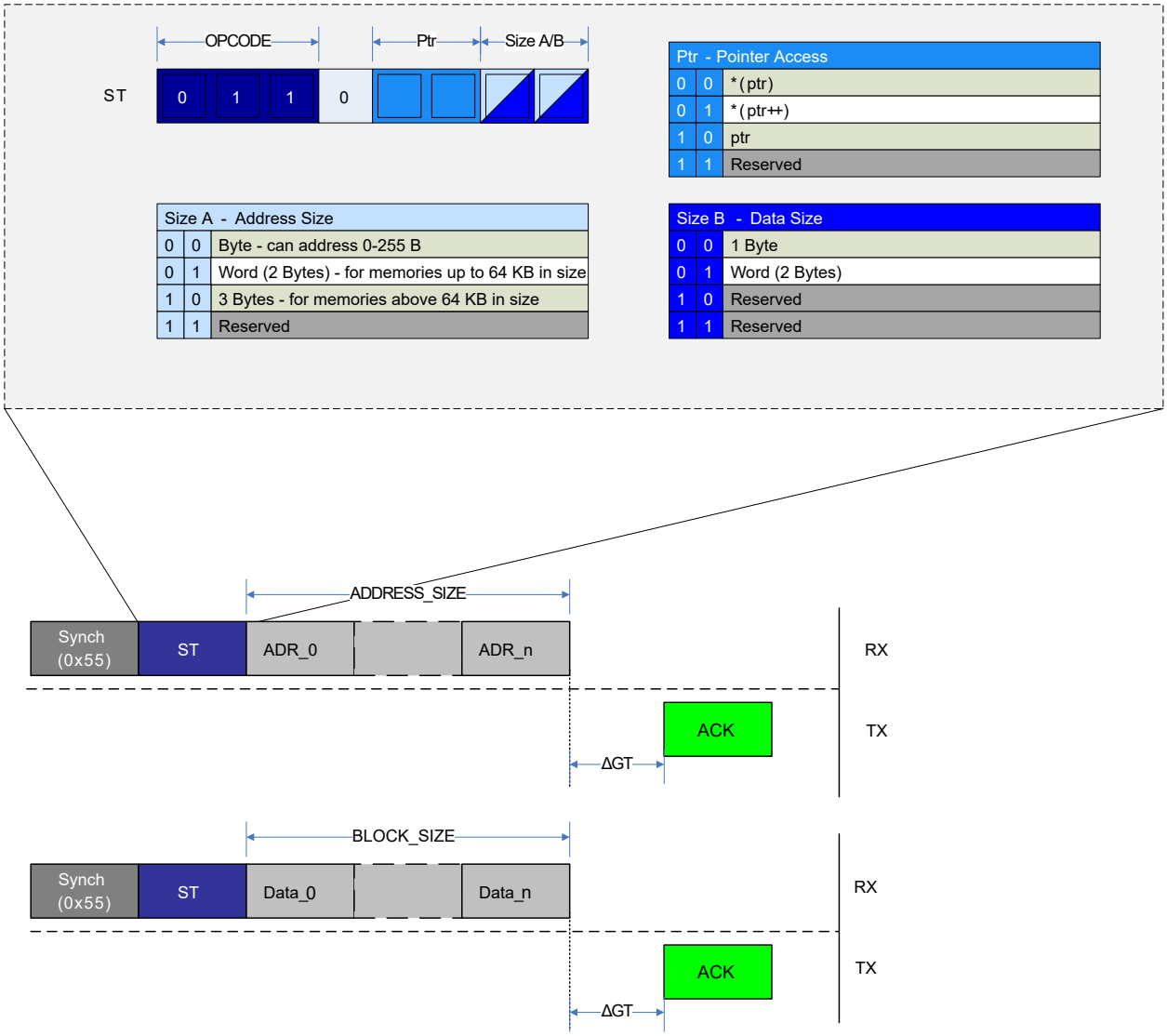

The figure above shows an example of a typical LD sequence, where the data are received after the Guard Time (GT) period. Loading data from the UPDI Pointer register follows the same transmission protocol.

For the LD instruction from the data space, the pointer register must be set up by using an ST instruction to the UPDI Pointer register. After the ACK has been received on a successful Pointer register write, the LD instruction must be set up with the desired DATA SIZE operands. An LD to the UPDI Pointer register is done directly with the LD instruction.

### 37.3.3.4 ST - Store Data from UPDI to Data Space Using Indirect Addressing

The ST instruction is used to store data from the UPDI PHY shift register to the data space. The ST instruction is used to store data that are shifted serially into the PHY layer. The ST instruction is based on indirect addressing, which means that the Address Pointer in the UPDI needs to be written before the data space. The automatic pointer post-increment operation is supported and is useful when the ST instruction is utilized with the REPEAT instruction. The ST instruction is also used to store the UPDI Address Pointer into the Pointer register. The maximum supported size for storing address and data is 32 bits.

**Figure 37-10. ST Instruction Operation**




The figure above gives an example of an `ST` instruction to the UPDI Pointer register and the storage of regular data. A SYNCH character is sent before each instruction. In both cases, an Acknowledge (ACK) is sent back by the UPDI if the `ST` instruction was successful.

To write the UPDI Pointer register, the following procedure has to be followed:

1. Set the PTR field in the `ST` instruction to signature `0x2`.
2. Set the address size (Size A) field to the desired address size.
3. After issuing the `ST` instruction, send Size A bytes of address data.
4. Wait for the ACK character, which signifies a successful write to the Address register.

After the Address register is written, sending data is done in a similarly:

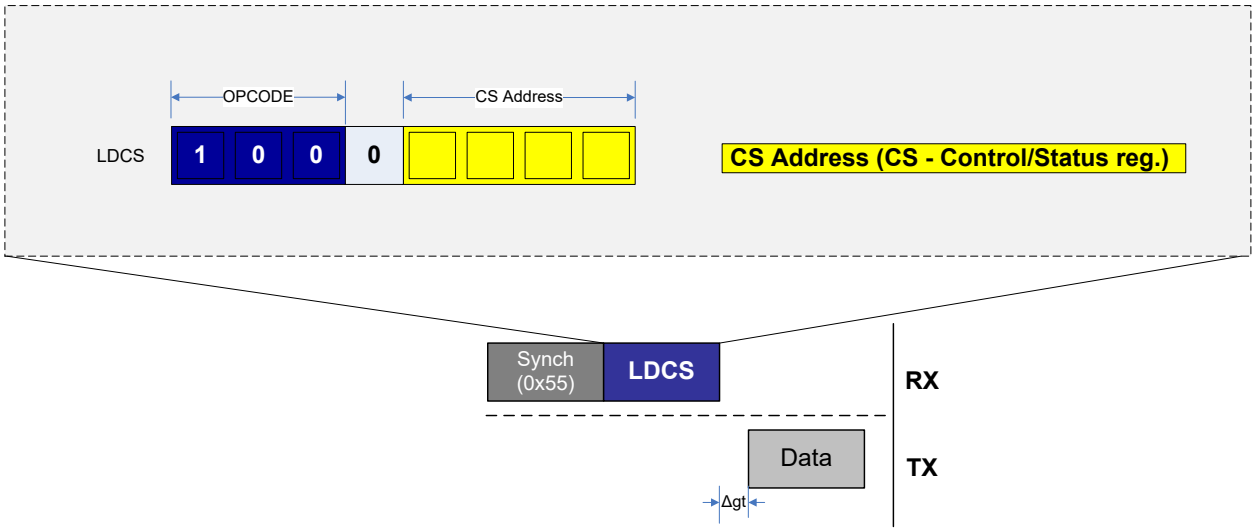
1. Set the PTR field in the `ST` instruction to signature `0x0` to write to the address specified by the UPDI Pointer register. If the PTR field is set to `0x1`, the UPDI pointer is automatically updated to the next address according to the data size Size B field of the instruction after the write is executed.
2. Set the Size B field in the instruction to the desired data size.
3. After sending the `ST` instruction, send Size B bytes of data.
4. Wait for the ACK character, which signifies a successful write to the bus matrix.

When used with the REPEAT instruction, it is recommended to set up the Address register with the start address for the block to be written and use the Pointer Post Increment register to automatically increase the address for each repeat cycle. When using the REPEAT instruction, the data frame of Size B data bytes can be sent after each received ACK.

#### 37.3.3.5 LDCS - Load Data from Control and Status Register Space

The LDCS instruction is used to load serial readout data from the UPDI Control and the Status register space located in the DL layer into the PHY layer shift register. The LDCS instruction is based on direct addressing, where the address is part of the instruction operands. The LDCS instruction can access only the UPDI CS register space. This instruction supports only byte access, and the data size is not configurable.

**Figure 37-11. LDCS Instruction Operation**

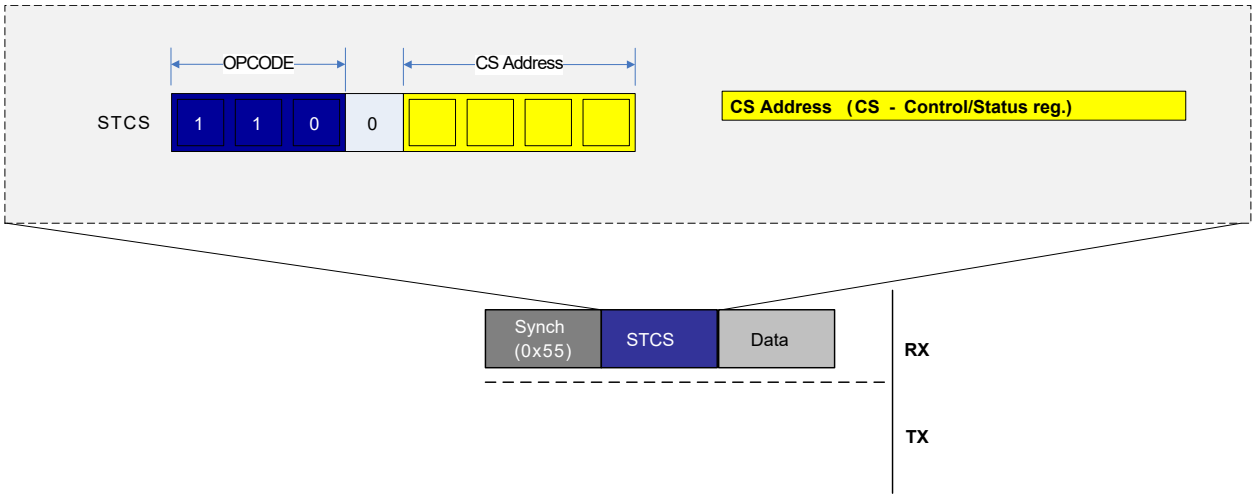



The figure above shows a typical example of LDCS data transmission. A data byte from the LDCS is transmitted from the UPDI after the guard time is completed.

#### 37.3.3.6 STCS (Store Data to Control and Status Register Space)

The STCS instruction is used to store data to the UPDI Control and Status register space. Data are shifted in serially into the PHY layer shift register and written as a whole byte to a selected CS register. The STCS instruction is based on direct addressing, where the address is part of the instruction operand. The STCS instruction can access only the internal UPDI register space. This instruction supports only byte access, and the data size is not configurable.

**Figure 37-12. STCS Instruction Operation**



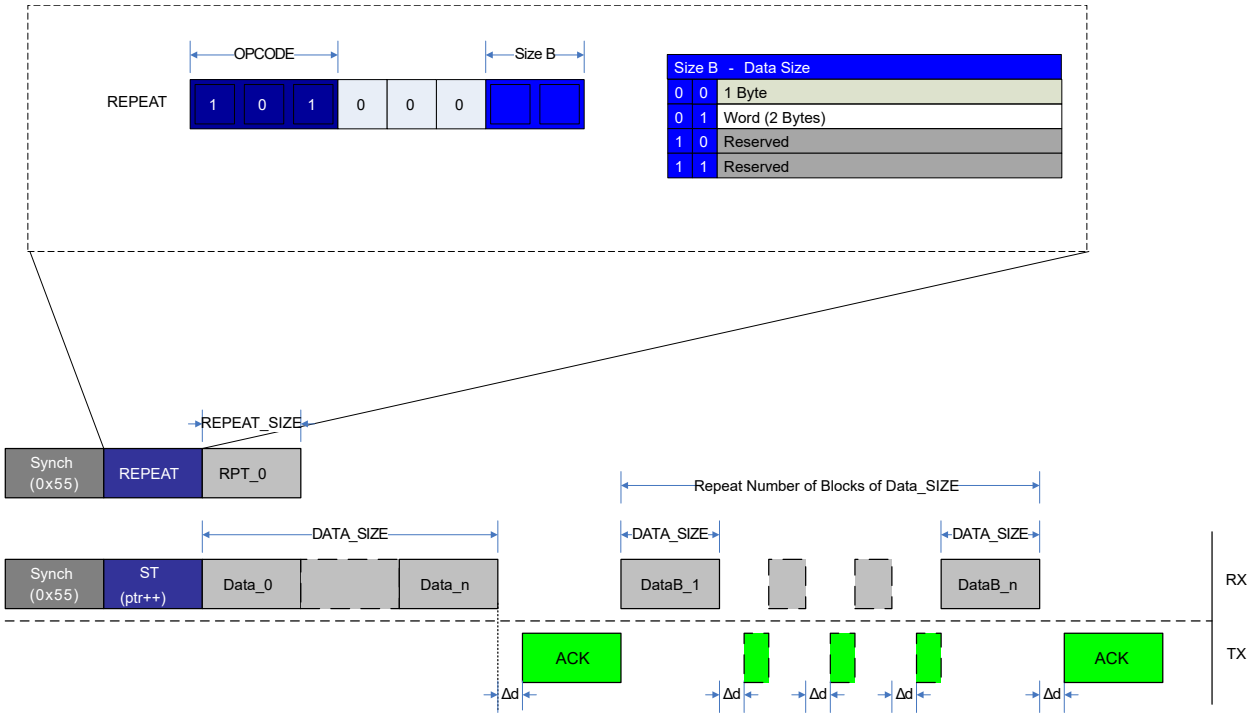
The figure above shows the data frame transmitted after the SYNCH character and the instruction frames. The `STCS` instruction byte can be immediately followed by the data byte. There is no response generated from the `STCS` instruction, as is the case for the `ST` and `STS` instructions.

#### 37.3.3.7 `REPEAT` - Set Instruction Repeat Counter

The `REPEAT` instruction is used to store the repeat count value into the UPDI Repeat Counter register on the DL layer. When instructions are used with `REPEAT`, the protocol overhead for SYNCH and instruction frame can be omitted on all instructions except the first instruction after the `REPEAT` is issued. `REPEAT` is most useful for memory instructions (`LD`, `ST`, `LDS`, `STS`), but all instructions can be repeated, except for the `REPEAT` instruction itself.

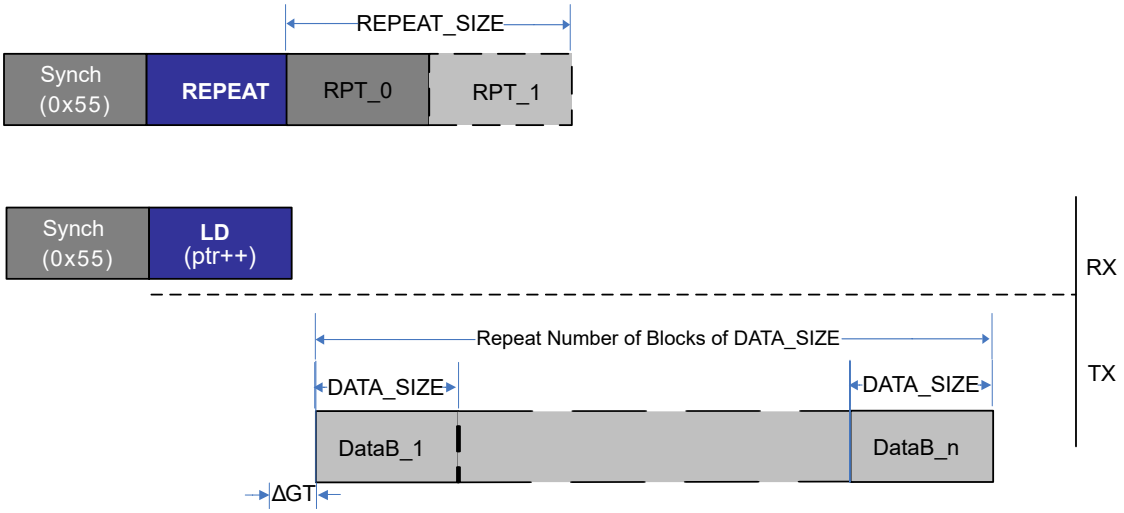
The DATA_SIZE operand field refers to the size of the repeat value. Only up to 255 repeats are supported. The instruction loaded directly after the `REPEAT` instruction will be issued for `RPT_0 + 1` times. If the Repeat Counter register is '`0`', the instruction will run just once. An ongoing repeat can be aborted only by sending a BREAK character.

**Figure 37-13. `REPEAT` Instruction Operation used with `ST` Instruction**




The figure above gives an example of repeat operation with an `ST` instruction using pointer post-increment operation. After the `REPEAT` instruction is sent with `RPT_0 =` ***n***, the first `ST` instruction is issued with SYNCH and instruction frame, while the next ***n*** `ST` instructions are executed by only sending data bytes according to the `ST` operand `DATA_SIZE`, and maintaining the Acknowledge (ACK) handshake protocol.

**Figure 37-14. REPEAT used with LD Instruction**



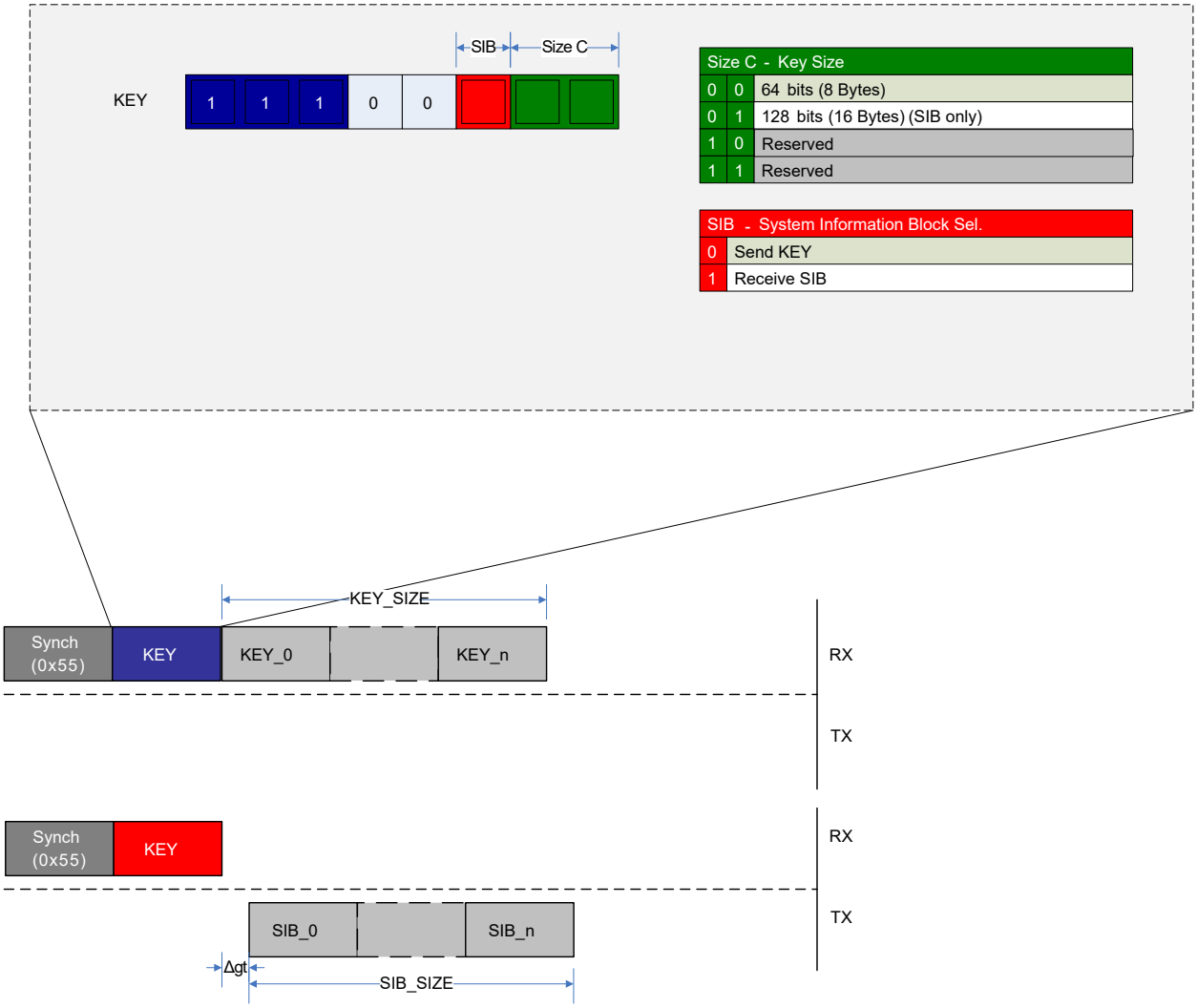

For LD, data will come out continuously after the LD instruction. Note the guard time on the first data block.

If using indirect addressing instructions (LD/ST), it is recommended to always use the pointer post-increment option when combined with REPEAT. The ST/LD instruction is necessary only before the first data block (number of data bytes determined by DATA_SIZE). Otherwise, the same address will be accessed in all repeated access operations. For direct addressing instructions (LDS/STS), the address must always be transmitted as specified in the instruction protocol, before data can be received (LDS) or sent (STS).

#### 37.3.3.8 KEY - Set Activation Key or Send System Information Block

The KEY instruction is used for communicating key bytes to the UPDI or for providing the programmer with a System Information Block (SIB), opening up for executing protected features on the device. See Table 37-4 for an overview of functions that are activated by keys. For the KEY instruction, only a 64-bit key size is supported. The maximum supported size for SIB is 128 bits.

**Figure 37-15. KEY Instruction Operation**




The figure above shows the transmission of a key and the reception of a SIB. In both cases, the Size C (`SIZE_C`) field in the operand determines the number of frames being sent or received. There is no response after sending a `KEY` to the UPDI. When requesting the SIB, data will be transmitted from the UPDI according to the current guard time setting.

### 37.3.4 CRC Checking of Flash During Boot

Some devices support running a CRC check of the Flash contents as part of the boot process. This check can be performed even when the device is locked. The result of this CRC check can be read from the ASI_CRC_STATUS register. Refer to the *CRCSCAN* section in the device data sheet for more information on this feature.

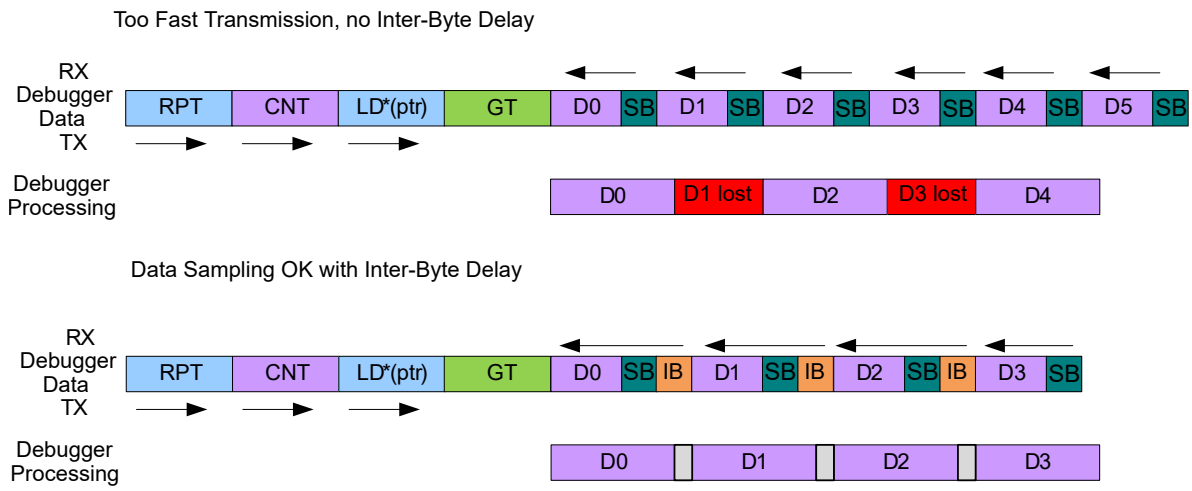
### 37.3.5 Inter-Byte Delay

When performing a multi-byte transfer (`LD` combined with `REPEAT`), or reading out the System Information Block (SIB), the output data will come out in a continuous stream. Depending on the application, on the receiver side, the data might come out too fast, and there might not be enough time for the data to be processed before the next Start bit arrives.

The inter-byte delay works by inserting a fixed number of Idle bits for multi-byte transfers. The reason for adding an inter-byte delay is that there is no guard time inserted when all data is going in the same direction.

The inter-byte delay feature can be enabled by writing a '`1`' to the Inter-Byte Delay Enable (IBDLY) bit in the Control A (UPDI.CTRLA) register. As a result, two extra Idle bits will be inserted between each byte to relax the sampling time for the debugger.

**Figure 37-16. Inter-Byte Delay Example with LD and RPT**




**Notes:**

1. GT denotes the guard time insertion.
2. SB is for Stop bit.
3. IB is the inserted inter-byte delay.
4. The rest of the frames are data and instructions.

## 37.3.6 System Information Block

The System Information Block (SIB) can be read out at any time by setting the SIB bit according to the `KEY` instruction from 37.3.3.8 KEY - Set Activation Key or Send System Information Block. The SIB is always accessible to the debugger, regardless of lock bit settings, and provides a compact form of supplying information about the device and system parameters for the debugger. The information is vital in identifying and setting up the proper communication channel with the device. The output of the SIB is interpreted as ASCII symbols. The key size field must be set to 16 bytes when reading out the complete SIB, and an 8-byte size can be used to read out only the Family_ID. See the figure below for SIB format description and which data are available at different readout sizes.

**Figure 37-17. System Information Block Format**

| 16 | 8 | [Byte][Bits] | Field Name |
|---|---|---|---|
| | | [6:0] [55:0] | Family_ID |
| | | [7][7:0] | Reserved |
| | | [10:8][23:0] | NVM_VERSION |
| | | [13:11][23:0] | OCD_VERSION |
| | | [14][7:0] | RESERVED |
| | | [15][7:0] | DBG_OSC_FREQ |

## 37.3.7 Enabling of Key Protected Interfaces

The access to some internal interfaces and features is protected by the UPDI key mechanism. To activate a key, the correct key data must be transmitted by using the `KEY` instruction, as described in 37.3.3.8 KEY - Set Activation Key or Send System Information Block. The table below describes the available keys and the condition required when doing the operation with the key active.

**Table 37-4. Key Activation Overview**

| Key Name | Description | Requirements for Operation | Conditions for Key Invalidation |
|---|---|---|---|
| Chip Erase | Start NVM chip erase. Clear lock bits | - | UPDI Disable/UPDI Reset |

.....continued

| Key Name | Description | Requirements for Operation | Conditions for Key Invalidation |
|---|---|---|---|
| NVMPROG | Activate NVM Programming | Lock bits cleared. ASI_SYS_STATUS.NVMPROG set | Programming done/UPDI Reset |
| USERROW-Write | Program the user row on the locked device | Lock bits set. ASI_SYS_STATUS.UROWPROG set | Write to key Status bit/ UPDI Reset |

The table below gives an overview of the available key signatures that must be shifted in to activate the interfaces.

**Table 37-5. Key Activation Signatures**

| Key Name | Key Signature (LSB Written First) | Size |
|---|---|---|
| Chip Erase | 0x4E564D4572617365 | 64 bits |
| NVMPROG | 0x4E564D50726F6720 | 64 bits |
| USERROW-Write | 0x4E564D5573267465 | 64 bits |

#### 37.3.7.1 Chip Erase

The following steps must be followed to issue a chip erase:

1. Enter the Chip Erase key by using the `KEY` instruction. See Table 37-5 for the CHIPERASE signature.
2. **Optional:** Read the Chip Erase (CHIPERASE) bit in the ASI Key Status (UPDI.ASI_KEY_STATUS) register to see that the key is successfully activated.
3. Write the signature to the Reset Request (RSTREQ) bit in the ASI Reset Request (UPDI.ASI_RESET_REQ) register. This will issue a System Reset.
4. Write `0x00` to the ASI Reset Request (UPDI.ASI_RESET_REQ) register to clear the System Reset.
5. Read the NVM Lock Status (LOCKSTATUS) bit from the ASI System Status (UPDI.ASI_SYS_STATUS) register.
6. The chip erase is done when LOCKSTATUS bit is '`0`'. If the LOCKSTATUS bit is '`1`', return to step 5.
7. Check the Chip Erase Key Failed (ERASE_FAILED) bit in the ASI System Status (UPDI.ASI_SYS_STATUS) register to verify if the chip erase was successful.
8. If the ERASE_FAILED bit is '`0`', the chip erase was successful.

After a successful chip erase, the lock bits will be cleared, and the UPDI will have full access to the system. Until the lock bits are cleared, the UPDI cannot access the system bus, and only CS-space operations can be performed.

**⚠ CAUTION** During chip erase, the BOD is forced in ON state by writing to the Active (ACTIVE) bit field from the Control A (BOD.CTRLA) register and uses the BOD Level (LVL) bit field from the BOD Configuration (FUSE.BODCFG) fuse and the BOD Level (LVL) bit field from the Control B (BOD.CTRLB) register. If the supply voltage $V_{DD}$ is below that threshold level, the device is unavailable until $V_{DD}$ is increased adequately. See the *BOD* section for more details.

#### 37.3.7.2 NVM Programming

If the device is unlocked, it is possible to write directly to the NVM Controller or to the Flash memory using the UPDI. This will lead to unpredictable code execution if the CPU is active during the NVM programming. To avoid this, the following NVM Programming sequence has to be executed.

1. Follow the chip erase procedure, as described in 37.3.7.1 Chip Erase. If the part is already unlocked, this point can be skipped.
2. Enter the NVMPROG key by using the `KEY` instruction. See Table 37-5 for the NVMPROG signature.
3. **Optional:** Read the NVM Programming Key Status (NVMPROG) bit from the ASI Key Status (UPDI.KEY_STATUS) register to see if the key has been activated.

4. Write the signature to the Reset Request (RSTREQ) bit in the ASI Reset Request (UPDI.ASI_RESET_REQ) register. This will issue a System Reset.
5. Write `0x00` to the ASI Reset Request (UPDI.ASI_RESET_REQ) register to clear the System Reset.
6. Read the NVM Programming Key Status (NVMPROG) bit from the ASI System Status (UPDI.ASI_SYS_STATUS) register.
7. NVM Programming can start when the NVMPROG bit is '`1`'. If the NVMPROG bit is '`0`', return to step 6.
8. Write data to NVM through the UPDI.
9. Write the signature to the Reset Request (RSTREQ) bit in the ASI Reset Request (UPDI.ASI_RESET_REQ) register. This will issue a System Reset.
10. Write `0x00` to the ASI Reset Request (UPDI.ASI_RESET_REQ) register to clear the System Reset.
11. Programming is complete.

#### 37.3.7.3 User Row Programming

The User Row Programming feature allows programming new values to the user row (USERROW) on a locked device. To program with this functionality enabled, the following sequence must be followed:

1. Enter the USERROW-Write key located in Table 37-5 by using the `KEY` instruction. See Table 37-5 for the USERROW-Write signature.
2. **Optional:** Read the User Row Write Key Status (UROWWRITE) bit from the ASI Key Status (UPDI.ASI_KEY_STATUS) register to see if the key has been activated.
3. Write the signature to the Reset Request (RSTREQ) bit in the ASI Reset Request (UPDI.ASI_RESET_REQ) register. This will issue a System Reset.
4. Write `0x00` to the ASI Reset Request (UPDI.ASI_RESET_REQ) register to clear the System Reset.
5. Read the Start User Row Programming (UROWPROG) bit from the ASI System Status (UPDI.ASI_SYS_STATUS) register.
6. User Row Programming can start when the UROWPROG bit is '`1`'. If UROWPROG is '`0`', return to step 5.
7. The data to be written to the User Row must first be written to a buffer in the RAM. The writable area in the RAM has a size of 32 bytes, and it is only possible to write user row data to the first 32 byte addresses of the RAM. Addressing outside this memory range will result in a nonexecuted write. The data will map 1:1 with the user row space when the data is copied into the user row upon completion of the Programming sequence.
8. When all user row data has been written to the RAM, write the User Row Programming Done (UROWDONE) bit in the ASI System Control A (UPDI.ASI_SYS_CTRLA) register.
9. Read the Start User Row Programming (UROWPROG) bit from the ASI System Status (UPDI.ASI_SYS_STATUS) register.
10. The User Row Programming is completed when UROWPROG bit is '`0`'. If UROWPROG bit is '`1`', return to step 9.
11. Write to the User Row Write Key Status (UROWWRITE) bit in the ASI Key Status (UPDI.ASI_KEY_STATUS) register.
12. Write the signature to the Reset Request (RSTREQ) bit in the ASI Reset Request (UPDI.ASI_RESET_REQ) register. This will issue a System Reset.
13. Write `0x00` to the ASI Reset Request (UPDI.ASI_RESET_REQ) register to clear the System Reset.
14. The User Row Programming is complete.

It is not possible to read back data from the RAM in this mode. Only writes to the first 32 bytes of the RAM are allowed.

### 37.3.8 Events

The UPDI can generate the following events:

**Table 37-6. Event Generators in UPDI**

| Generator Name | | Description | Event Type | Generating Clock Domain | Length of Event |
|---|---|---|---|---|---|
| Module | Event | | | | |
| UPDI | SYNCH | SYNCH character | Level | CLK_UPDI | SYNCH char on UPDI pin synchronized to CLK_UPDI |

This event is set on the UPDI clock for each detected positive edge in the SYNCH character, and it is not possible to disable this event from the UPDI.

The UPDI has no event users.

Refer to the *Event System* section for more details regarding event types and Event System configuration.

### 37.3.9 Sleep Mode Operation

The UPDI PHY layer runs independently of all sleep modes, and the UPDI is always accessible for a connected debugger independent of the device's sleep state. If the system enters a sleep mode that turns the system clock off, the UPDI will not be able to access the system bus and read memories and peripherals. When enabled, the UPDI will request the system clock so that the UPDI always has contact with the rest of the device. Thus, the UPDI PHY layer clock is unaffected by the sleep mode's settings. By reading the System Domain in Sleep (INSLEEP) bit in the ASI System Status (UPDI.ASI_SYS_STATUS) register, it is possible to monitor if the system domain is in a sleep mode.

It is possible to prevent the system clock from stopping when going into a sleep mode, by writing to the Request System Clock (CLKREQ) bit in the ASI System Control A (UPDI.ASI_SYS_CTRLA) register. If this bit is set, the system sleep mode state is emulated, and the UPDI can access the system bus and read the peripheral registers even in the deepest sleep modes.

The CLKREQ bit is by default '1' when the UPDI is enabled, which means that the default operation is keeping the system clock in ON state during the sleep modes.

## 37.4 Register Summary

| Offset | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | STATUSA | 7:0 | UPDIREV[3:0] | | | | | | | |
| 0x01 | STATUSB | 7:0 | | | | | | PESIG[2:0] | | |
| 0x02 | CTRLA | 7:0 | IBDLY | | PARD | DTD | RSD | GTVAL[2:0] | | |
| 0x03 | CTRLB | 7:0 | | | | NACKDIS | CCDETDIS | UPDIDIS | | |
| 0x04<br>...<br>0x06 | Reserved | | | | | | | | | |
| 0x07 | ASI_KEY_STATUS | 7:0 | | | UROWWRITE | NVMPROG | CHIPERASE | | | |
| 0x08 | ASI_RESET_REQ | 7:0 | RSTREQ[7:0] | | | | | | | |
| 0x09 | ASI_CTRLA | 7:0 | | | | | | | UPDICLKDIV[1:0] | |
| 0x0A | ASI_SYS_CTRLA | 7:0 | | | | | | | UROWDONE | CLKREQ |
| 0x0B | ASI_SYS_STATUS | 7:0 | | ERASE_FAILED | SYSRST | INSLEEP | NVMPROG | UROWPROG | | LOCKSTATUS |
| 0x0C | ASI_CRC_STATUS | 7:0 | | | | | | CRC_STATUS[2:0] | | |

## 37.5 Register Description

These registers are readable only through the UPDI with special instructions and are not readable through the CPU.

### 37.5.1 Status A

**Name:** STATUSA
**Offset:** 0x00
**Reset:** 0x10
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | UPDIREV[3:0] | | | | | | | |
| Access | R | R | R | R | | | | |
| Reset | 0 | 0 | 0 | 1 | | | | |

**Bits 7:4 – UPDIREV[3:0]** UPDI Revision
This bit field contains the revision of the current UPDI implementation.

### 37.5.2 Status B

**Name:** STATUSB
**Offset:** 0x01
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | PESIG[2:0] | | |
| Access | | | | | | R | R | R |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – PESIG[2:0]** UPDI Error Signature
This bit field describes the UPDI error signature and is set when an internal UPDI Error condition occurs. The PESIG bit field is cleared on a read from the debugger.

**Table 37-7. Valid Error Signatures**

| PESIG[2:0] | Error Type | Error Description |
|---|---|---|
| 0x0 | No error | No error detected (Default) |
| 0x1 | Parity error | Wrong sampling of the Parity bit |
| 0x2 | Frame error | Wrong sampling of the Stop bits |
| 0x3 | Access Layer Time-Out Error | UPDI can get no data or response from the Access layer |
| 0x4 | Clock Recovery error | Wrong sampling of the Start bit |
| 0x5 | - | Reserved |
| 0x6 | Bus error | Address error or access privilege error |
| 0x7 | Contention error | Signalize Driving Contention on the UPDI pin |

### 37.5.3 Control A

**Name:** CTRLA
**Offset:** 0x02
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IBDLY | | PARD | DTD | RSD | GTVAL[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 – IBDLY** Inter-Byte Delay Enable
Writing a '1' to this bit enables a fixed-length inter-byte delay between each data byte transmitted from the UPDI when doing multi-byte LD(S). The fixed length is two IDLE bits.

**Bit 5 – PARD** Parity Disable
Writing a '1' to this bit will disable the parity detection in the UPDI by ignoring the Parity bit. This feature is recommended to be used only during testing.

**Bit 4 – DTD** Disable Time-Out Detection
Writing a '1' to this bit will disable the time-out detection on the PHY layer, which requests a response from the ACC layer within a specified time (65536 UPDI clock cycles).

**Bit 3 – RSD** Response Signature Disable
Writing a '1' to this bit will disable any response signatures generated by the UPDI. This reduces the protocol overhead to a minimum when writing large blocks of data to the NVM space. When accessing the system bus, the UPDI may experience delays. If the delay is predictable, the response signature may be disabled, otherwise loss of data may occur.

**Bits 2:0 – GTVAL[2:0]** Guard Time Value
This bit field selects the guard time value that will be used by the UPDI when the transmission direction switches from RX to TX.

| Value | Description |
|---|---|
| 0x0 | UPDI guard time: 128 cycles (default) |
| 0x1 | UPDI guard time: 64 cycles |
| 0x2 | UPDI guard time: 32 cycles |
| 0x3 | UPDI guard time: 16 cycles |
| 0x4 | UPDI guard time: 8 cycles |
| 0x5 | UPDI guard time: 4 cycles |
| 0x6 | UPDI guard time: 2 cycles |
| 0x7 | Reserved |

### 37.5.4 Control B

**Name:** CTRLB
**Offset:** 0x03
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | NACKDIS | CCDETDIS | UPDIDIS | | |
| Access | | | | R/W | R/W | R/W | | |
| Reset | | | | 0 | 0 | 0 | | |

**Bit 4 – NACKDIS** Disable NACK Response
Writing a '1' to this bit disables the NACK signature sent by the UPDI when a System Reset is issued during ongoing LD(S) and ST(S) operations.

**Bit 3 – CCDETDIS** Collision and Contention Detection Disable
Writing a '1' to this bit disables the contention detection. Writing a '0' to this bit enables the contention detection.

**Bit 2 – UPDIDIS** UPDI Disable
Writing a '1' to this bit disables the UPDI PHY interface. The clock request from the UPDI is lowered, and the UPDI is reset. All the UPDI PHY configurations and keys will be reset when the UPDI is disabled.

### 37.5.5 ASI Key Status

**Name:** ASI_KEY_STATUS
**Offset:** 0x07
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | UROWWRITE | NVMPROG | CHIPERASE | | | |
| Access | | | R/W | R | R | | | |
| Reset | | | 0 | 0 | 0 | | | |

**Bit 5 – UROWWRITE** User Row Write Key Status
This bit is set to '1' if the UROWWRITE key is successfully decoded. This bit must be written as the final part of the user row write procedure to correctly reset the programming session.

**Bit 4 – NVMPROG** NVM Programming Key Status
This bit is set to '1' if the NVMPROG key is successfully decoded. The bit is cleared when the NVM Programming sequence is initiated, and the NVMPROG bit in ASI_SYS_STATUS is set.

**Bit 3 – CHIPERASE** Chip Erase Key Status
This bit is set to '1' if the Chip Erase key is successfully decoded. The bit is cleared by the Reset Request issued as part of the Chip Erase sequence described in the 37.3.7.1 Chip Erase section.

### 37.5.6 ASI Reset Request

**Name:** ASI_RESET_REQ
**Offset:** 0x08
**Reset:** 0x00
**Property:** -

A Reset is signalized to the System when writing the Reset signature to this register.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RSTREQ[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 – RSTREQ[7:0]** Reset Request
The UPDI will not be reset when issuing a System Reset from this register.

| Value | Name | Description |
|---|---|---|
| 0x00 | RUN | Clear Reset condition |
| 0x59 | RESET | Normal Reset |
| Other | | Reset condition is cleared |

### 37.5.7 ASI Control A

**Name:** ASI_CTRLA
**Offset:** 0x09
**Reset:** 0x03
**Property:** -

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

**Bits 1:0 – UPDICLKDIV[1:0]** UPDI Clock Divider Select
This bit field selects the prescaler setting for the UPDI internal oscillator. The default setting after Reset and enable is 4 MHz.

| Value | Description |
|---|---|
| 0x0 | 32 MHz UPDI clock |
| 0x1 | 16 MHz UPDI clock |
| 0x2 | 8 MHz UPDI clock |
| 0x3 | 4 MHz UPDI clock |

### 37.5.8 ASI System Control A

**Name:** ASI_SYS_CTRLA
**Offset:** 0x0A
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | UROWDONE | CLKREQ |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – UROWDONE** User Row Programming Done
This bit must be written when the user row data have been written to the RAM. Writing a '1' to this bit will start the process of programming the user row data to the Flash.
If this bit is written before the user row data is written to the RAM by the UPDI, the CPU will proceed without the written data.
This bit is writable only if the USERROW-Write key is successfully decoded.

**Bit 0 – CLKREQ** Request System Clock
If this bit is written to '1', the ASI is requesting the system clock, independent of the system sleep modes. This makes it possible for the UPDI to access the ACC layer even if the system is in a sleep mode.
Writing a '0' to this bit will lower the clock request.
This bit is set by default when the UPDI is enabled.

### 37.5.9 ASI System Status

**Name:** ASI_SYS_STATUS
**Offset:** 0x0B
**Reset:** 0x01
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | ERASE_FAILED | SYSRST | INSLEEP | NVMPROG | UROWPROG | | LOCKSTATUS |
| Access | | R | R | R | R | R | | R |
| Reset | | 0 | 0 | 0 | 0 | 0 | | 1 |

**Bit 6 – ERASE_FAILED** Chip Erase Key Failed
This bit is set to '1' if the chip erase has failed. This bit is set to '0' on Reset. A Reset held from the ASI_RESET_REQ register will also affect this bit.

**Bit 5 – SYSRST** System Reset Active
When this bit is set to '1', there is an active Reset on the system domain. When this bit is set to '0', the system is not in the Reset state.
This bit is set to '0' on read.
A Reset held from the ASI_RESET_REQ register will also affect this bit.

**Bit 4 – INSLEEP** System Domain in Sleep
When this bit is set to '1', the system domain is in Idle or deeper Sleep mode. When this bit is set to '0', the system is not in any sleep mode.

**Bit 3 – NVMPROG** Start NVM Programming
When this bit is set to '1', NVM Programming can start from the UPDI.
When the UPDI is done, the system must be reset through the UPDI Reset register.

**Bit 2 – UROWPROG** Start User Row Programming
When this bit is set to '1', User Row Programming can start from the UPDI.
When the User Row data have been written to the RAM, the UROWDONE bit in the ASI_SYS_CTRLA register must be written.

**Bit 0 – LOCKSTATUS** NVM Lock Status
When this bit is set to '1', the device is locked. If a chip erase is done, and the lock bits are set to '0', this bit will be read as '0'.

### 37.5.10 ASI CRC Status

**Name:** ASI_CRC_STATUS
**Offset:** 0x0C
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | CRC_STATUS[2:0] | | |
| Access | | | | | | R | R | R |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – CRC_STATUS[2:0]** CRC Execution Status
This bit field signalizes the status of the CRC conversion. This bit field is one-hot encoded.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLED | Not enabled |
| 0x1 | BUSY | CRC enabled, busy |
| 0x2 | PASS | CRC enabled, done with PASS signature |
| 0x4 | FAIL | CRC enabled, done with FAILED signature |

# 38. Instruction Set Summary

The instruction set summary can be found as part of the *AVR Instruction Set Manual*, located at www.microchip.com/DS40002198. Refer to the CPU version called AVRxt, for details regarding the devices documented in this data sheet.

# 39. Electrical Characteristics

## 39.1 Disclaimer

All typical values are measured at T = 25°C and $V_{DD}$ = $V_{DDIO2}$ = 3.0V unless otherwise specified. All minimum and maximum values are valid across operating temperature and voltage unless otherwise specified.

Typical values given should be considered for design guidance only, and actual part variation around these values is expected.

## 39.2 Absolute Maximum Ratings

Stresses beyond those listed in this section 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 39-1. Absolute Maximum Ratings**

| Parameter | Condition | Rating | Units |
|---|---|---|---|
| Ambient temperature under bias | | -40 to +125 | °C |
| Storage temperature | | -65 to +150 | °C |
| **Voltage on Pins With Respect to GND** | | | |
| • On the $V_{DD}$ pin | | -0.3 to +6.5 | V |
| • On the $V_{DDIO2}$ pin | | -0.3 to +6.5 | V |
| • On the $\overline{RESET}$ pin | | -0.3 to ($V_{DD}$ + 0.3) | V |
| • On all other pins | | -0.3 to ($V_{DD}$ + 0.3) | V |
| **Maximum Current** | | | |
| • On the GND pin(1) | -40°C ≤ $T_A$ ≤ +85°C | 350 | mA |
| | +85°C < $T_A$ ≤ +125°C | 120 | mA |
| • On the $V_{DD}$ pin(1) | -40°C ≤ $T_A$ ≤ +85°C | 350 | mA |
| | +85°C < $T_A$ ≤ +125°C | 120 | mA |
| • On the $V_{DDIO2}$ pin(1) | -40°C ≤ $T_A$ ≤ +85°C | 350 | mA |
| | +85°C < $T_A$ ≤ +125°C | 120 | mA |
| • On any standard I/O pin | | ±50 | mA |
| Clamp current, $I_K$ ($V_{PIN}$ < 0 or $V_{PIN}$ > $V_{DD}$) | | ±20 | mA |
| Total power dissipation(2) | | 800 | mW |

**Note:**

1. The maximum current rating requires even load distribution across I/O pins. The maximum current rating may be limited by the device package power dissipation characterizations, see the *Thermal Characteristics* section to calculate device specifications.
2. Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OI} \times I_{OL})$

## 39.3 Standard Operating Conditions

The device must operate within the ratings listed in this section for all other electrical characteristics and typical characteristics of the device to be valid.

**Table 39-2. General Operating Conditions**

| | |
|---|---|
| **Operating Voltage** | $V_{DDMIN} \le V_{DD} \le V_{DDMAX}$ |
| **Operating Temperature** | $T_{A_MIN} \le T_A \le T_{A_MAX}$ |

The standard operating conditions for any device, are defined as:

**Table 39-3. Standard Operating Conditions**

| Parameter | | Ratings | Units |
|---|---|---|---|
| **$V_{DD}$ — Operating Supply Voltage(1)** | | | |
| Industrial and Extended temperature | $V_{DDMIN}$ | +1.8 | V |
| | $V_{DDMAX}$ | +5.5 | V |
| **$T_A$ — Operating Ambient Temperature Range** | | | |
| Industrial temperature | $T_{A_MIN}$ | -40 | °C |
| | $T_{A_MAX}$ | +85 | °C |
| Extended temperature | $T_{A_MIN}$ | -40 | °C |
| | $T_{A_MAX}$ | +125 | °C |

**Note:**
1. Refer to section 39.4.1 Supply Voltage.

## 39.4 DC Characteristics

### 39.4.1 Supply Voltage

**Table 39-4. Supply Voltage**

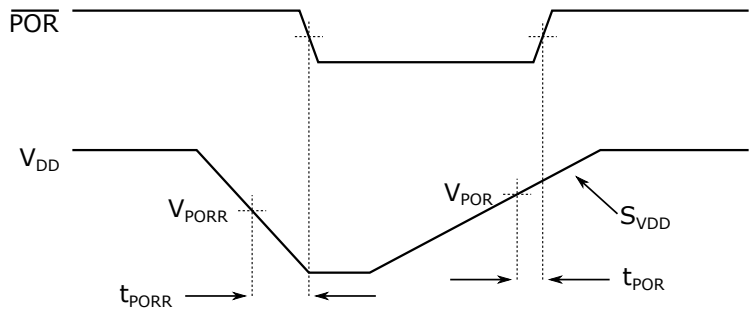
| Symbol | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|
| **Supply Voltage(1)(2)** | | | | | |
| $V_{DD}$ | 1.8 | — | 5.5 | V | |
| $V_{DDIO2}$ | 1.62 | — | 5.5 | V | |
| $S_{VDD}$ | — | — | 250 | V/ms | $1.8V < V_{DD} < 5.5V$ |
| **RAM Data Retention(3)** | | | | | |
| $V_{DR}$ | 1.7 | — | — | V | Device in Power-Down mode |
| **Power-on Reset Release(4)** | | | | | |
| $V_{POR}$ | — | 1.6 | — | V | BOD disabled(2) |
| $t_{POR}$ | — | 1 | — | µs | BOD disabled(2) |
| **Power-on Reset Rearm(4)** | | | | | |
| $V_{PORR}$ | — | 1.25 | — | V | BOD disabled(2) |
| $t_{PORR}$ | — | 2.7 | — | µs | BOD disabled(2) |
| **$V_{DD}$ Slope to Ensure Internal Power-on Reset Signal(4)** | | | | | |
| $S_{VDD}$ | 0.05 | — | — | V/ms | BOD disabled(2) |

| .....continued | | | | | |
|---|---|---|---|---|---|
| **Symbol** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |

**Notes:**

1. During Chip Erase, the Brown-out Detector (BOD) configured with BODLEVEL0 is forced ON. If the supply voltage $V_{DD}$ is below $V_{BOD}$ for BODLEVEL0, the erase attempt will fail.
2. Refer to section 39.5.2 Reset Controller Specifications for BOD trip point information.
3. This is the limit to which $V_{DD}$ can be lowered in sleep mode without losing RAM data.
4. Refer to Figure 39-1.

**Figure 39-1. $\overline{POR}$ and $\overline{PORR}$ with Slow Rising $V_{DD}$**




**Note:**

- When $\overline{POR}$ is low, the device is held in Reset

## 39.4.2 Power Consumption

**Table 39-5. Device Power Consumption**

**Operating Conditions:**

- **$V_{DD}$ = $V_{DDIO2}$ = 3V**
- **$T_A$ = 25°C**
- **System power consumption measured with peripherals disabled and I/O ports driven low with inputs disabled**

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $I_{DD}$ | Active power consumption | — | 4.1 | — | mA | OSCHF = 24 MHz |
| | | — | 1.0 | — | mA | OSCHF = 4 MHz |
| | | — | 7.0 | — | µA | OSC32K = 32.768 kHz |
| | | — | 3.8 | — | mA | EXTCLK = 24 MHz |
| | | — | — | — | µA | EXTCLK = 4 MHz |
| | | — | 9.0 | — | µA | XOSC32K = 32.768 kHz (High Power) |
| | | — | 7.5 | — | µA | XOSC32K = 32.768 kHz (Low Power) |

.....continued

**Operating Conditions:**
- $V_{DD} = V_{DDIO2} = 3V$
- $T_A = 25°C$
- System power consumption measured with peripherals disabled and I/O ports driven low with inputs disabled

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $I_{DD_IDLE}$ | Idle power consumption | — | 1.9 | — | mA | OSCHF = 24 MHz |
| | | — | 580 | — | µA | OSCHF = 4 MHz |
| | | — | 4.0 | — | µA | OSC32K = 32.768 kHz |
| | | — | 1.7 | — | mA | EXTCLK = 24 MHz |
| | | — | — | — | µA | EXTCLK = 4 MHz |
| | | — | 7.5 | — | µA | XOSC32K = 32.768 kHz (High Power) |
| | | — | 6.0 | — | µA | XOSC32K = 32.768 kHz (Low Power) |
| $I_{DD_STDBY}$(1) | Standby power consumption | — | 3.2 | — | µA | RTC running at 1.024 kHz from XOSC32K (CL = 9 pF, High Power) |
| | | — | 1.6 | — | µA | RTC running at 1.024 kHz from XOSC32K (CL = 9 pF, Low Power) |
| | | — | 1.2 | — | µA | RTC running at 1.024 kHz from OSC32K |
| $I_{DD_BASE}$(1) | Minimum power consumption in different sleep modes | — | 2 | — | µA | Idle mode, all peripherals disabled(2) |
| | | — | 700 | — | nA | Standby mode, all peripherals disabled(3) |
| | | — | 700 | — | nA | Power-Down mode, all peripherals disabled(3) |
| $I_{RST}$ | Reset power consumption | — | 170 | — | µA | $\overline{RESET}$ line pulled low |

**Notes:**
1. Single supply mode.
2. PMODE configured to AUTO in the Voltage Regulator Control (SLPCTRL.VREGCTRL) register and 32 kHz clock source selected.
3. PMODE configured to AUTO in the Voltage Regulator Control (SLPCTRL.VREGCTRL) register.

### 39.4.3 Peripherals Power Consumption

The table below can be used to calculate the additional current consumption for the different I/O peripherals in the various operating modes. Some peripherals will request the clock to be enabled when operating in STANDBY. Refer to the peripheral section for further information.

**Table 39-6. Peripherals Power Consumption[1]**

| Operating Conditions:<br>• $V_{DD} = V_{DDIO2} = 3V$<br>• $T_A = 25°C$<br>• OSCHF at 4 MHz used as clock source<br>• Device in Standby sleep mode | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| $I_{DD_WDT}$ | Watchdog Timer (WDT) | — | 350 | — | nA | 32 kHz Internal Oscillator |
| $I_{DD_MVIO}$ | Dual supply configuration, Multi-Voltage I/O (MVIO) | — | 500 | — | nA | |
| $I_{DD_VREF}$ | Voltage Reference (VREF) | — | — | — | µA | ADC0REF enabled, $V_{REF}$ = 2.048V |
| | | — | — | — | µA | ACREF enabled, $V_{REF}$ = 2.048V |
| | | — | — | — | µA | DACREF enabled, $V_{REF}$ = 2.048V |
| $I_{DD_BOD}$ | Brown-out Detector (BOD) | — | 20 | — | µA | Brown-out Detect (BOD) continuous, including bandgap |
| | | — | 1.3 | — | µA | BOD sampling @128 Hz, BODLEVEL3, including bandgap |
| | | — | 600 | — | nA | BOD sampling @32 Hz, BODLEVEL3, including bandgap |
| $I_{DD_TCA}$ | 16-bit Timer/Counter Type A (TCA) | — | 13 | — | µA | CLK_PER = HFOSC/4 = 1 MHz |
| $I_{DD_TCB}$ | 16-bit Timer/Counter Type B (TCB) | — | 10 | — | µA | |
| $I_{DD_TCD}$ | 12-bit Timer/Counter Type D (TCD) | — | — | — | µA | |
| $I_{DD_RTC}$ | Real-Time Counter (RTC) | — | 700 | — | nA | CLK_RTC = 32.768 kHz Internal Oscillator |
| $I_{DD_OSCHF}$ | Internal High-Frequency Oscillator (OSCHF) | — | 150 | — | µA | Internal Oscillator running at 4 MHz |
| $I_{DD_XOSCHF}$ | High Frequency Crystal Oscillator (XOSCHF) | — | 350 | — | µA | 20 MHz XTAL, $C_L$ = 15 pF |
| $I_{DD_OSC32K}$ | 32.768 kHz Internal Oscillator (OSC32K) | — | 350 | — | nA | |
| $I_{DD_XOSC32K}$ | 32.768 kHz Crystal Oscillator (XOSC32K) | — | 2.4 | — | µA | XOSC32K High Power, $C_L$ = 9 pF |
| | | — | 580 | — | nA | XOSC32K Low Power, $C_L$ = 9 pF |

.....continued

**Operating Conditions:**
- $V_{DD} = V_{DDIO2} = 3V$
- $T_A = 25°C$
- OSCHF at 4 MHz used as clock source
- Device in Standby sleep mode

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $I_{DD_ADC}$ | Analog-to-Digital Converter (ADC) | — | — | — | µA | ADC - Non-converting |
| | | — | 980 | — | µA | ADC @60 ksps(2) |
| | | — | 1.1 | — | mA | ADC @120 ksps(2) |
| $I_{DD_AC}$ | Analog Comparator (AC) | — | 70 | — | µA | Power Profile 0 |
| | | — | 12 | — | µA | Power Profile 1 |
| | | — | 6 | — | µA | Power Profile 2 |
| $I_{DD_OPAMP}$ | Analog Signal Conditioning (OPAMP), IRSEL = 0(3) | — | 1.2 | — | mA | One OPAMP in voltage follower mode, $V_{CM} = V_{DD}/2$ |
| | Analog Signal Conditioning (OPAMP), IRSEL = 1(3) | — | 880 | — | µA | One OPAMP in voltage follower mode, $V_{CM} = V_{DD}/2$ |
| $I_{DD_DAC}$ | Digital-to-Analog Converter (DAC) | — | — | — | µA | DAC + DACOUT, $DAC_{VREF} = V_{DD}/2$ |
| | | — | — | — | µA | DAC ($V_{DD}$), $DAC_{VREF} = V_{DD}/2$ |
| $I_{DD_USART}$ | Universal Synchronous and Asynchronous Receiver and Transmitter (USART) | — | 9 | — | µA | USART Enabled @9600 Baud |
| $I_{DD_SPI}$ | Serial Peripheral Interface (SPI) | — | — | — | µA | SPI Master @100 kHz |
| $I_{DD_TWI}$ | Two-Wire Interface (TWI) | — | 9 | — | µA | TWI Master @100 kHz |
| | | — | 7 | — | µA | TWI Slave @100 kHz |
| $I_{DD_NVM_ERASE}$ | Flash Programming Erase | — | — | — | mA | Flash Programming Erase |
| $I_{DD_NVM_WRITE}$ | Flash Programming Write | — | — | — | mA | Flash Programming Write |
| $I_{DD_ZCD}$ | Zero Cross Detector (ZCD) | — | 12 | — | µA | Excluding sink/source currents |

**Notes:**
1. Current consumption of the module only. To calculate the total internal power consumption of the microcontroller, add this value to the base power consumption given in the *Power Consumption* section in *Electrical Characteristics*.
2. Average power consumption with ADC active in Free-Running mode.
3. Without resistor ladder.

### 39.4.4 I/O Pin Characteristics

**Table 39-7. I/O Pin Characteristics(1)**

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| **Input Low Voltage** | | | | | | |

| .....continued | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| $V_{IL}$ | I/O PORT: | | | | | |
| | • With Schmitt Trigger buffer | — | — | $0.2 \times V_{DD}$ | V | |
| | • With $I^2C$ levels | — | — | $0.3 \times V_{DD}$ | V | |
| | • With SMBus 3.0 levels | — | — | 0.8 | V | |
| | $\overline{\text{RESET}}$ Pin | — | — | $0.2 \times V_{DD}$ | V | |
| | TTL level | — | — | 0.8 | V | |
| **Input High Voltage** | | | | | | |
| $V_{IH}$ | I/O PORT: | | | | | |
| | • With Schmitt Trigger buffer | $0.8 \times V_{DD}$ | — | — | V | |
| | • With $I^2C$ levels | $0.7 \times V_{DD}$ | — | — | V | |
| | • With SMBus 3.0 levels | 1.35 | — | — | V | |
| | $\overline{\text{RESET}}$ Pin | $0.8 \times V_{DD}$ | — | — | V | |
| | TTL level | 1.6 | — | — | V | |
| **Input Leakage Current**(2) | | | | | | |
| $I_{IL}$ | I/O PORTS(3) | — | <5 | — | nA | $GND \le V_{PIN} \le V_{DD}$, pin at high-impedance, 85°C |
| | | — | <5 | — | nA | $GND \le V_{PIN}$(5) $\le V_{DD}$, pin at high-impedance, 125°C |
| | $\overline{\text{RESET}}$ Pin(4) | — | ±50 | — | nA | $GND \le V_{PIN} \le V_{DD}$, pin at high-impedance, 85°C |
| **Pull-up Current** | | | | | | |
| $I_{PUR}$ | | — | 140 | — | µA | $V_{DD}$ = 3.0V, $V_{PIN}$ = GND |
| **Output Low Voltage** | | | | | | |
| $V_{OL}$ | Standard I/O Ports | — | — | 0.6 | V | $I_{OL}$ = 10 mA, $V_{DD}$ = 3.0V |
| **Output High Voltage** | | | | | | |
| $V_{OH}$ | Standard I/O Ports | $V_{DD}$-0.7 | — | — | V | $I_{OH}$ = 6 mA, $V_{DD}$ = 3.0V |
| **Pin Capacitance** | | | | | | |
| $C_{IO}$ | Op amp output | — | 9 | — | pF | |
| | $V_{REF}$pin | — | 7 | — | pF | |
| | XTAL pins | — | 4 | — | pF | |
| | Other pins | — | 4 | — | pF | |

**Notes:**

1. These figures are valid for all I/O ports regardless of if they are connected to the $V_{DD}$ or $V_{DDIO2}$ power domain.
2. The negative current is defined as current sourced by the pin.
3. The leakage current numbers for I/O PORTS are valid also when the pin is used as an input to an enabled analog peripheral.
4. The leakage current on the $\overline{\text{RESET}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.
5. A requirement for all I/O pins.

### 39.4.5 Memory Programming Specifications

**Table 39-8. Memory Programming Specifications[1]**

| Symbol | Description | Min. | Typ | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| **Data EEPROM Memory Specifications** | | | | | | |
| $E_D$ | Data EEPROM byte endurance | 100k | — | — | Erase/Write cycles | -40°C ≤ $T_A$ ≤ +85°C |
| $t_{D_RET}$ | Characteristic retention | — | 40 | — | Year | Provided no other specifications are violated |
| $N_{D_REF}$ | Total Erase/Write cycles before refresh | 1M | 4M | — | Erase/Write cycles | -40°C ≤ $T_A$ ≤ +85°C |
| $t_{D_CE}$ | Full EEPROM Erase | — | 10 | — | ms | |
| $V_{D_RW}$ | $V_{DD}$ for Read or Erase/Write operation | $V_{DDMIN}$ | — | $V_{DDMAX}$ | V | |
| $t_{D_BEW}$ | Byte Erase and Write cycle time | — | 11 | — | ms | |
| **Program Flash Memory Specifications** | | | | | | |
| $E_P$ | Flash memory cell endurance | 10k | — | — | Erase/Write cycles | -40°C ≤ $T_A$ ≤ +85°C |
| $t_{P_RET}$ | Characteristic retention | — | 40 | — | Year | Provided no other specifications are violated |
| $V_{P_RD}$ | $V_{DD}$ for Read operation | $V_{DDMIN}$ | — | $V_{DDMAX}$ | V | |
| $V_{P_REW}$ | $V_{DD}$ for Erase/Write operation | $V_{DD}$[2] | — | $V_{DDMAX}$ | V | |
| $t_{P_PE}$ | Page Erase | — | 10 | — | ms | |
| $t_{P_CE}$ | Chip Erase | — | — | — | ms | |
| $t_{P_WRD}$ | Byte/Word Write | — | 70 | — | µs | |

**Notes:**

1. These parameters are not tested but ensured by design.
2. During Chip Erase, the Brown-out Detector (BOD) configured with BODLEVEL0 is forced ON. If the supply voltage $V_{DD}$ is below $V_{BOD}$ for BODLEVEL0, the erase attempt will fail.

### 39.4.6 Thermal Characteristics

**Table 39-9. Thermal Characteristics**

| Symbol | Description | Typ. | Units | Conditions |
|---|---|---|---|---|
| $\theta_{JA}$ | Thermal Resistance Junction to Ambient (Thermal simulation, no airflow) | 60 | °C/W | 28-pin PDIP package (SP) |
| | | 74 | °C/W | 28-pin SOIC package (SO) |
| | | 67 | °C/W | 28-pin SSOP package (SS) |
| | | 36 | °C/W | 32-pin VQFN package (RXB) |
| | | 59 | °C/W | 32-pin TQFP package (PT) |
| | | 34 | °C/W | 48-pin VQFN package (6LX) |
| | | 56 | °C/W | 48-pin TQFP package (PT) |
| | | 30 | °C/W | 64-pin VQFN package (MR) |
| | | 39 | °C/W | 64-pin TQFP package (PT) |
| $T_{JMAX}$ | Maximum Junction Temperature | 145 | °C | |

**Notes:**

1. Power dissipation is calculated as follows:
   $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OI} \times I_{OL})$
2. Internal Power Dissipation is calculated as follows: $P_{INTERNAL} = I_{DD} \times V_{DD}$, where $I_{DD}$ is current to run the chip alone without driving any load on the output pins.
3. Derated Power is calculated as follows: $P_{DER} = PD_{MAX} (T_J-T_A)/\theta_{JA}$, where $T_A$ = Ambient Temperature, $T_J$ = Junction Temperature.

## 39.5 AC Characteristics

### 39.5.1 Clock Parameters

#### 39.5.1.1 Internal Oscillator Parameters

**Table 39-10. Internal Oscillators Characteristics(1)**

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $f_{OSCHF}$ | Precision calibrated OSCHF frequency | — | 1(2)<br>2(2)<br>3(2)<br>4<br>8(2)<br>12(2)<br>16(2)<br>20(2)<br>24(2) | — | MHz | |
| $\%_{CAL}$ | OSCHF tune step size | — | 0.4 | — | % | |

.....continued

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $t_{OSCHF_ST}$ | OSCHF wake-up from sleep start-up time | — | 24 | — | µs | Device in Idle or Standby mode<br>VREGCTRL.PMODE = FULL |
| | | — | 100 | — | µs | Device in Power-Down mode<br>VREGCTRL.PMODE = AUTO |
| $f_{OSC32K}$ | Internal OSC32K frequency | — | 32.768 | — | kHz | |
| $t_{OSC32K_ST}$ | OSC32K wake-up from sleep start-up time | — | 400 | — | µs | |

**Notes:**
1. To ensure these oscillator frequency tolerances, $V_{DD}$ and GND must be capacitively decoupled as close to the device as possible. 0.1 µF and 0.01 µF values in parallel are recommended.
2. These parameters are not calibrated.

### 39.5.1.2 32.768 kHz Crystal Oscillator (XOSC32K) Characteristics

**Table 39-11. 32.768 kHz Crystal Oscillator (XOSC32K) Characteristics(1)**

| Symbol | Description | Min. | Typ. | Max.(2) | Units | Conditions |
|---|---|---|---|---|---|---|
| $f_{XOSC32}$ | Frequency | — | 32.768 | — | kHz | |
| $C_L$ | Crystal load capacitance | — | — | 18 | pF | LPMODE = 0 |
| | | — | — | 8 | pF | LPMODE = 1 |
| ESR | Equivalent Series Resistance | — | — | 100 | kΩ | LPMODE = 0 |
| | | — | — | 50 | kΩ | LPMODE = 1 |
| $t_{XOSC32_ST}$ | XOSC32 start-up time | — | 300 | — | ms | $C_L$ = 9 pF, XOSC32 in High Power mode |

**Notes:**
1. See Pin capacitance in *I/O pin characteristics* for pin capacitance.
2. These values are based on simulation and not covered by production test limits.

### 39.5.1.3 High Frequency Crystal Oscillator (XOSCHF) Characteristics

**Table 39-12. High Frequency Crystal Oscillator (XOSCHF) Characteristics(1)**

| Symbol | Description | Min. | Typ. | Max. (2) | Units | Conditions |
|---|---|---|---|---|---|---|
| $f_{XOSCHF}$ | Frequency | 4 | — | 24 | MHz | |
| $C_L$ | Crystal load capacitance | — | 9 | — | pF | |

| .....continued | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** (2) | **Units** | **Conditions** |
| ESR | Equivalent Series Resistance | — | — | 200 | kΩ | 8 MHz (XOSCHFCTRLA.FRQRANGE = 0x0) |
| | | — | — | 60 | kΩ | 16 MHz (XOSCHFCTRLA.FRQRANGE = 0x1) |
| | | — | — | 40 | kΩ | 24 MHz (XOSCHFCTRLA.FRQRANGE = 0x2) |
| | | — | — | 40 | kΩ | 32 MHz (XOSCHFCTRLA.FRQRANGE = 0x3) |
| $t_{XOSCHF_ST}$ | XOSCHF start-up time | — | 700 | — | ns | 4 MHz crystal |

**Notes:**
1. See Pin capacitance in *I/O pin characteristics* for pin capacitance
2. These values are based on simulation and not covered by production test limits.

### 39.5.1.4 External Clock Characteristics

**Figure 39-2. External Clock Waveform**

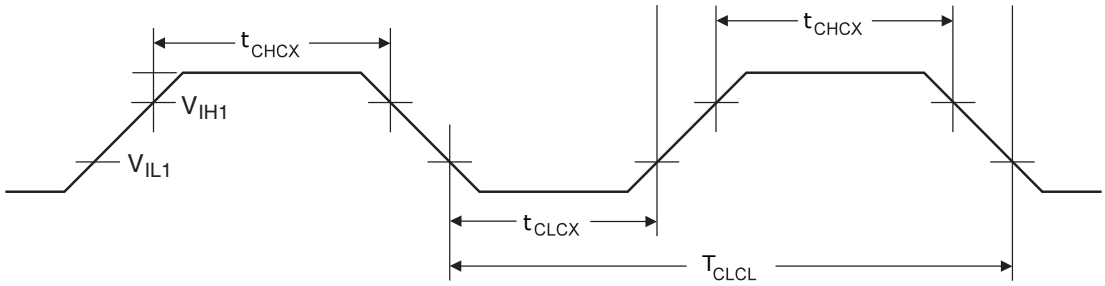



**Table 39-13. External Clock Characteristics**

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $f_{CLCL}$ | Clock frequency | — | — | 24 | MHz | |
| $T_{CLCL}$ | Clock period | 41.6 | — | — | ns | |
| $t_{CHCX}$ | High time | — | 40 | — | % | |
| $t_{CLCX}$ | Low time | — | 40 | — | % | |
| $\Delta T_{CLCL}$ | Change in period from cycle to cycle time | — | 20 | — | % | |

### 39.5.1.5 PLL Specifications

**Table 39-14. PLL Specifications**

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $f_{PLLIN}$ | PLL input frequency range | 16 | — | 24 | MHz | |
| $f_{PLLOUT}$ | PLL output frequency range | 32 | — | 48 | MHz | |

| .....continued | | | | | | |
|---|---|---|---|---|---|---|
| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
| $t_{PLLST}$ | PLL lock time | — | 10 | — | µs | |

### 39.5.2 Reset Controller Specifications

**Table 39-15. Reset, WDT, Oscillator Start-up Timer, Power-up Timer, Brown-out Detector Specifications**

| Sym. | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $t_{RST}$ | $\overline{\text{RESET}}$ pin pulse-width low to ensure reset(1) | 2.5 | — | — | µs | |
| $R_{RST_UP}$ | $\overline{\text{RESET}}$ pin pull-up resistor | — | 35 | — | kΩ | |
| $T_{WDT}$ | Watchdog Timer time-out period | — | 500 | — | ms | 1:512 Prescaler |
| $T_{SUT}$ | Power-up timer period | — | 64 | — | ms | SUT = `0x07` |
| $T_{OST}$ | Oscillator start-up timer period(2) | — | 1024 | — | cycles | |
| $V_{BOD}$ | Brown-out Detect voltage(3) | — | 1.90 | — | V | BODLEVEL0 |
| | | | 2.45 | | V | BODLEVEL1 |
| | | | 2.7 | | V | BODLEVEL2 |
| | | | 2.85 | | V | BODLEVEL3 |
| $V_{BOD_HYS}$ | Brown-out Detect hysteresis | — | 44 | — | mV | |
| $t_{BOD_ST}$ | Brown-out Detect start-up time | — | 1.9 | — | µs | |
| $t_{BOD_128HZ}$ | BOD response time Sampling Mode @128 Hz | — | 7.81 | — | ms | SAMPFREQ=`0` |
| $t_{BOD_32HZ}$ | BOD response time Sampling Mode @32 Hz | — | 31.25 | — | ms | SAMPFREQ=`1` |
| $t_{BOD_RST}$ | Brown-out reset response time | — | 3 | — | µs | |

**Notes:**

1. These values are based on simulation and not covered by production test limits.
2. By design, the Oscillator Start-up Timer ($T_{OST}$) counts the first 1024 cycles, independent of frequency.
3. To ensure these voltage tolerances, $V_{DD}$ and GND must be capacitively decoupled as close to the device as possible. 0.1 µF and 0.01 µF values in parallel are recommended.

**Table 39-16. Voltage Level Monitor Threshold Characteristics**

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $V_{DET}$ | Voltage detection threshold | — | 5 | — | % of BOD Threshold | VLMLVL = `0x01` |
| | | — | 15 | — | % of BOD Threshold | VLMLVL = `0x02` |
| | | — | 25 | — | % of BOD Threshold | VLMLVL = `0x03` |

### 39.5.3 Internal Voltage Reference ($V_{REF}$) Characteristics

**Table 39-17. Internal Voltage Reference ($V_{REF}$) Characteristics(1)**

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $V_{VREF_1V024}$(2) | Internal Voltage Reference 1.024V | — | ±4 | — | % | $V_{DD}$ ≥ 2.5V, -40°C to 85°C |
| $V_{VREF_2V048}$(2) | Internal Voltage Reference 2.048V | — | ±4 | — | % | $V_{DD}$ ≥ 2.5V, -40°C to 85°C |
| $V_{VREF_4V096}$(2) | Internal Voltage Reference 4.096V | — | ±4 | — | % | $V_{DD}$ ≥ 4.55V, -40°C to 85°C |

| .....continued | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| $V_{VREF_2V500}$[2] | Internal Voltage Reference 2.5V | — | ±4 | — | % | $V_{DD}$ ≥ 2.7V, -40°C to 85°C |
| $T_{VREF_ST}$ | VREF start-up time | — | 50 | — | µs | |

**Notes:**
1. These values are based on characterization and not covered by production test limits.
2. The symbol $V_{VREF_xVxxx}$ refers to the respective values of the REFSEL bit fields in the VREF.ADC0REF, VREF.DAC0REF and VREF.ACREF registers.

### 39.5.4 TCD

**Table 39-18. TCD Characteristics[1]**

| **Operating Conditions:**<br>• **CLK_TCD frequencies above maximum CLK_TCD_SYNC must be prescaled with the Synchronization Prescaler (SYNCPRES in TCDn.CTRLA), so the synchronizer clock meets these specifications** | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Unit** | **Condition** |
| $f_{CLK_TCD_SYNC}$ | CLK_TCD_SYNC maximum frequency | — | — | 48 | MHz | |

**Note:**
1. These parameters are for design guidance only and are not covered by production test limits.

### 39.5.5 USART

**Figure 39-3. USART in SPI Mode - Timing Requirements in Master Mode**

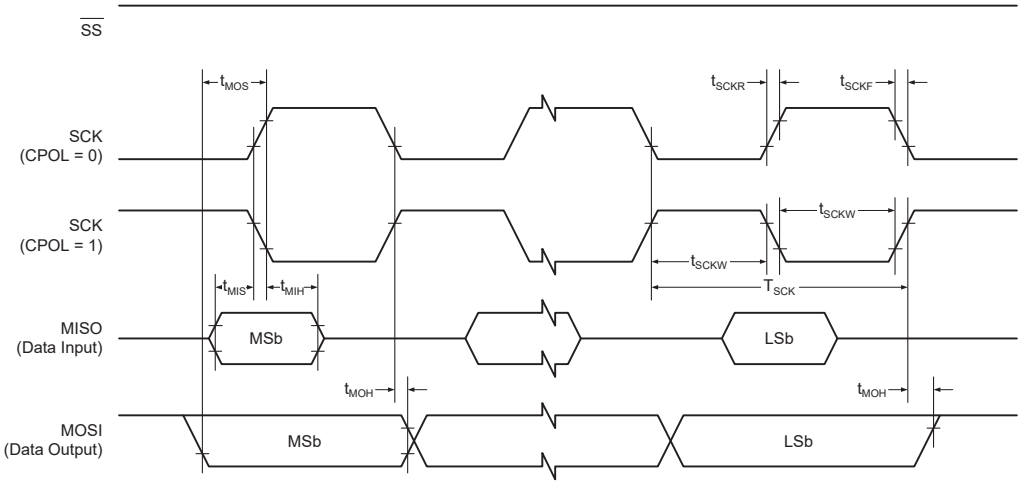



**Table 39-19. USART in SPI Master Mode - Timing Characteristics**

| Symbol | Description | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|
| $f_{SCK}$ | SCK clock frequency | — | — | 10 | MHz | |
| $T_{SCK}$ | SCK period | 100 | — | — | ns | |
| $t_{SCKW}$ | SCK high/low width | — | $0.5 \times T_{SCK}$ | — | ns | |
| $t_{SCKR}$ | SCK rise time | — | 2.7 | — | ns | |
| $t_{SCKF}$ | SCK fall time | — | 2.7 | — | ns | |

| .....continued | | | | | | |
|---|---|---|---|---|---|---|
| Symbol | Description | Min. | Typ. | Max. | Unit | Condition |
| $t_{MIS}$ | MISO setup to SCK | — | 10 | — | ns | |
| $t_{MIH}$ | MISO hold after SCK | — | 10 | — | ns | |
| $t_{MOS}$ | MOSI setup to SCK | — | $0.5 \times T_{SCK}$ | — | ns | |
| $t_{MOH}$ | MOSI hold after SCK | — | 1.0 | — | ns | |

### 39.5.6 SPI

**Figure 39-4. SPI - Timing Requirements in Master Mode**

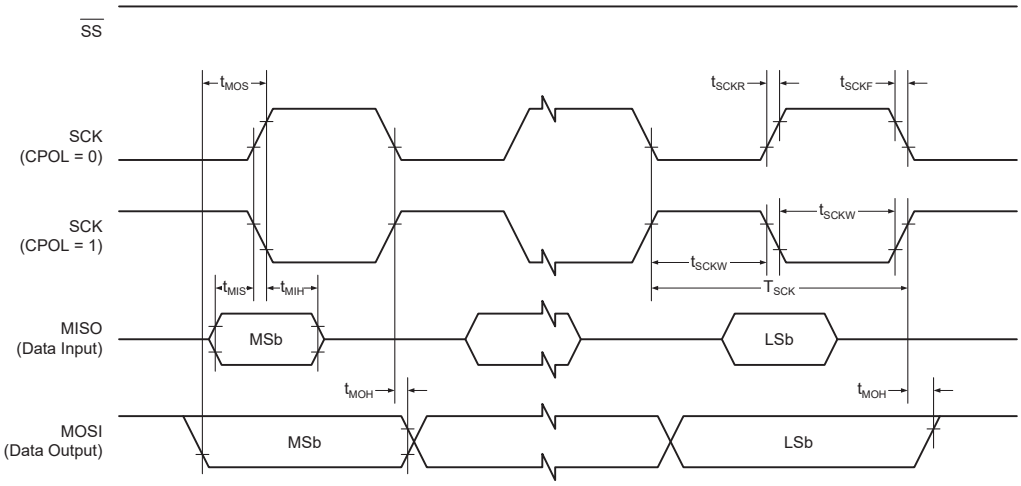



**Table 39-20. SPI - Timing Characteristics in Master Mode**

| Symbol | Description | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|
| $f_{SCK}$ | SCK clock frequency | — | — | 10 | MHz | |
| $T_{SCK}$ | SCK period | 100 | — | — | ns | |
| $t_{SCKW}$ | SCK high/low width | — | $0.5 \times T_{SCK}$ | — | ns | |
| $t_{SCKR}$ | SCK rise time | — | 2.7 | — | ns | |
| $t_{SCKF}$ | SCK fall time | — | 2.7 | — | ns | |
| $t_{MIS}$ | MISO setup to SCK | — | 10 | — | ns | |
| $t_{MIH}$ | MISO hold after SCK | — | 10 | — | ns | |
| $t_{MOS}$ | MOSI setup to SCK | — | $0.5 \times T_{SCK}$ | — | ns | |
| $t_{MOH}$ | MOSI hold after SCK | — | 1.0 | — | ns | |

**Figure 39-5. SPI - Timing Requirements in Slave Mode**

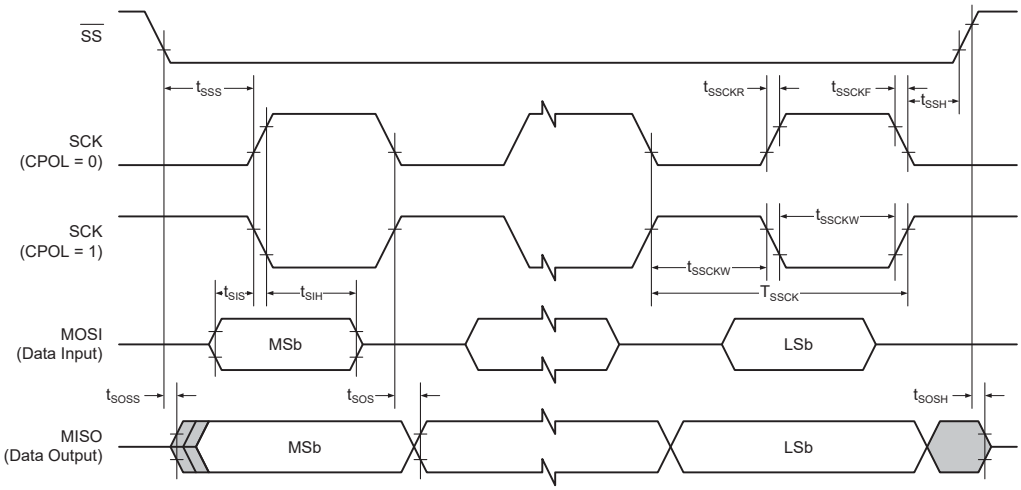



**Table 39-21. SPI - Timing Characteristics in Slave Mode**

| Symbol | Description | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|
| $f_{SSCK}$ | Slave SCK clock frequency | — | — | 5 | MHz | |
| $T_{SSCK}$ | Slave SCK period | $4 \times T_{CLK_PER}$ | — | — | ns | |
| $t_{SSCKW}$ | SCK high/low width | $2 \times T_{CLK_PER}$ | — | — | ns | |
| $t_{SSCKR}$ | SCK rise time | — | — | 1600 | ns | |
| $t_{SSCKF}$ | SCK fall time | — | — | 1600 | ns | |
| $t_{SIS}$ | MOSI setup to SCK | 3.0 | — | — | ns | |
| $t_{SIH}$ | MOSI hold after SCK | $T_{CLK_PER}$ | — | — | ns | |
| $t_{SSS}$ | SS setup to SCK | 21 | — | — | ns | |
| $t_{SSH}$ | SS hold after SCK | 20 | — | — | ns | |
| $t_{SOS}$ | MISO setup to SCK | — | 8.0 | — | ns | |
| $t_{SOH}$ | MISO hold after SCK | — | 13 | — | ns | |
| $t_{SOSS}$ | MISO setup after SS low | — | 11 | — | ns | |
| $t_{SOSH}$ | MISO hold after SS low | — | 8.0 | — | ns | |

## 39.5.7 TWI

**Figure 39-6. TWI - Timing Requirements**

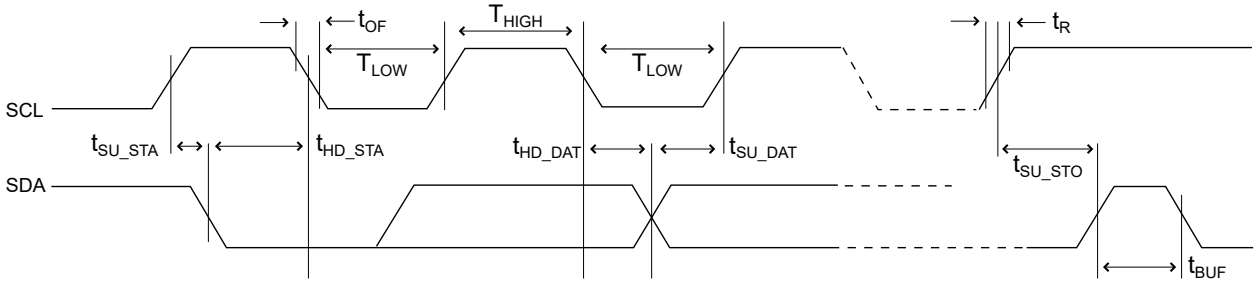


**Table 39-22. TWI - Timing Characteristics**

| Symbol | Description | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|
| $f_{SCL}$ | SCL clock frequency | 0 | — | 1000 | kHz | Max. frequency requires system clock at 10 MHz |
| $V_{IH}$ | Input high voltage | $0.7 \times V_{DD}$ | — | — | V | |
| $V_{IL}$ | Input low voltage | — | — | $0.3 \times V_{DD}$ | V | |
| $V_{HYS}$ | Hysteresis of Schmitt Trigger inputs | $0.1 \times V_{DD}$ | | $0.4 \times V_{DD}$ | V | |
| $V_{OL}$ | Output low voltage | — | — | $0.2xV_{DD}$ | V | $I_{load}$ = 20 mA, Fast mode+ |
| | | — | — | 0.4V | | $I_{load}$ = 3 mA, Normal mode, $V_{DD}$ > 2V |
| | | — | — | $0.2 \times V_{DD}$ | | $I_{load}$ = 3 mA, Normal mode, $V_{DD} \leq$ 2V |
| $I_{OL}$ | Low-level output current | 3 | — | — | mA | $f_{SCL} \leq$ 400 kHz, $V_{OL}$ = 0.4V |
| | | 20 | — | — | | $f_{SCL} \leq$ 1 MHz, $V_{OL}$ = 0.4V |
| $C_B$ | Capacitive load for each bus line | — | — | 400 | pF | $f_{SCL} \leq$ 100 kHz |
| | | — | — | 400 | | $f_{SCL} \leq$ 400 kHz |
| | | — | — | 550 | | $f_{SCL} \leq$1 MHz |
| $t_R$ | Rise time for both SDA and SCL | — | — | 1000 | ns | $f_{SCL} \leq$ 100 kHz |
| | | 20 | — | 300 | | $f_{SCL} \leq$ 400 kHz |
| | | — | — | 120 | | $f_{SCL} \leq$ 1 MHz |
| $t_{OF}$ | Output fall time from $V_{IHmin}$ to $V_{ILmax}$ | — | — | 250 | ns | $f_{SCL} \leq$ 100 kHz<br>10 pF < $C_B$ < 400 pF |
| | | $20 \times (V_{DD}/5.5V)$ | — | 250 | | $f_{SCL} \leq$ 400 kHz<br>10 pF < $C_B$ < 400 pF |
| | | $20 \times (V_{DD}/5.5V)$ | — | 120 | | $f_{SCL} \leq$ 1 MHz<br>10 pF < $C_B$ < 400 pF |
| $t_{SP}$ | Spikes suppressed by the input filter | 0 | — | 50 | ns | |
| $I_L$ | Input current for each I/O pin | — | — | 1 | µA | $0.1 \times V_{DD} < V_I < 0.9 \times V_{DD}$ |
| $C_I$ | Capacitance for each I/O pin | — | — | 10 | pF | |
| $R_P$ | Value of pull-up resistor | $(V_{DD}-V_{OL}(max)) / I_{OL}$ | — | 1000 ns/ $(0.8473 \times C_B)$ | Ω | $f_{SCL} \leq$ 100 kHz |
| | | — | — | 300 ns/ $(0.8473 \times C_B)$ | | $f_{SCL} \leq$ 400 kHz |
| | | — | — | 120 ns/ $(0.8473 \times C_B)$ | | $f_{SCL} \leq$ 1 MHz |

.....continued

| Symbol | Description | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|
| $t_{HD_STA}$ | Hold time (repeated) Start condition | 4.0 | — | — | µs | $f_{SCL}$ ≤ 100 kHz |
| | | 0.6 | — | — | | $f_{SCL}$ ≤ 400 kHz |
| | | 0.26 | — | — | | $f_{SCL}$ ≤ 1 MHz |
| $T_{LOW}$ | Low period of SCL Clock | 4.7 | — | — | µs | $f_{SCL}$ ≤ 100 kHz |
| | | 1.3 | — | — | | $f_{SCL}$ ≤ 400 kHz |
| | | 0.5 | — | — | | $f_{SCL}$ ≤ 1 MHz |
| $T_{HIGH}$ | High period of SCL Clock | 4.0 | — | — | µs | $f_{SCL}$ ≤ 100 kHz |
| | | 0.6 | — | — | | $f_{SCL}$ ≤ 400 kHz |
| | | 0.26 | — | — | | $f_{SCL}$ ≤ 1 MHz |
| $t_{SU_STA}$ | Setup time for a repeated Start condition | 4.7 | — | — | µs | $f_{SCL}$ ≤ 100 kHz |
| | | 0.6 | — | — | | $f_{SCL}$ ≤ 400 kHz |
| | | 0.26 | — | — | | $f_{SCL}$ ≤ 1 MHz |
| $t_{HD_DAT}$ | Data hold time | — | 0 | — | ns | SDAHOLD[1:0] = `0x0` |
| | | 30 | — | 300 | | SDAHOLD[1:0] = `0x1` |
| | | 120 | — | 420 | | SDAHOLD[1:0] = `0x2` |
| | | 300 | — | 900 | | SDAHOLD[1:0] = `0x3` |
| $t_{SU_DAT}$ | Data setup time | 250 | — | — | ns | $f_{SCL}$ ≤ 100 kHz |
| | | 100 | — | — | | $f_{SCL}$ ≤ 400 kHz |
| | | 50 | — | — | | $f_{SCL}$ ≤ 1 MHz |
| $t_{SU_STO}$ | Setup time for Stop condition | 4 | — | — | µs | $f_{SCL}$ ≤ 100 kHz |
| | | 0.6 | — | — | | $f_{SCL}$ ≤ 400 kHz |
| | | 0.26 | — | — | | $f_{SCL}$ ≤ 1 MHz |
| $t_{BUF}$ | Bus free time between a Stop and Start condition | 4.7 | — | — | µs | $f_{SCL}$ ≤ 100 kHz |
| | | 1.3 | — | — | | $f_{SCL}$ ≤ 400 kHz |
| | | 0.5 | — | — | | $f_{SCL}$ ≤ 1 MHz |

**Note:** $I^2C$ Fm+ is only supported for above 2.7V.

### 39.5.8 UPDI Timing

**Figure 39-7. UPDI Enable Sequence with Dedicated UPDI Pin**

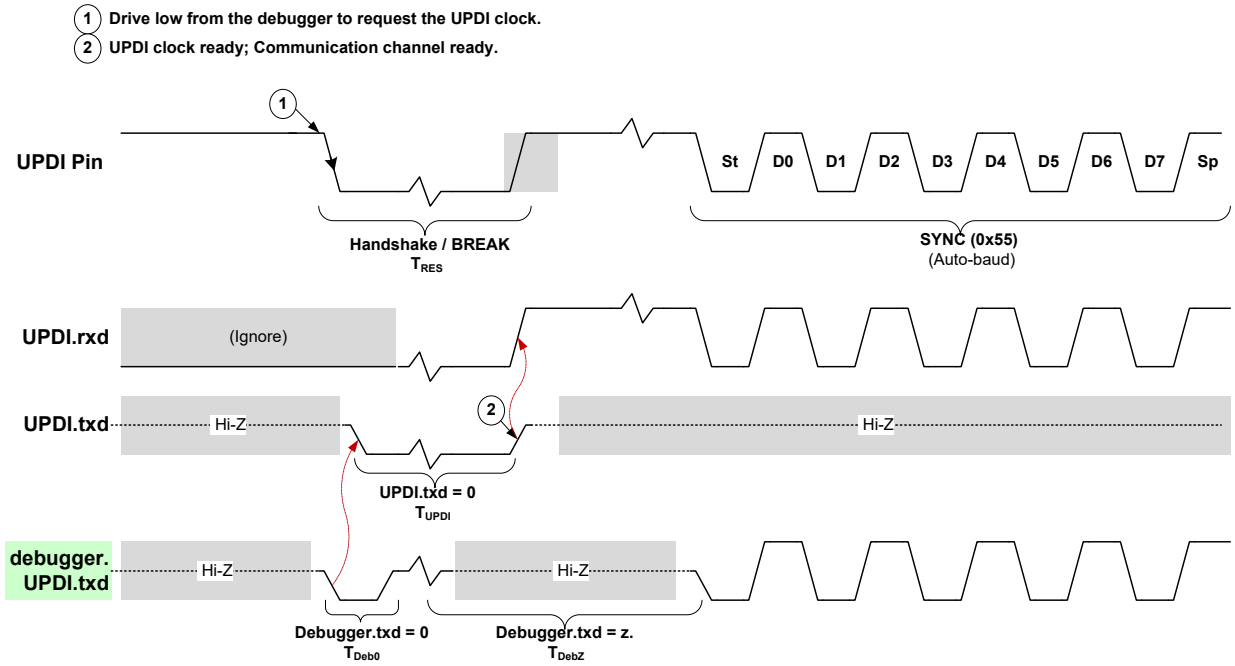



**Table 39-23. UPDI Timing Characteristics**

| Symbol | Description | Min. | Max. | Unit |
|---|---|---|---|---|
| $t_{RES}$ | Duration of Handshake/Break on $\overline{RESET}$ | 10 | 200 | µs |
| $t_{UPDI}$ | Duration of UPDI.txd = 0 | 10 | 200 | µs |
| $t_{Deb0}$ | Duration of Debugger.txd = 0 | 0.2 | 1 | µs |
| $t_{DebZ}$ | Duration of Debugger.txd = z | 200 | 14000 | µs |

**Table 39-24. UPDI Max. Bit Rates vs. Temperature**

| Symbol | Description | Min. | Max. | Unit | Condition |
|---|---|---|---|---|---|
| $f_{UPDI}$ | UPDI baud rate | — | 1.8 | Mbps | Temp. 0ºC to 50ºC |
| | | — | 0.9 | Mbps | Temp. <0ºC or >50ºC |

### 39.5.9 DAC Specifications

**Table 39-25. DAC Electrical Specifications**

**Operating Conditions:**
- $V_{DD} = V_{DDIO2} = 3.0V$
- $T_A = 25ºC$

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $V_{OUT}$ | Output voltage range | GND + 100 mV | — | $V_{DD}$ - 100 mV | V | |

.....continued

| Operating Conditions:<br>• $V_{DD} = V_{DDIO2} = 3.0V$<br>• $T_A = 25ºC$ | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| $I_{OUT}$ | Output sink | — | — | — | A | |
| | Output source | — | — | — | A | |
| $V_{LSB}$ | Resolution | — | 10 | — | Bit | |
| $V_{ACC}$ | Absolute accuracy | — | 1 | — | LSb | |
| $t_{ST}$ | Settling time(1) | — | 7 | — | µs | $V_{DD}$ = 3.0V<br>$V_{DACREF} = V_{DD}$ |
| | | — | 10 | — | µs | $V_{DD}$ = 5.5V<br>$V_{DACREF} = V_{DD}$ |
| INL | Integral Non-Linearity | — | 1 | — | LSb | |
| DNL | Differential Non-Linearity | — | 0.2 | — | LSb | |
| $E_{OFF}$ | Offset error | — | 0.55 | — | LSb | |
| $E_{GAIN}$ | Gain error | — | 2.8 | — | LSb | |

**Note:**
1. Settling time measured while DACR[9:0] transitions from '`0x000`' to '`0x3FF`'.

### 39.5.10 ADC Specifications

**Table 39-26. ADC Accuracy Specifications**

| Operating Conditions:<br>• $V_{DD} = V_{DDIO2} = 3.0V$<br>• $T_A = 25ºC$ | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| $N_R$ | Resolution | — | — | 12 | bit | |
| $E_{INL}$ | Integral Non-Linearity error | — | 0.1 | — | LSb | |
| $E_{DNL}$ | Differential Non-Linearity error | — | 0.1 | — | LSb | |
| $E_{OFF}$ | Offset error | — | 1.25 | — | LSb | |
| $E_{GAIN}$ | Gain error | — | 2.5 | — | LSb | |
| $E_{ABS}$ | Absolute error | — | 4 | — | LSb | |
| $V_{ADCREF}$ | ADC reference voltage | 1.8 | — | $V_{DD}$ | V | |
| $V_{AIN}$ | ADC input pin voltage | GND | — | $V_{ADCREF}$ | V | |
| $Z_{AIN}$ | Recommended impedance of analog voltage source | — | — | 10 | kΩ | |
| $R_{VREFA}$ | ADC voltage reference ladder impedance(2) | — | 50 | — | kΩ | |

**Notes:**
1. The ADC conversion result never decreases with an increase in the input and has no missing codes.
2. This is the impedance seen by the VREFA pin when the external reference is selected.

**Table 39-27. ADC Conversion Timing Specifications**

| Symbol | Description | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $T_{CLK_ADC}$ | ADC clock period | 0.5 | — | 8 | µs | |
| $t_{CNV}$ | Conversion time | — | $13.5 \times T_{CLK_ADC} + 2 \times T_{CLK_PER}$ | — | µs | |
| $t_{ACQ}$ | Acquisition time | — | $2 \times T_{CLK_ADC}$ | — | µs | |
| $f_{ADC}$ | Sample rate | 8 | — | 130 | ksps | |
| $t_S$ | Sampling time | — | $2 \times T_{CLK_ADC}$ | — | µs | |

### 39.5.11 Analog Comparator Specifications

**Table 39-28. Analog Comparator Specifications**

| Operating Conditions:<br>• $V_{DD} = V_{DDIO2} = 3.0V$<br>• $T_A = 25ºC$ | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Unit** | **Condition** |
| $V_{IN}$ | Input voltage range | -0.2 | — | $V_{DD}$ | V | |
| $V_{OFF}$ | Input offset voltage | — | ±5 | — | mV | $0.7V < V_{IN} < (V_{DD}-0.7V)$ |
| | | — | ±20 | — | | $-0.2V < V_{IN} < V_{DD}$ |
| CMRR | Common Mode input Rejection Ratio | — | — | — | dB | |
| $V_{HYST}$ | Hysteresis | — | 10 | — | mV | HYSMODE = `0x1` |
| | | — | 25 | — | | HYSMODE = `0x2` |
| | | — | 50 | — | | HYSMODE = `0x3` |
| $t_{RESP}$ | Response time, rising edge | — | 50 | — | ns | Power Profile 0 |
| | Response time, falling edge | — | 50 | — | ns | |
| | Response time, rising edge | — | 200 | — | ns | Power Profile 1 |
| | Response time, falling edge | — | 250 | — | ns | |
| | Response time, rising edge | — | 400 | — | ns | Power Profile 2 |
| | Response time, falling edge | — | 480 | — | ns | |

### 39.5.12 OPAMP Specifications

**Table 39-29. OPAMP Electrical Specifications**

| Operating Conditions:<br>• $V_{DD} = AVDD = 3.0V$<br>• $V_{CM} = V_{DD}/2$<br>• $T_A = 25ºC$ | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| GBWP | Gain Bandwidth Product | — | 5.0 | — | MHz | |
| $T_{ON}$ | Turn-on time | — | — | — | µs | |
| $P_M$ | Phase margin | — | 80 | — | Degrees | |

.....continued

| Operating Conditions:<br>• $V_{DD}$ = AVDD = 3.0V<br>• $V_{CM}$ = $V_{DD}$/2<br>• $T_A$ = 25ºC | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| $S_R$ | Slew rate rise | — | 9.5 | — | V/µs | |
| | Slew rate fall | — | 13 | | V/µs | |
| $V_{OS}$ | Offset | — | < ±8 | — | mV | $V_{DD}$ = 1.8 - 5.5V, $V_{CM}$ = -0.1 - ($V_{DD}$ + 0.1) V, Temp. -40ºC to 85ºC |
| | Offset | — | ±0.5 | — | mV | $V_{DD}$ = 3V, $V_{CM}$= $V_{DD}$/2 , Temp. 25ºC |
| | Input offset drift with temperature | — | ±30 | — | µV/ºC | |
| | Offset calibration step | — | 0.5 | — | mV | |
| CMRR | Common-Mode Rejection Ratio | — | 70 | — | dB | |
| PSRR | Power Supply Rejection Ratio | — | 70 | — | dB | |
| $A_{OL}$ | Open-loop gain | — | 90 | — | dB | |
| $V_{ICM}$ | Input common-mode voltage | -0.3 | — | $V_{DD}$ + 0.3 | V | IRSEL = 0 |
| | Input common-mode voltage | -0.3 | — | $V_{DD}$ - 0.7 | V | IRSEL = 1 |
| $I_{SC}$ | Short circuit sourcing current | — | 33 | — | mA | |
| | Short circuit sinking current | — | 27 | — | mA | |
| $E_{NI}$ | Input noise voltage | — | 180 | — | µVpp | f = 0.1 - 10 Hz |
| $V_{OL}$ | Maximum output voltage swing | — | < 0.1 | — | V | $I_{load}$= 1.5 mA |
| $V_{OH}$ | Maximum output voltage swing | — | > $V_{DD}$ - 0.1 | — | V | $I_{load}$= 1.5 mA |
| | System gain accuracy with internal resistor ladder | — | < 5 | — | % | |

### 39.5.13 Zero-Cross Detector Specifications

**Table 39-30. Zero-Cross Detector Specifications**

| Operating Conditions:<br>• $V_{DD}$ = $V_{DDIO2}$ = 3.0V<br>• $T_A$ = 25ºC | | | | | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| $V_{PINZC}$ | Voltage on ZCD pin | — | 0.9 | — | V | |
| $I_{ZCD_MAX}$ | Maximum source or sink current | — | — | 600 | µA | |
| $t_{RESPH}$ | Response time, rising edge | — | 1 | — | µs | |
| $t_{RESPL}$ | Response time, falling edge | — | 1 | — | µs | |

# 40. Typical Characteristics

## 40.1 OPAMP

Unless otherwise noted, the typical graphs are valid for the following conditions:

- Output load: 50 pF||3 kΩ
- Input common-mode voltage at $V_{DD}/2$
- Internal voltage follower mode
- IRSEL = 0

**Figure 40-1. Offset at 25°C over $V_{CM}$ and $V_{DD}$**

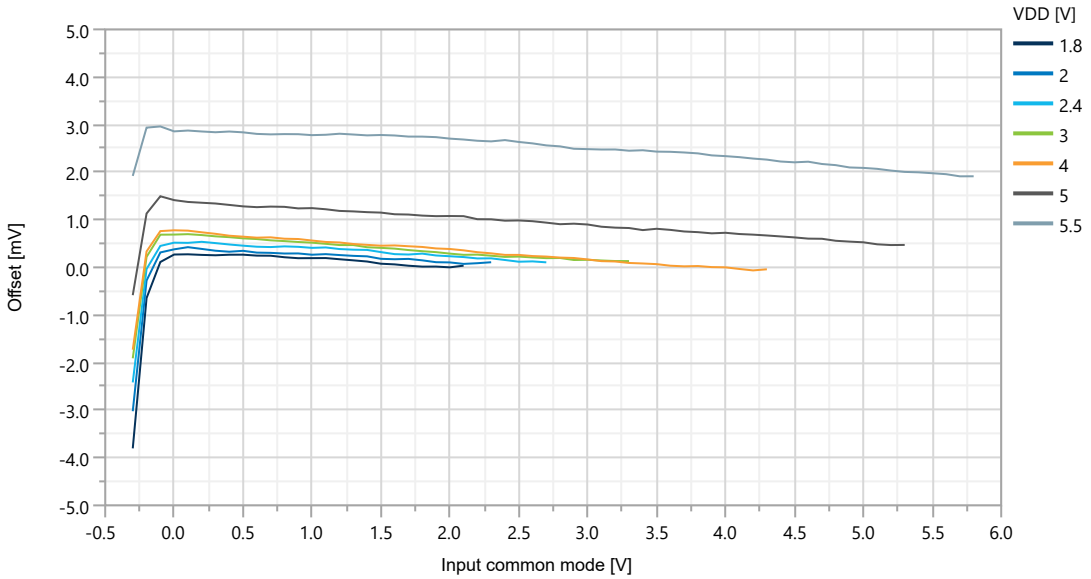



**Figure 40-2. Offset at 25°C over $V_{CM}$ and $V_{DD}$ with IRSEL = 1**

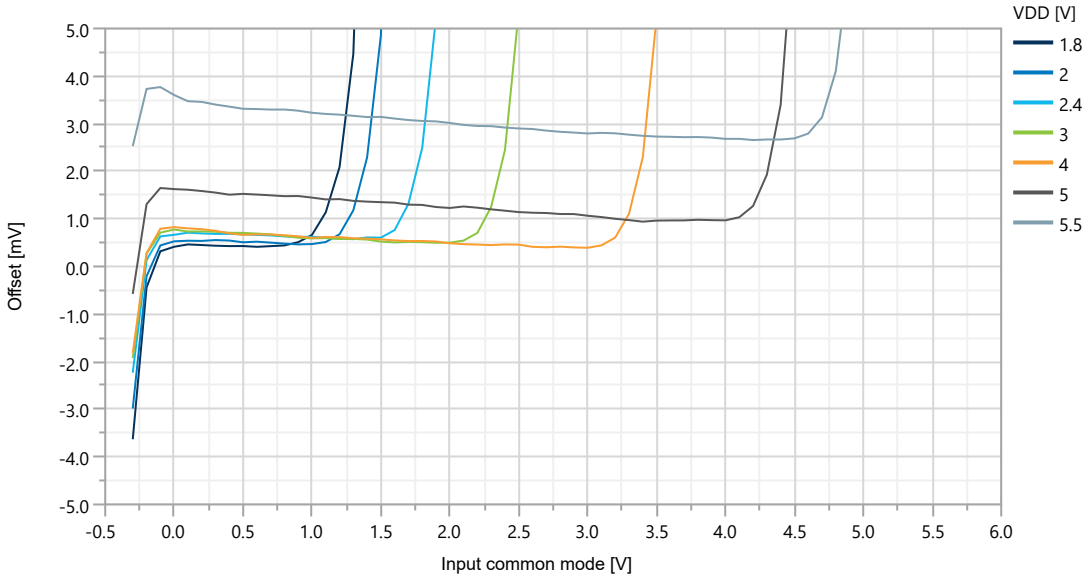


**Figure 40-3. Offset at 3V over $V_{CM}$ and Temperature**

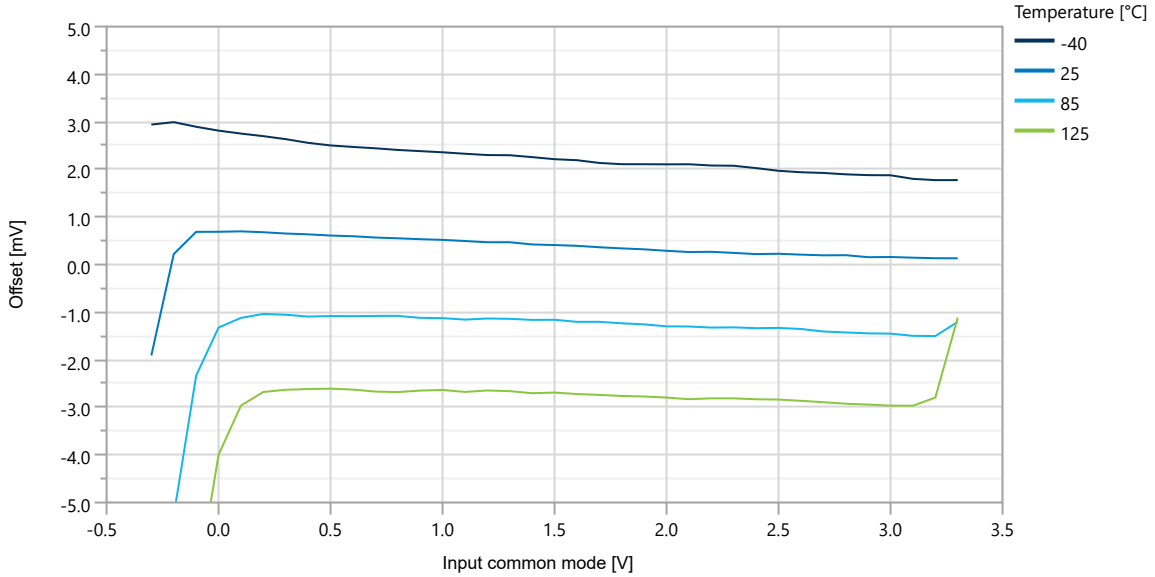



**Figure 40-4. Offset at 5V over $V_{CM}$ and Temperature**

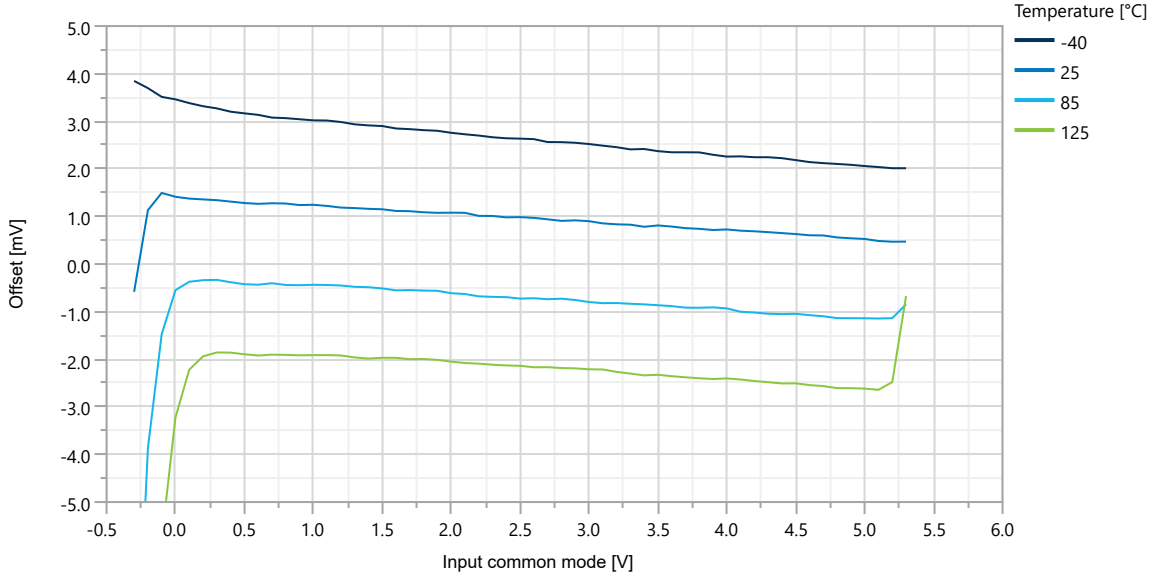


**Figure 40-5. PSRR at 3V $V_{DD}$ and $V_{CM}$ = $V_{DD}$/2 over Frequency and Temperature**

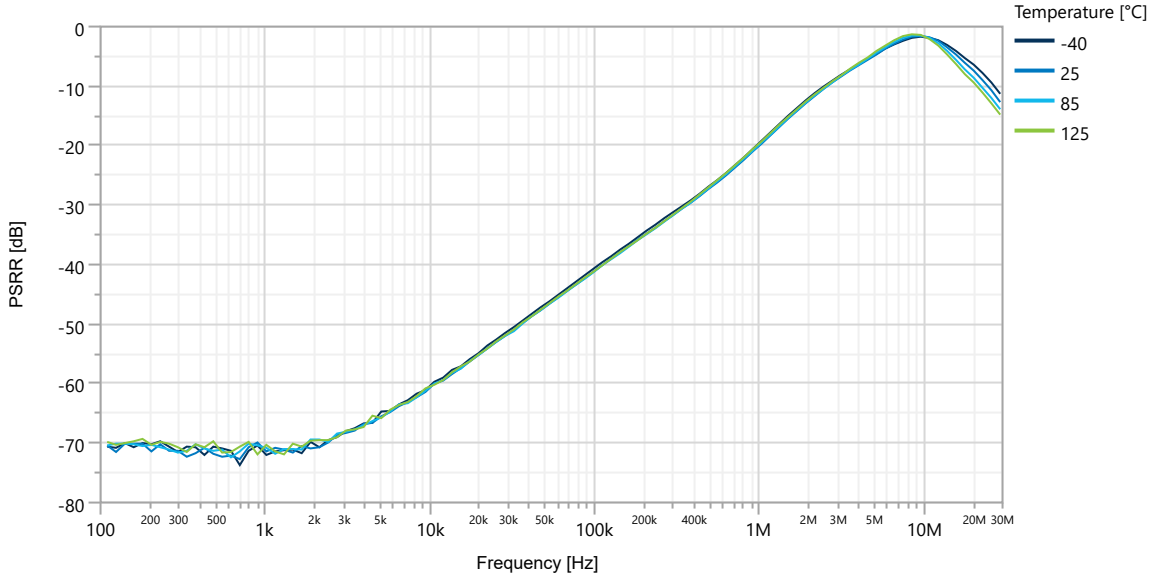



**Figure 40-6. PSRR at DC and $V_{CM}$ =$V_{DD}$/2 over $V_{DD}$ and Temperature**

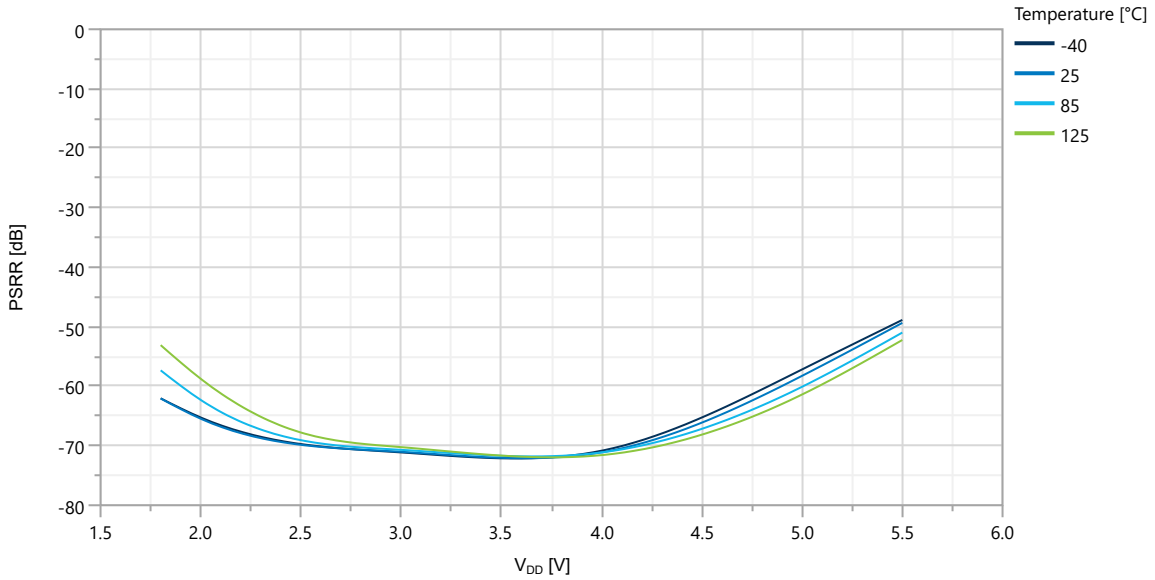


**Figure 40-7. Open Loop Gain and Phase bodeplot at 3V and 25°C**

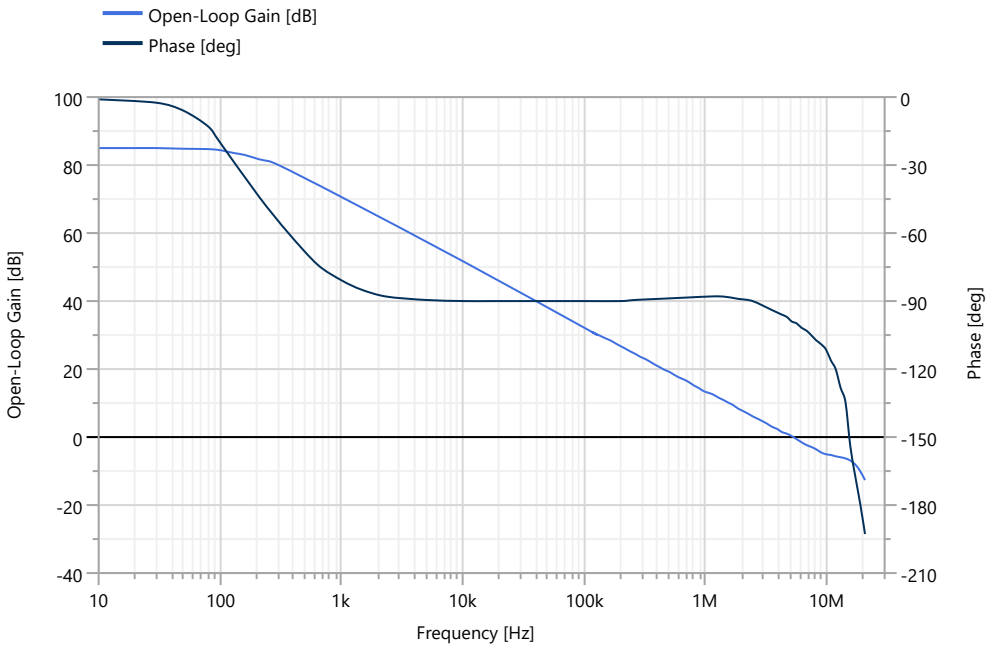



**Figure 40-8. Open Loop Gain at 25°C over $V_{DD}$ and $V_{CM}$**

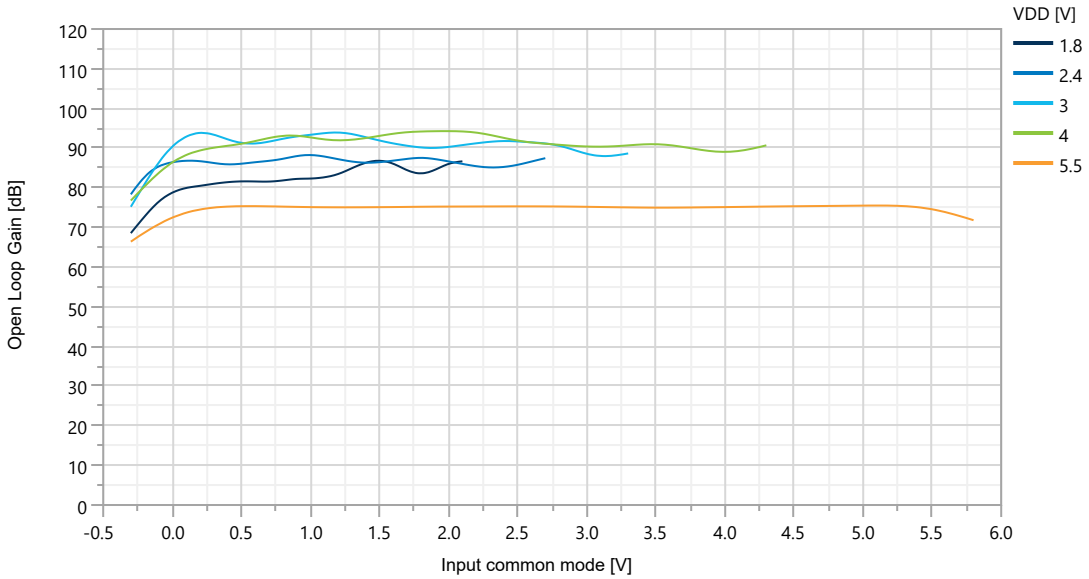


**Figure 40-9. Open Loop Gain at $V_{CM} = V_{DD}/2$ over $V_{DD}$ and Temperature**

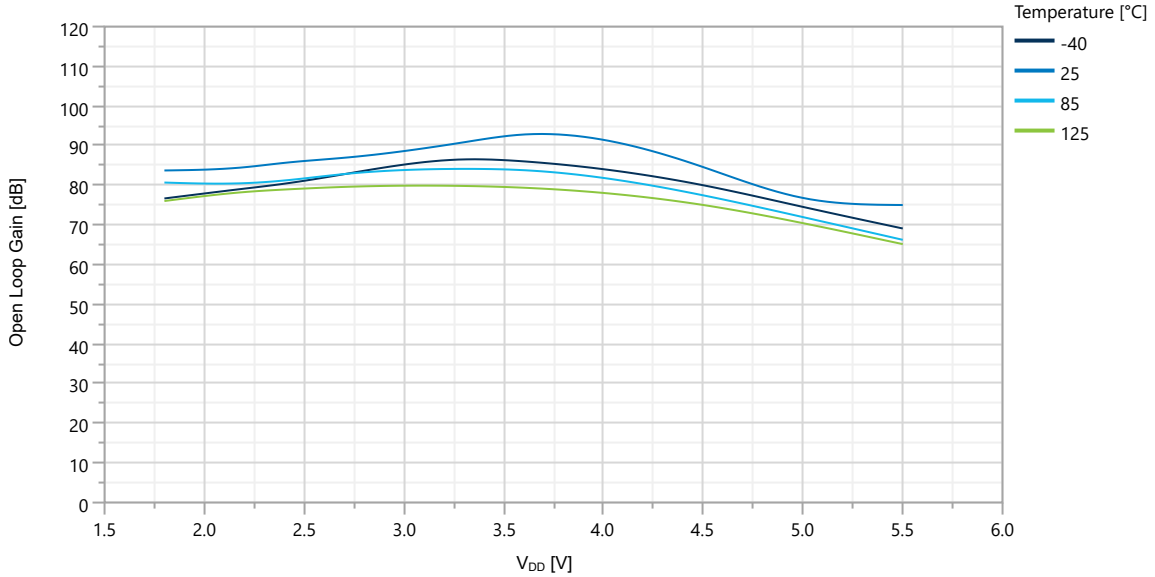



**Figure 40-10. Slew Rate Falling over $V_{DD}$ and Temperature**

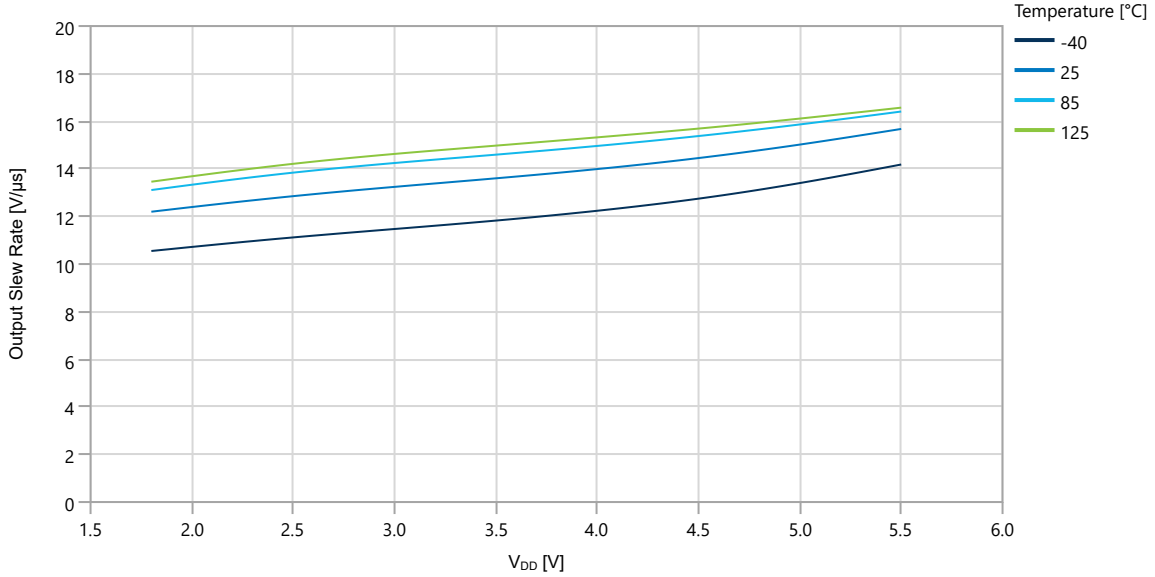


**Figure 40-11. Slew Rate Rising over $V_{DD}$ and Temperature**

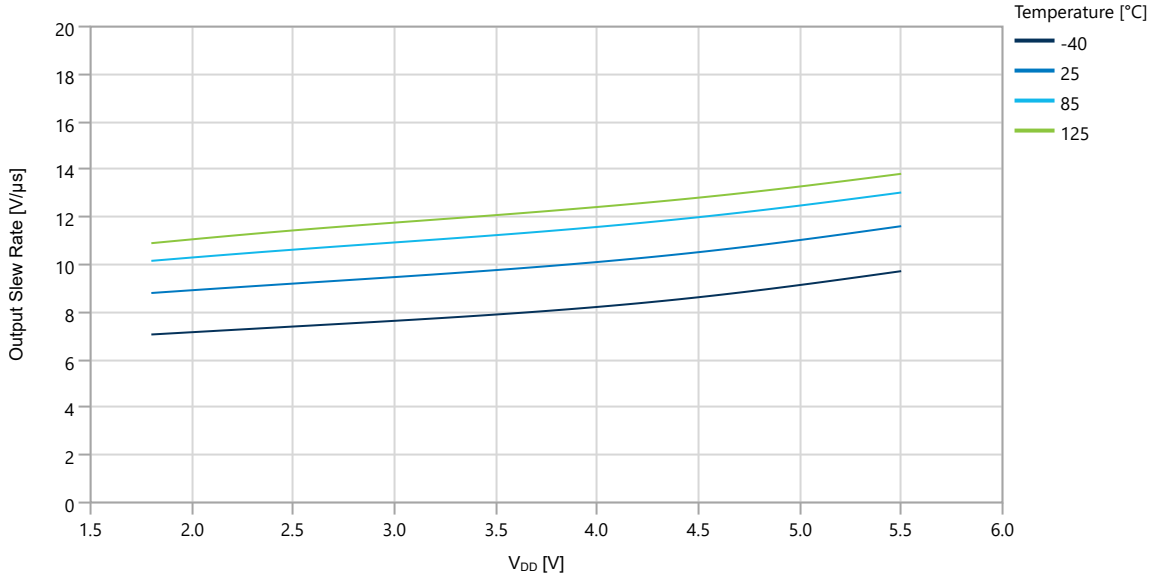



**Figure 40-12. $V_{OH}$ at 3V over Temperature and Load Current**

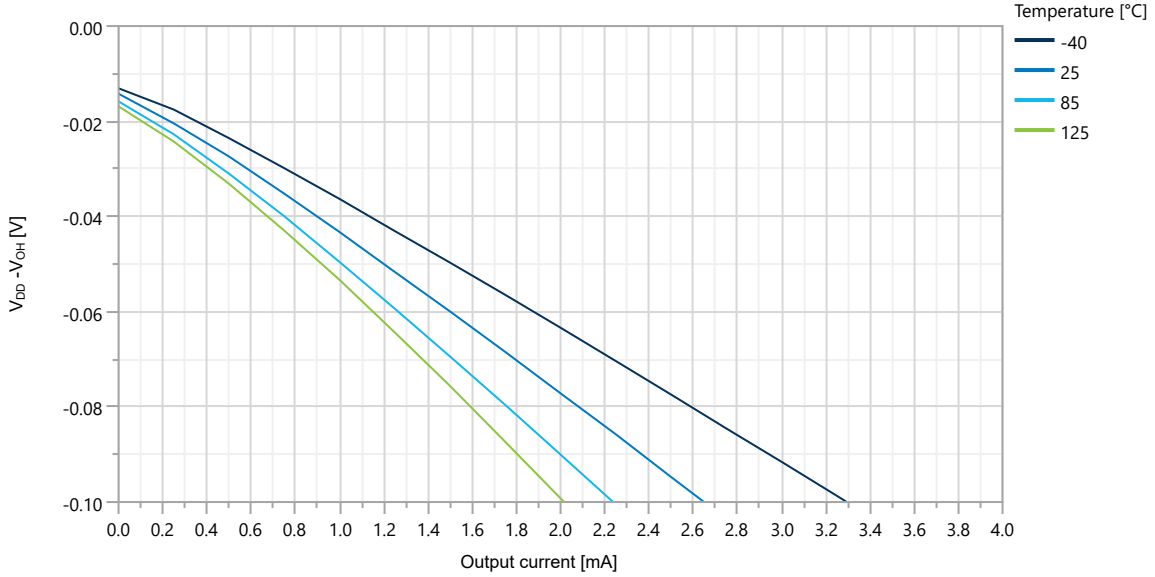


**Figure 40-13. $V_{OH}$ at 5V over Temperature and Load Current**

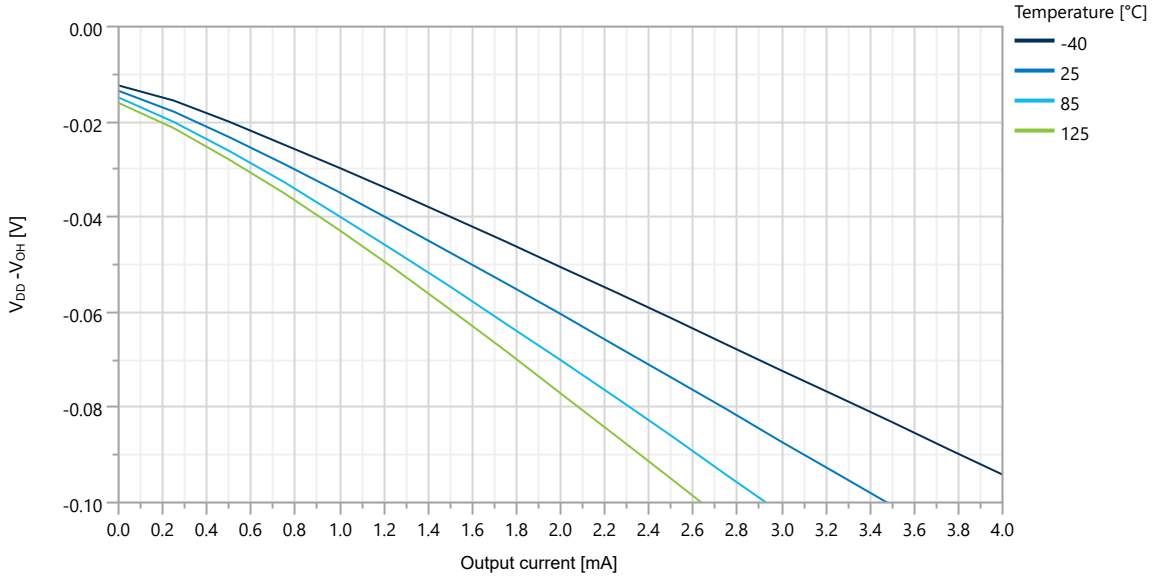



**Figure 40-14. $V_{OL}$ at 3V over Temperature and Load Current**

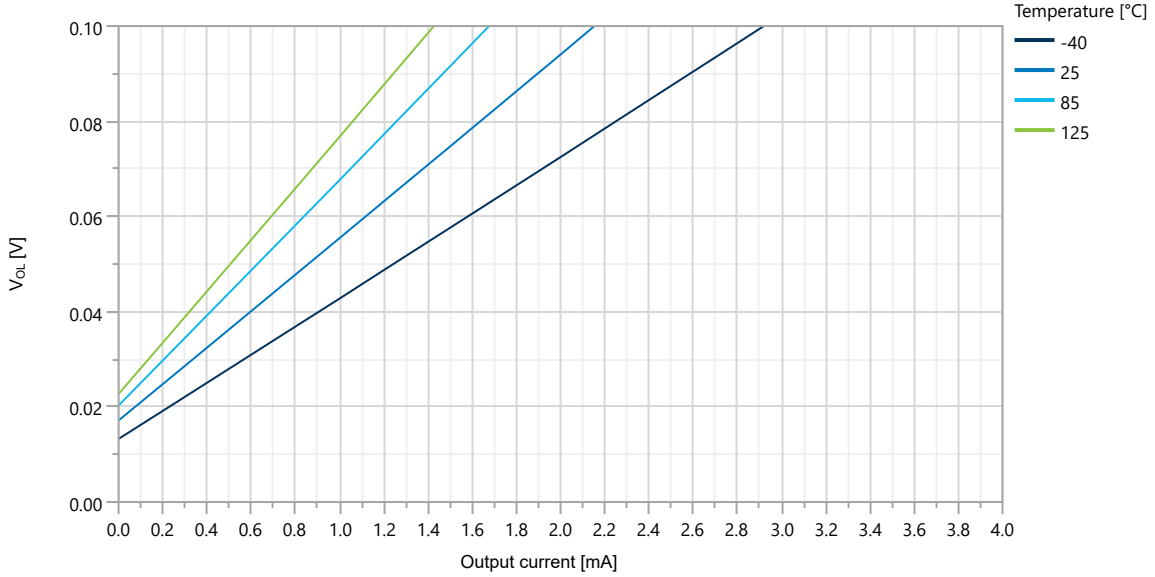


**Figure 40-15. $V_{OL}$ at 5V over Temperature and Load Current**

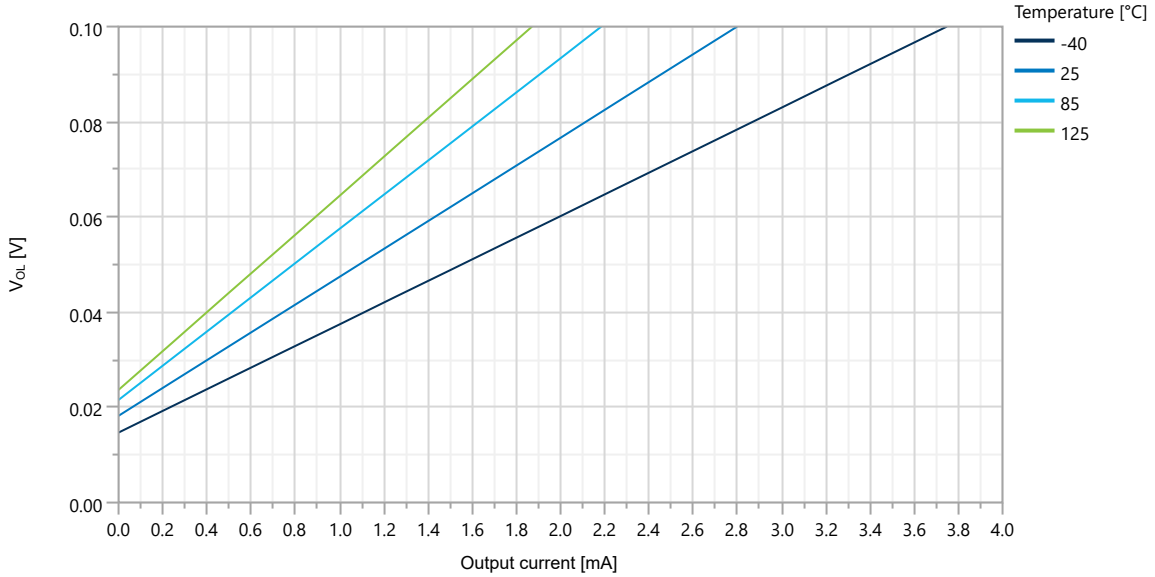



**Figure 40-16. Output Short Circuit Current over $V_{DD}$ and Temperature**

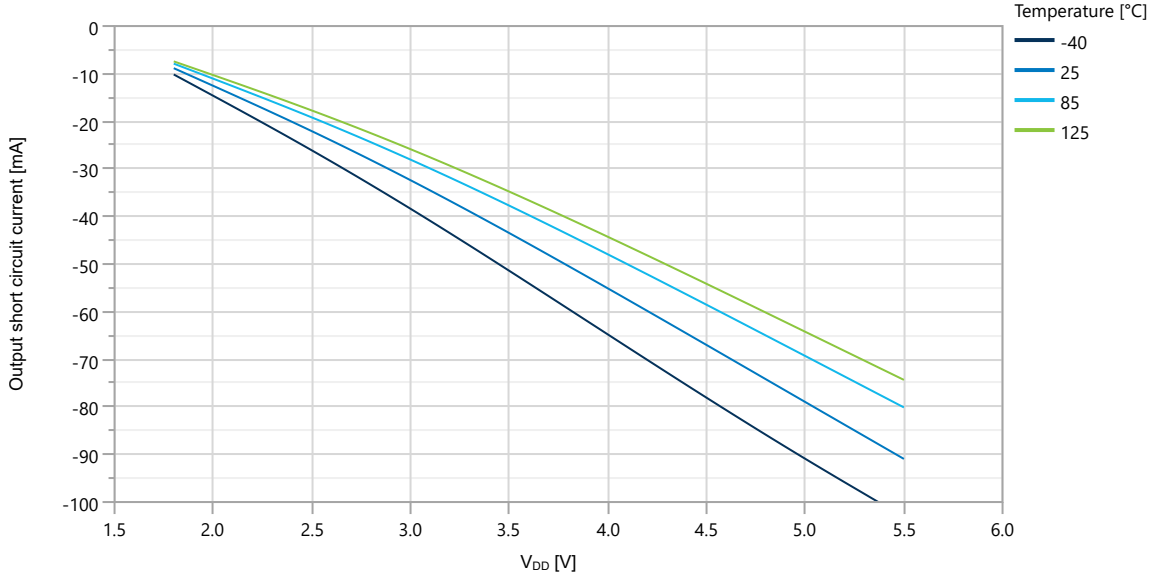


**Figure 40-17. $I_{DD}$ over $V_{DD}$ and Temperature with IRSEL = 0**

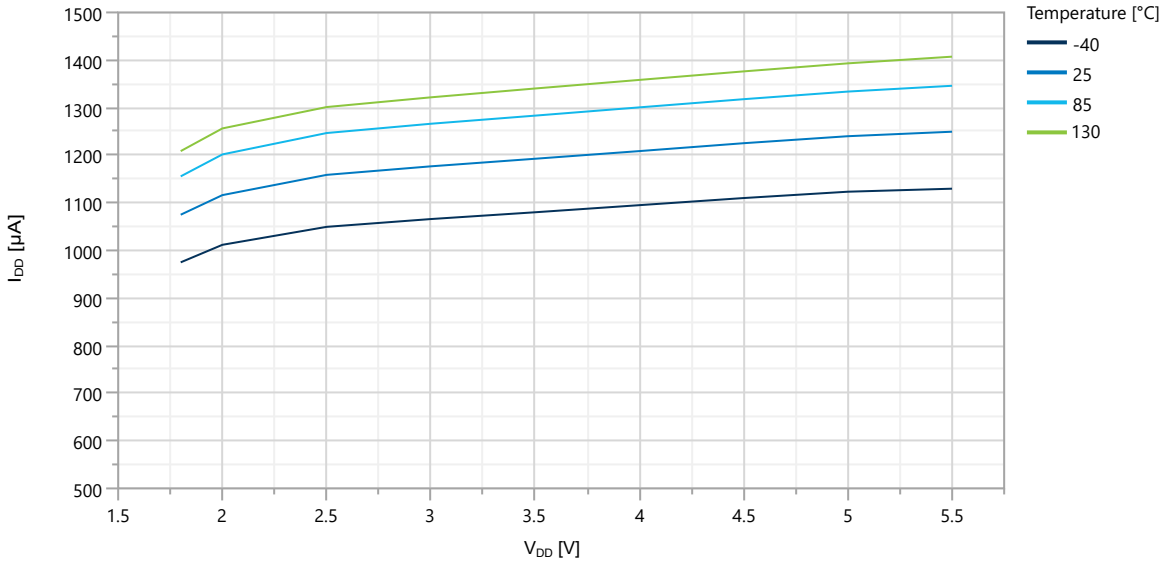
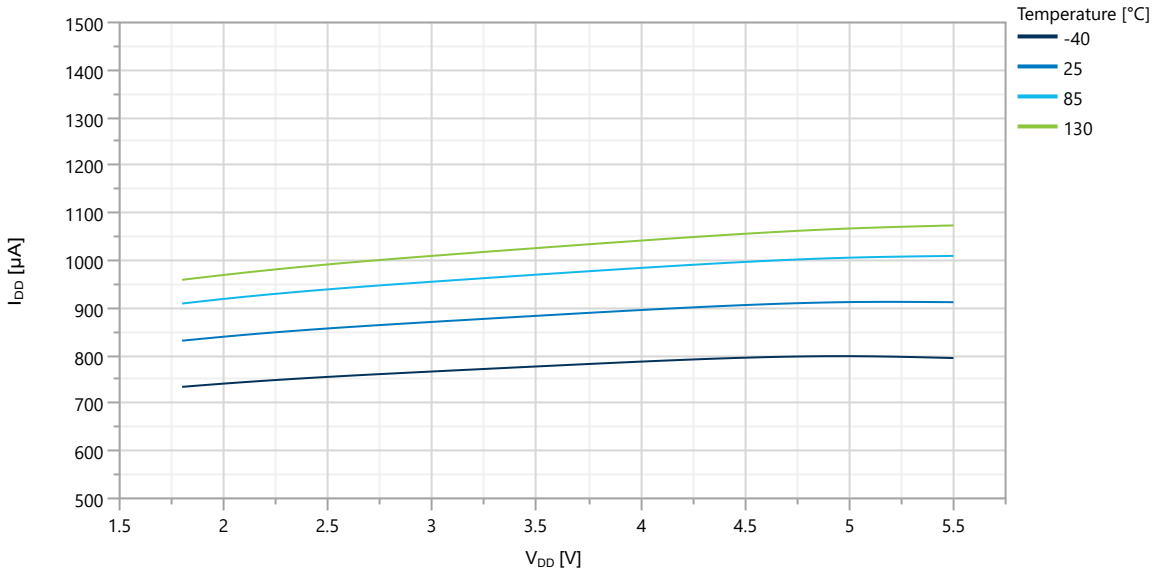



**Figure 40-18. $I_{DD}$ over $V_{DD}$ and Temperature with IRSEL = 1**



# 41. Ordering Information

- Available ordering options can be found by:
  - Clicking on one of the following product page links:
    - AVR128DB28 Product Page
    - AVR128DB32 Product Page
    - AVR128DB48 Product Page
    - AVR128DB64 Product Page
  - Searching by product name at www.microchipdirect.com
  - Contacting your local sales representative

**Table 41-1. Available Product Numbers**

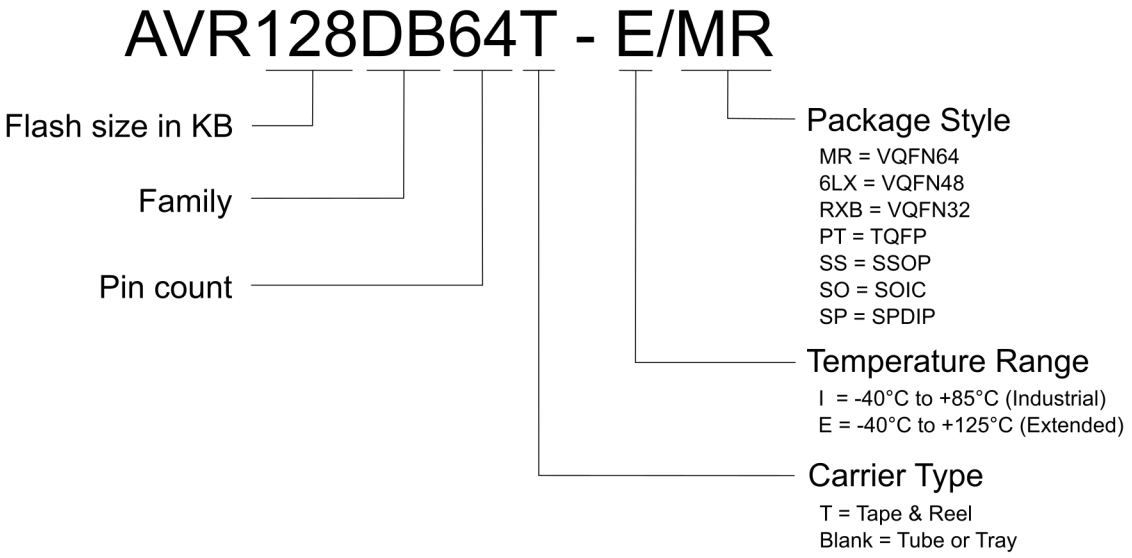
| Ordering Code | Flash/SRAM | Pin Count | Package Type | Supply Voltage | Temperature Range | Carrier Type |
|---|---|---|---|---|---|---|
| AVR128DB28-E/SP | 128 KB/16 KB | 28 | SPDIP | 1.8-5.5V | -40°C to +125°C | Tube |
| AVR128DB28-I/SP | 128 KB/16 KB | 28 | SPDIP | 1.8-5.5V | -40°C to +85°C | Tube |
| AVR128DB28-E/SO | 128 KB/16 KB | 28 | SOIC | 1.8-5.5V | -40°C to +125°C | Tube |
| AVR128DB28T-E/SO | 128 KB/16 KB | 28 | SOIC | 1.8-5.5V | -40°C to +125°C | Tape & Reel |
| AVR128DB28-I/SO | 128 KB/16 KB | 28 | SOIC | 1.8-5.5V | -40°C to +85°C | Tube |
| AVR128DB28T-I/SO | 128 KB/16 KB | 28 | SOIC | 1.8-5.5V | -40°C to +85°C | Tape & Reel |
| AVR128DB28-E/SS | 128 KB/16 KB | 28 | SSOP | 1.8-5.5V | -40°C to +125°C | Tube |
| AVR128DB28T-E/SS | 128 KB/16 KB | 28 | SSOP | 1.8-5.5V | -40°C to +125°C | Tape & Reel |
| AVR128DB28-I/SS | 128 KB/16 KB | 28 | SSOP | 1.8-5.5V | -40°C to +85°C | Tube |
| AVR128DB28T-I/SS | 128 KB/16 KB | 28 | SSOP | 1.8-5.5V | -40°C to +85°C | Tape & Reel |
| AVR128DB32-E/PT | 128 KB/16 KB | 32 | TQFP | 1.8-5.5V | -40°C to +125°C | Tray |
| AVR128DB32T-E/PT | 128 KB/16 KB | 32 | TQFP | 1.8-5.5V | -40°C to +125°C | Tape & Reel |
| AVR128DB32-I/PT | 128 KB/16 KB | 32 | TQFP | 1.8-5.5V | -40°C to +85°C | Tray |
| AVR128DB32T-I/PT | 128 KB/16 KB | 32 | TQFP | 1.8-5.5V | -40°C to +85°C | Tape & Reel |
| AVR128DB32-E/RXB | 128 KB/16 KB | 32 | VQFN | 1.8-5.5V | -40°C to +125°C | Tray |
| AVR128DB32T-E/RXB | 128 KB/16 KB | 32 | VQFN | 1.8-5.5V | -40°C to +125°C | Tape & Reel |
| AVR128DB32-I/RXB | 128 KB/16 KB | 32 | VQFN | 1.8-5.5V | -40°C to +85°C | Tray |
| AVR128DB32T-I/RXB | 128 KB/16 KB | 32 | VQFN | 1.8-5.5V | -40°C to +85°C | Tape & Reel |
| AVR128DB48-E/PT | 128 KB/16 KB | 48 | TQFP | 1.8-5.5V | -40°C to +125°C | Tray |
| AVR128DB48T-E/PT | 128 KB/16 KB | 48 | TQFP | 1.8-5.5V | -40°C to +125°C | Tape & Reel |
| AVR128DB48-I/PT | 128 KB/16 KB | 48 | TQFP | 1.8-5.5V | -40°C to +85°C | Tray |
| AVR128DB48T-I/PT | 128 KB/16 KB | 48 | TQFP | 1.8-5.5V | -40°C to +85°C | Tape & Reel |
| AVR128DB48-E/6LX | 128 KB/16 KB | 48 | VQFN | 1.8-5.5V | -40°C to +125°C | Tube |

.....continued

| Ordering Code | Flash/SRAM | Pin Count | Package Type | Supply Voltage | Temperature Range | Carrier Type |
|---|---|---|---|---|---|---|
| AVR128DB48T-E/6LX | 128 KB/16 KB | 48 | VQFN | 1.8-5.5V | -40°C to +125°C | Tape & Reel |
| AVR128DB48-I/6LX | 128 KB/16 KB | 48 | VQFN | 1.8-5.5V | -40°C to +85°C | Tube |
| AVR128DB48T-I/6LX | 128 KB/16 KB | 48 | VQFN | 1.8-5.5V | -40°C to +85°C | Tape & Reel |
| AVR128DB64-E/PT | 128 KB/16 KB | 64 | TQFP | 1.8-5.5V | -40°C to +125°C | Tray |
| AVR128DB64T-E/PT | 128 KB/16 KB | 64 | TQFP | 1.8-5.5V | -40°C to +125°C | Tape & Reel |
| AVR128DB64-I/PT | 128 KB/16 KB | 64 | TQFP | 1.8-5.5V | -40°C to +85°C | Tray |
| AVR128DB64T-I/PT | 128 KB/16 KB | 64 | TQFP | 1.8-5.5V | -40°C to +85°C | Tape & Reel |
| AVR128DB64-E/MR | 128 KB/16 KB | 64 | QFN | 1.8-5.5V | -40°C to +125°C | Tray |
| AVR128DB64T-E/MR | 128 KB/16 KB | 64 | QFN | 1.8-5.5V | -40°C to +125°C | Tape & Reel |
| AVR128DB64-I/MR | 128 KB/16 KB | 64 | QFN | 1.8-5.5V | -40°C to +85°C | Tray |
| AVR128DB64T-I/MR | 128 KB/16 KB | 64 | QFN | 1.8-5.5V | -40°C to +85°C | Tape & Reel |

**Figure 41-1. Product Identification System**

To order or obtain information, for example on pricing or delivery, refer to the factory, or the listed sales office.




**Note:** Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

# 42. Package Drawings

## 42.1 Online Package Drawings

For the most recent package drawings:

1. Go to www.microchip.com/packaging.
2. Go to the package type-specific page, for example, VQFN.
3. Search for either Drawing Number and Style to find the most recent package drawings.

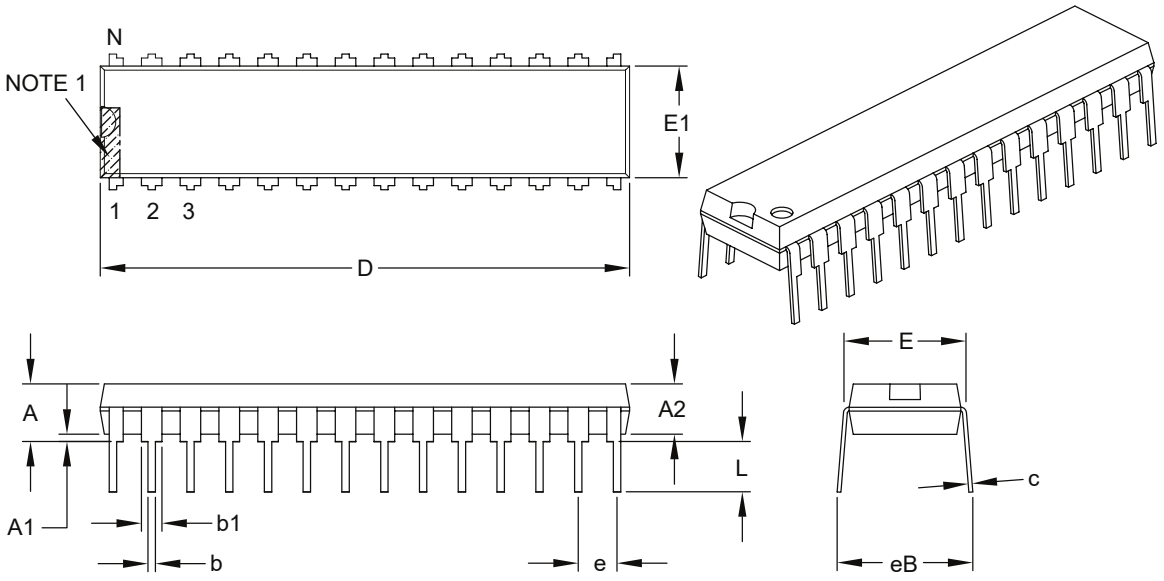
**Table 42-1. Drawing Numbers**

| Pin Count | Package Type | Drawing Number | Style |
|---|---|---|---|
| 28 | SPDIP | C04-00070 | SP |
| 28 | SOIC | C04-00052 | SO |
| 28 | SSOP | C04-00073 | SS |
| 32 | VQFN | C04-21395 | RXB |
| 32 | TQFP | C04-00074 | PT |
| 48 | VQFN | C04-00494 | 6LX |
| 48 | TQFP | C04-00300 | PT |
| 64 | VQFN | C04-00149 | MR |
| 64 | TQFP | C04-00085 | PT |

## 42.2 28-Pin SPDIP

### 28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | INCHES | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | 28 | | |
| Pitch | e | .100 BSC | | |
| Top to Seating Plane | A | – | – | .200 |
| Molded Package Thickness | A2 | .120 | .135 | .150 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .290 | .310 | .335 |
| Molded Package Width | E1 | .240 | .285 | .295 |
| Overall Length | D | 1.345 | 1.365 | 1.400 |
| Tip to Seating Plane | L | .110 | .130 | .150 |
| Lead Thickness | c | .008 | .010 | .015 |
| Upper Lead Width | b1 | .040 | .050 | .070 |
| Lower Lead Width | b | .014 | .018 | .022 |
| Overall Row Spacing § | eB | – | – | .430 |

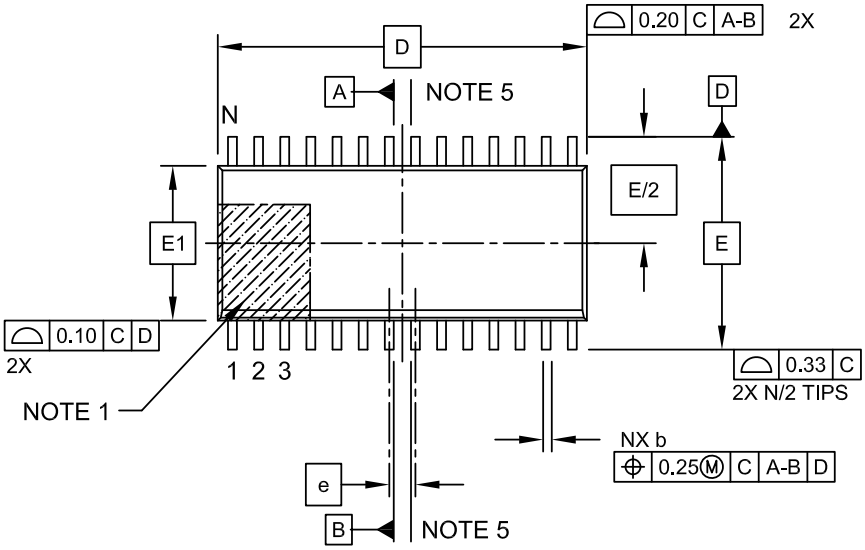
**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

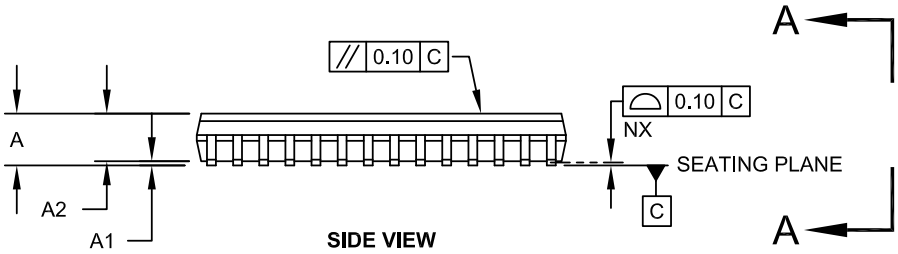
## 42.3 28-Pin SOIC

### 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

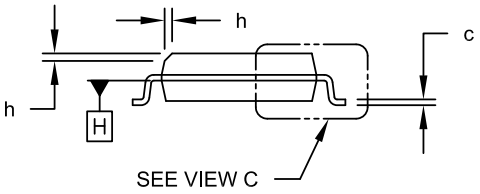
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




**TOP VIEW**




**SIDE VIEW**

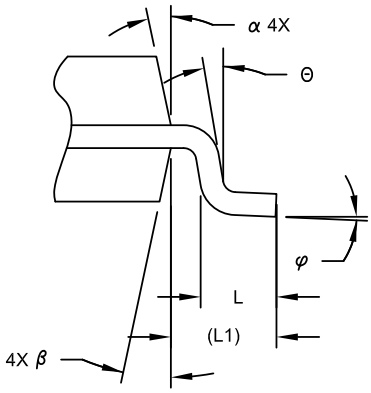



**VIEW A-A**

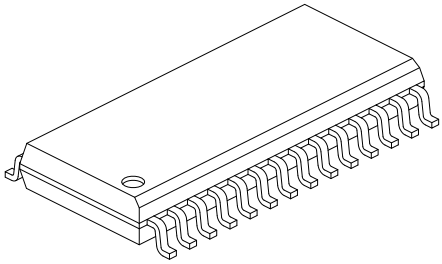
Microchip Technology Drawing C04-052C Sheet 1 of 2

## 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




VIEW C



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | 28 | | |
| Pitch | e | 1.27 BSC | | |
| Overall Height | A | - | - | 2.65 |
| Molded Package Thickness | A2 | 2.05 | - | - |
| Standoff § | A1 | 0.10 | - | 0.30 |
| Overall Width | E | 10.30 BSC | | |
| Molded Package Width | E1 | 7.50 BSC | | |
| Overall Length | D | 17.90 BSC | | |
| Chamfer (Optional) | h | 0.25 | - | 0.75 |
| Foot Length | L | 0.40 | - | 1.27 |
| Footprint | L1 | 1.40 REF | | |
| Lead Angle | Θ | 0° | - | - |
| Foot Angle | φ | 0° | - | 8° |
| Lead Thickness | c | 0.18 | - | 0.33 |
| Lead Width | b | 0.31 | - | 0.51 |
| Mold Draft Angle Top | α | 5° | - | 15° |
| Mold Draft Angle Bottom | β | 5° | - | 15° |

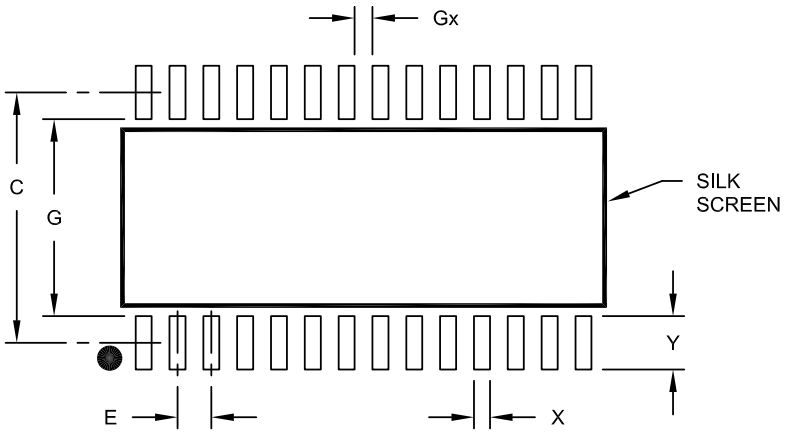
**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing C04-052C Sheet 2 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 1.27 BSC | |
| Contact Pad Spacing | C | | 9.40 | |
| Contact Pad Width (X28) | X | | | 0.60 |
| Contact Pad Length (X28) | Y | | | 2.00 |
| Distance Between Pads | Gx | 0.67 | | |
| Distance Between Pads | G | 7.40 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

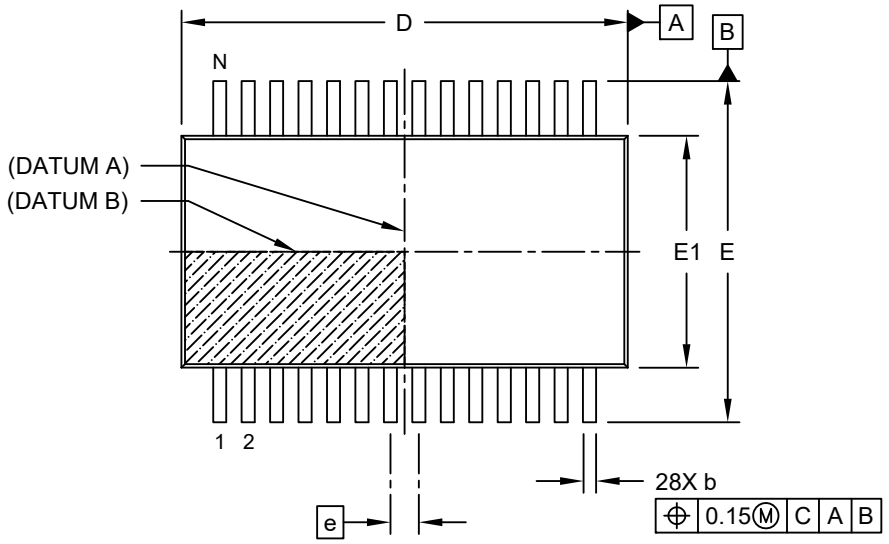
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A

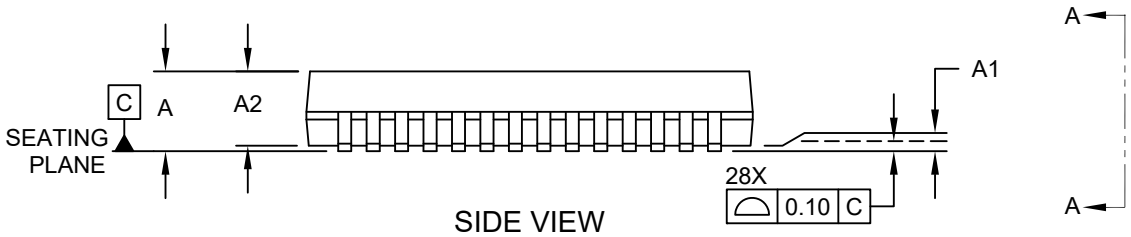
## 42.4 28-Pin SSOP

### 28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

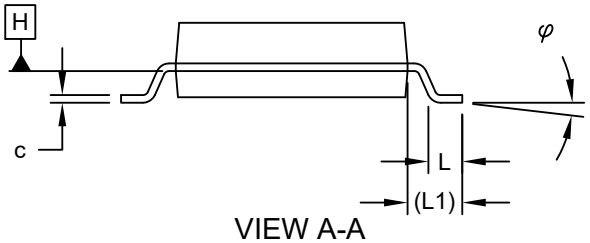
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




TOP VIEW




SIDE VIEW

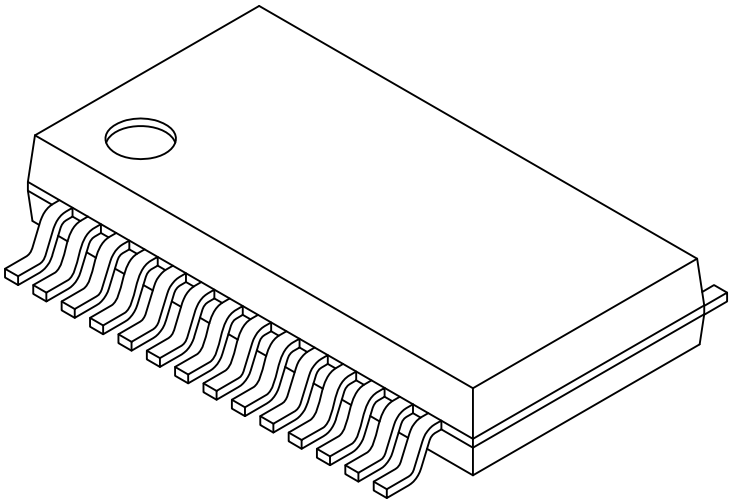



VIEW A-A

Microchip Technology Drawing C04-073 Rev C Sheet 1 of 2

## 28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | 28 | | |
| Pitch | e | 0.65 BSC | | |
| Overall Height | A | - | - | 2.00 |
| Molded Package Thickness | A2 | 1.65 | 1.75 | 1.85 |
| Standoff | A1 | 0.05 | - | - |
| Overall Width | E | 7.40 | 7.80 | 8.20 |
| Molded Package Width | E1 | 5.00 | 5.30 | 5.60 |
| Overall Length | D | 9.90 | 10.20 | 10.50 |
| Foot Length | L | 0.55 | 0.75 | 0.95 |
| Footprint | L1 | 1.25 REF | | |
| Lead Thickness | c | 0.09 | - | 0.25 |
| Foot Angle | φ | 0° | 4° | 8° |
| Lead Width | b | 0.22 | - | 0.38 |

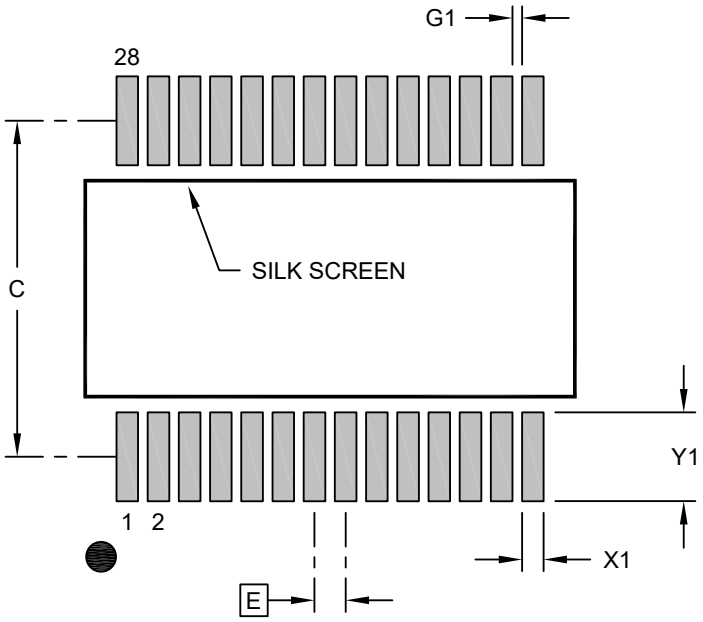
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073 Rev C Sheet 2 of 2

## 28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.65 BSC | | |
| Contact Pad Spacing | C | | 7.00 | |
| Contact Pad Width (X28) | X1 | | | 0.45 |
| Contact Pad Length (X28) | Y1 | | | 1.85 |
| Contact Pad to Center Pad (X26) | G1 | 0.20 | | |

Notes:

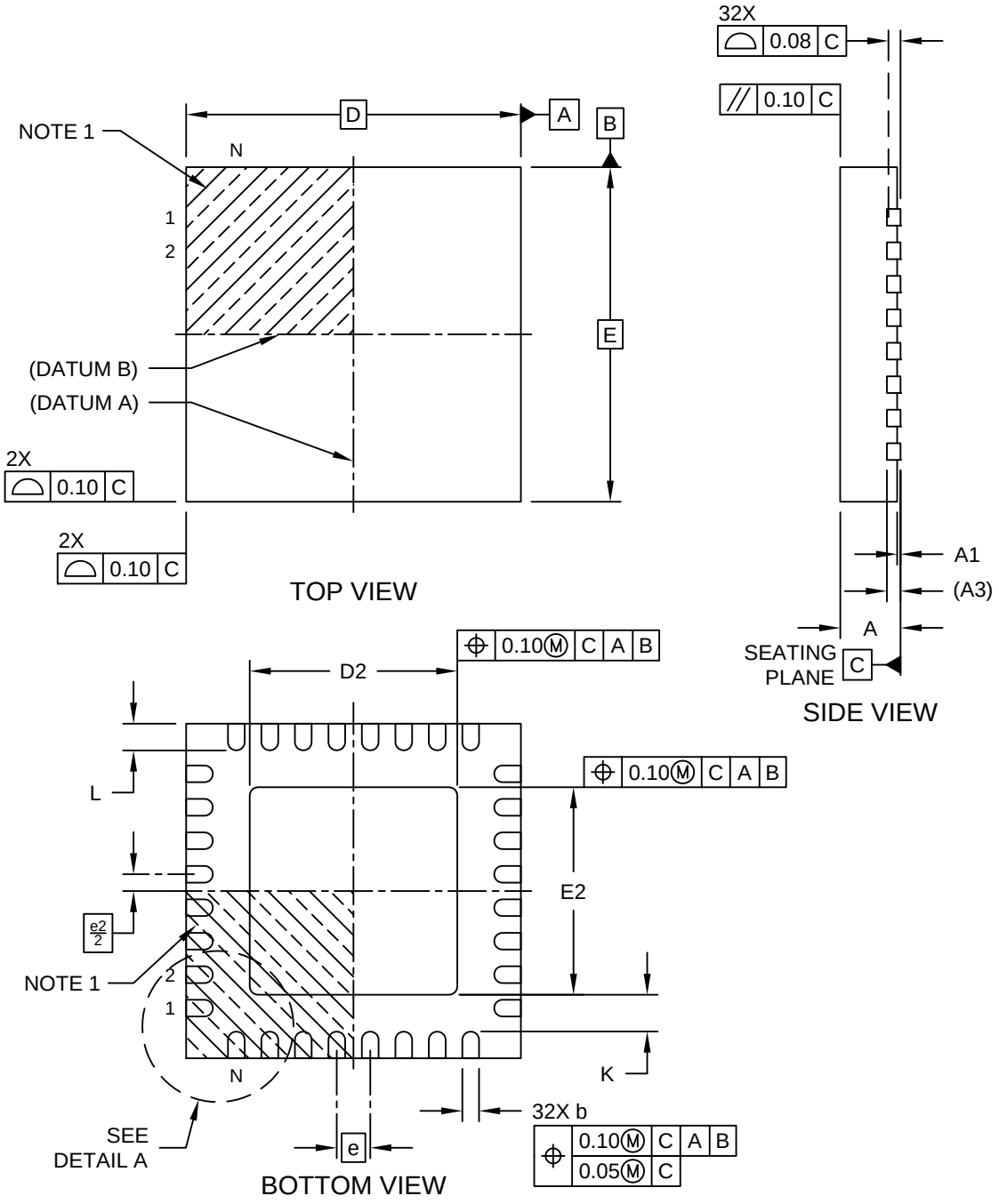
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2073 Rev B

## 42.5 32-Pin VQFN

**32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF**

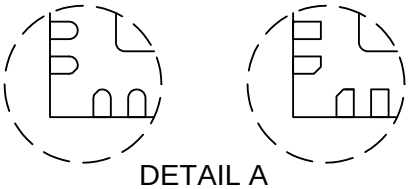
| | |
|---|---|
| **Note:** | For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging |



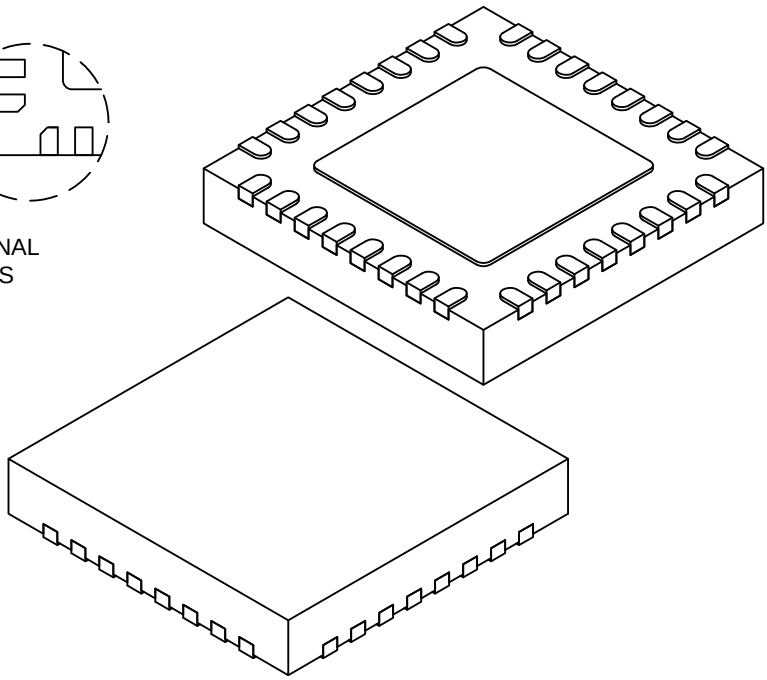

Microchip Technology Drawing C04-21395-RXB Rev B Sheet 1 of 2

### 32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



DETAIL A
ALTERNATE TERMINAL CONFIGURATIONS



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | 32 | | |
| Pitch | e | 0.50 BSC | | |
| Overall Height | A | 0.80 | 0.85 | 0.90 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Terminal Thickness | A3 | 0.203 REF | | |
| Overall Length | D | 5.00 BSC | | |
| Exposed Pad Length | D2 | 3.00 | 3.10 | 3.20 |
| Overall Width | E | 5.00 BSC | | |
| Exposed Pad Width | E2 | 3.00 | 3.10 | 3.20 |
| Terminal Width | b | 0.18 | 0.25 | 0.30 |
| Terminal Length | L | 0.30 | 0.40 | 0.50 |
| Terminal-to-Exposed-Pad | K | 0.20 | - | - |

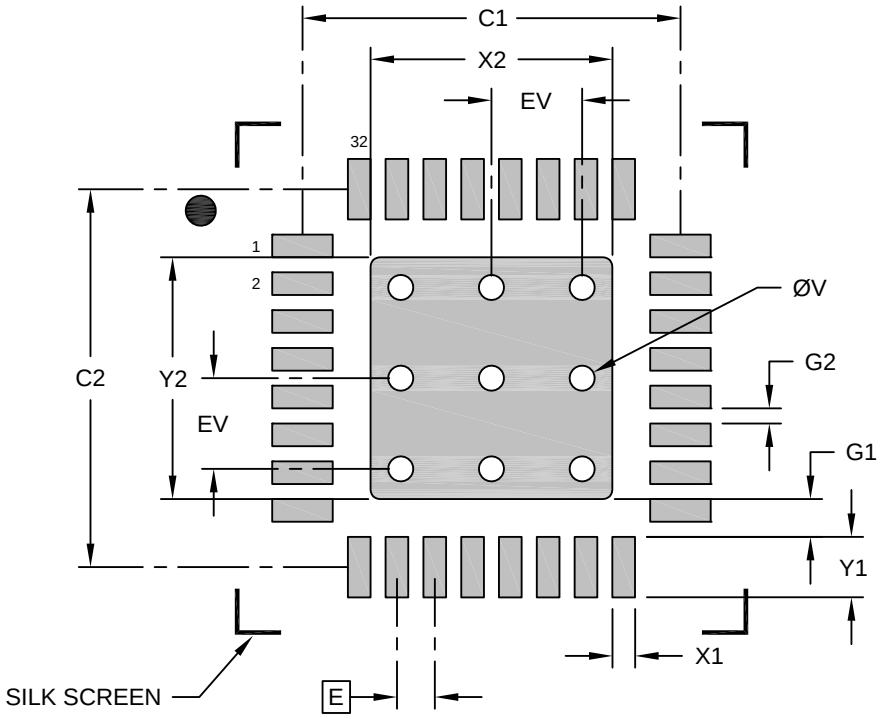
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21395-RXB Rev B Sheet 2 of 2

### 32-Lead Very Thin Plastic Quad Flat, No Lead Package (RXB) - 5x5x0.9 mm Body [VQFN] With 3.1x3.1 mm Exposed Pad; Atmel Legacy Global Package Code ZMF

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.50 BSC | | |
| Center Pad Width | X2 | | | 3.20 |
| Center Pad Length | Y2 | | | 3.20 |
| Contact Pad Spacing | C1 | | 5.00 | |
| Contact Pad Spacing | C2 | | 5.00 | |
| Contact Pad Width (X32) | X1 | | | 0.30 |
| Contact Pad Length (X32) | Y1 | | | 0.80 |
| Contact Pad to Center Pad (X32) | G1 | 0.20 | | |
| Contact Pad to Contactr Pad (X28) | G2 | 0.20 | | |
| Thermal Via Diameter | V | | 0.33 | |
| Thermal Via Pitch | EV | | 1.20 | |

Notes:

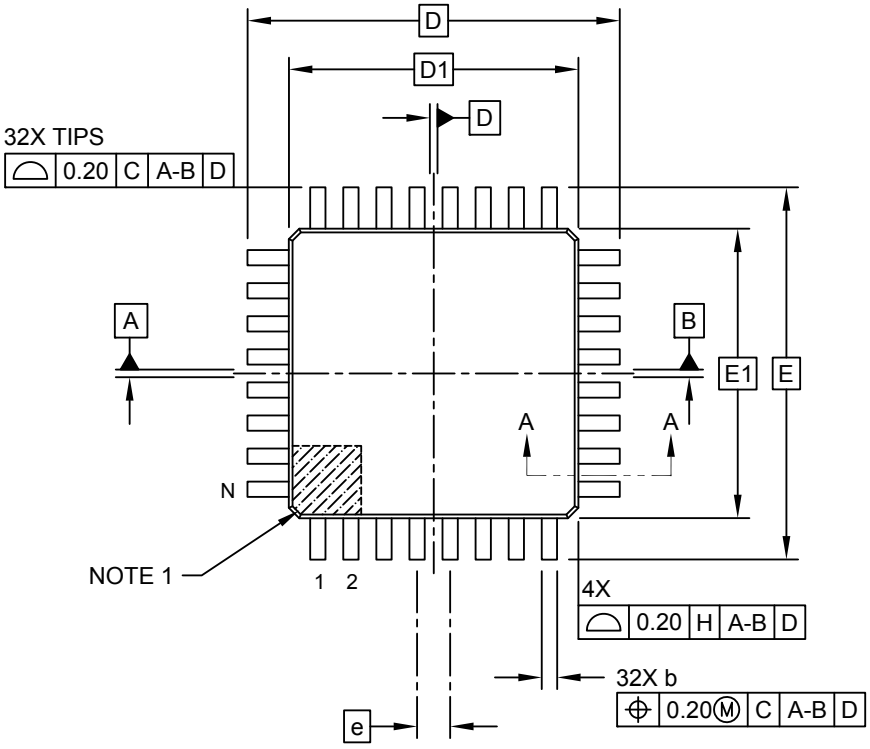
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23395-RXB Rev B

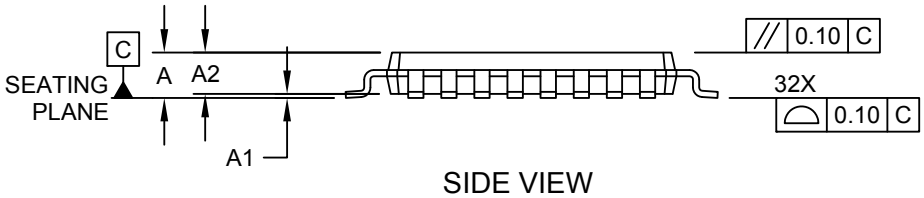
## 42.6 32-Pin TQFP

### 32-Lead Plastic Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP] 2.00 mm Footprint; Also Atmel Legacy Global Package Code AUT

| | |
|---|---|
| **Note:** | For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging |




TOP VIEW

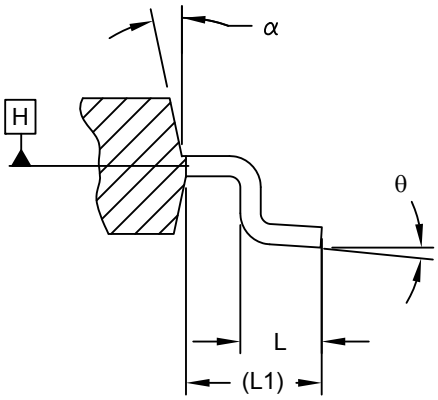



SIDE VIEW

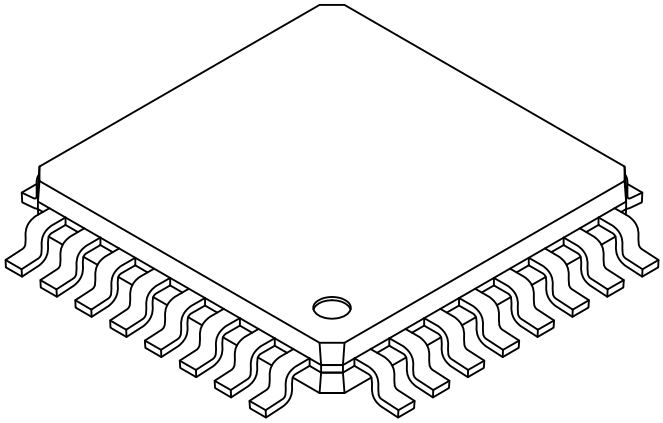
Microchip Technology Drawing C04-074 Rev C Sheet 1 of 2

## 32-Lead Plastic Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP] 2.00 mm Footprint; Also Atmel Legacy Global Package Code AUT

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




SECTION A-A



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Leads | N | 32 | | |
| Lead Pitch | e | 0.80 BSC | | |
| Overall Height | A | - | - | 1.20 |
| Standoff | A1 | 0.05 | - | 0.15 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Foot Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | 1.00 REF | | |
| Foot Angle | θ | 0° | - | 7° |
| Overall Width | E | 9.00 BSC | | |
| Overall Length | D | 9.00 BSC | | |
| Molded Package Width | E1 | 7.00 BSC | | |
| Molded Package Length | D1 | 7.00 BSC | | |
| Lead Width | b | 0.30 | 0.37 | 0.45 |
| Mold Draft Angle Top | α | 11° | - | 13° |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensioning and tolerancing per ASME Y14.5M

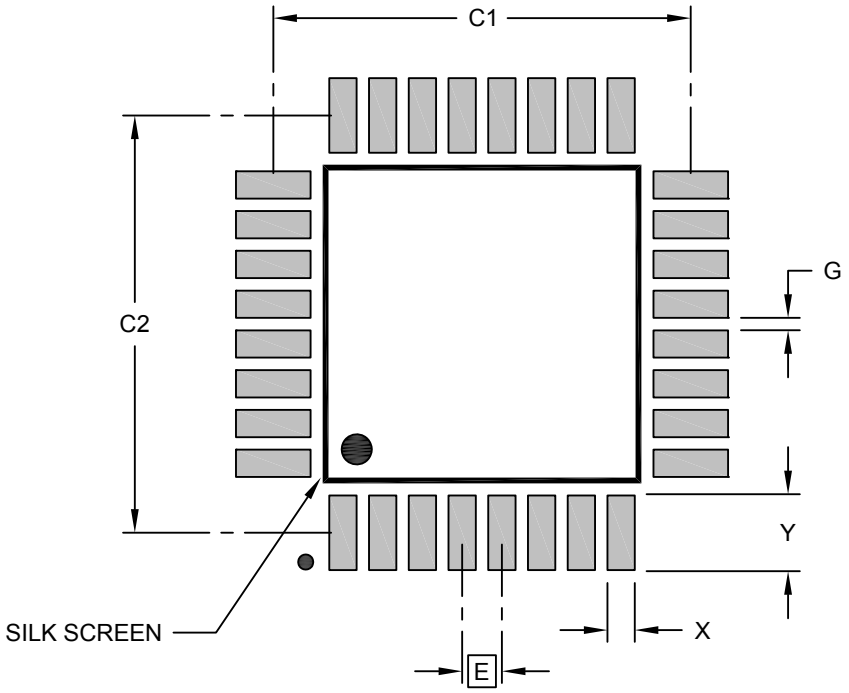
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-074 Rev C Sheet 2 of 2

## 32-Lead Thin Plastic Quad Flatpack (PT) - 7x7 mm Body [TQFP] 2.00 mm Footprint; Also Atmel Legacy Global Package Code AUT

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.80 BSC | |
| Contact Pad Spacing | C1 | | 8.40 | |
| Contact Pad Spacing | C2 | | 8.40 | |
| Contact Pad Width (Xnn) | X | | | 0.55 |
| Contact Pad Length (Xnn) | Y | | | 1.55 |
| Contact Pad to Contact Pad (Xnn) | G | 0.25 | | |

Notes:

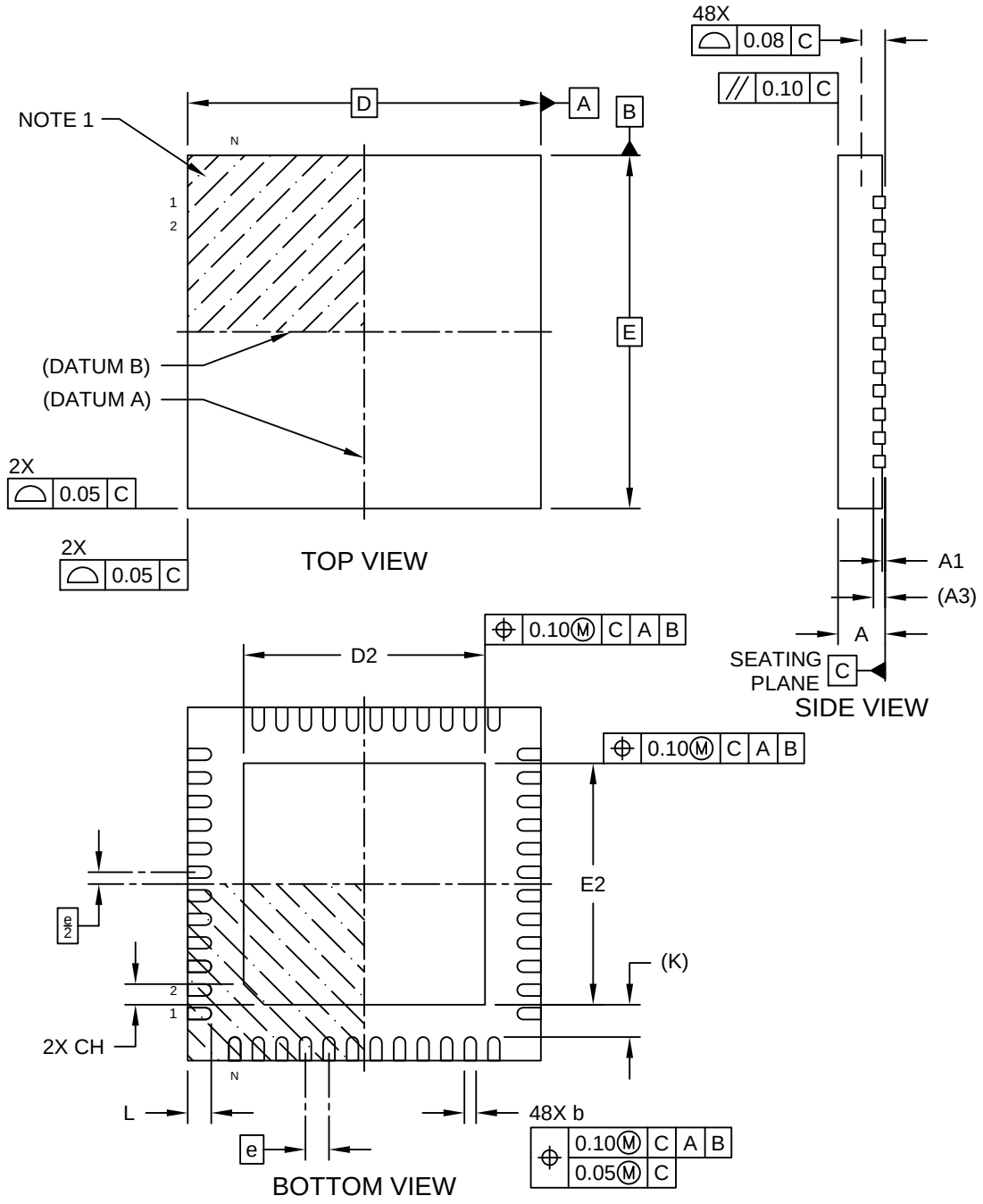
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-2074 Rev C

## 42.7 48-Pin VQFN

### 48-Lead Very Thin Plastic Quad Flat, No Lead Package (6LX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad

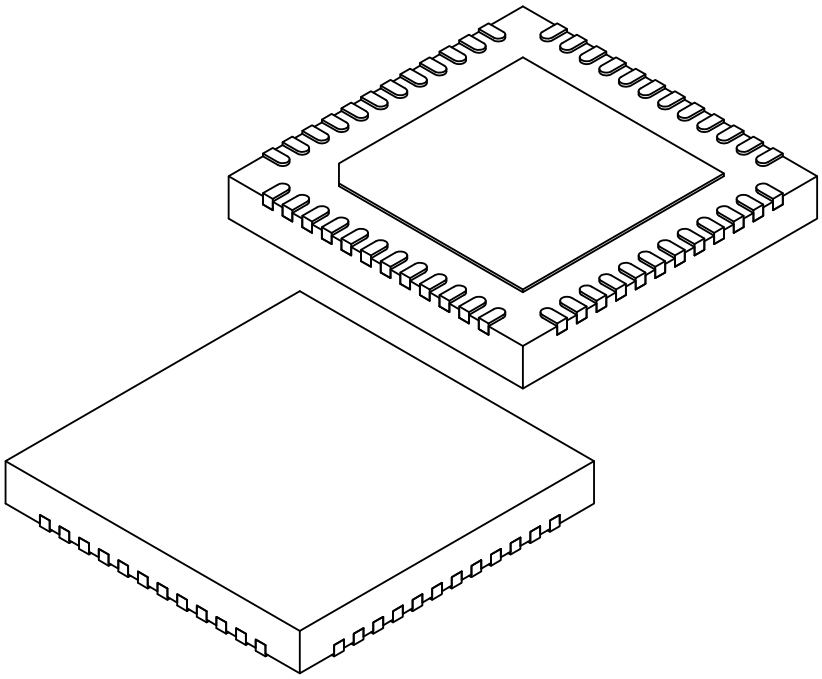
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing C04-494 Rev A Sheet 1 of 2

## 48-Lead Very Thin Plastic Quad Flat, No Lead Package (6LX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | 48 | | |
| Pitch | e | 0.40 BSC | | |
| Overall Height | A | 0.80 | 0.85 | 0.90 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Terminal Thickness | A3 | 0.20 REF | | |
| Overall Length | D | 6.00 BSC | | |
| Exposed Pad Length | D2 | 4.00 | 4.10 | 4.20 |
| Overall Width | E | 6.00 BSC | | |
| Exposed Pad Width | E2 | 4.00 | 4.10 | 4.20 |
| Exposed Pad Corner Chamfer | CH | 0.35 REF | | |
| Terminal Width | b | 0.15 | 0.20 | 0.25 |
| Terminal Length | L | 0.30 | 0.40 | 0.50 |
| Terminal-to-Exposed-Pad | K | 0.55 REF | | |

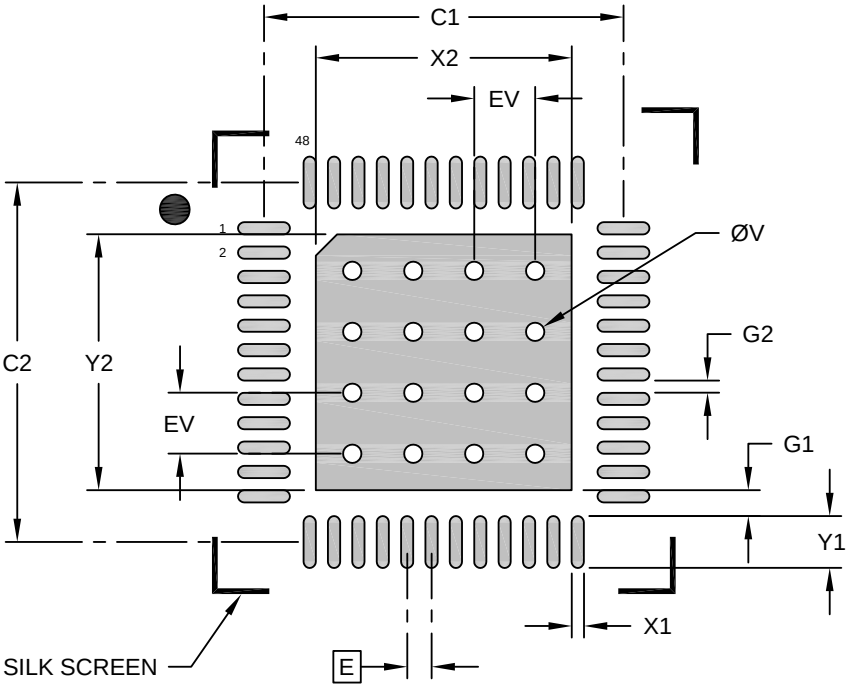
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-494 Rev A Sheet 1 of 2

## 48-Lead Very Thin Plastic Quad Flat, No Lead (6LX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.40 BSC | |
| Optional Center Pad Width | X2 | | | 4.20 |
| Optional Center Pad Length | Y2 | | | 4.20 |
| Contact Pad Spacing | C1 | | 5.90 | |
| Contact Pad Spacing | C2 | | 5.90 | |
| Contact Pad Width (X48) | X1 | | | 0.20 |
| Contact Pad Length (X48) | Y1 | | | 0.85 |
| Contact Pad to Center Pad (X48) | G1 | 0.20 | | |
| Contact Pad to Contact Pad (X44) | G2 | 0.20 | | |
| Thermal Via Diameter | V | | 0.30 | |
| Thermal Via Pitch | EV | | 1.00 | |

Notes:

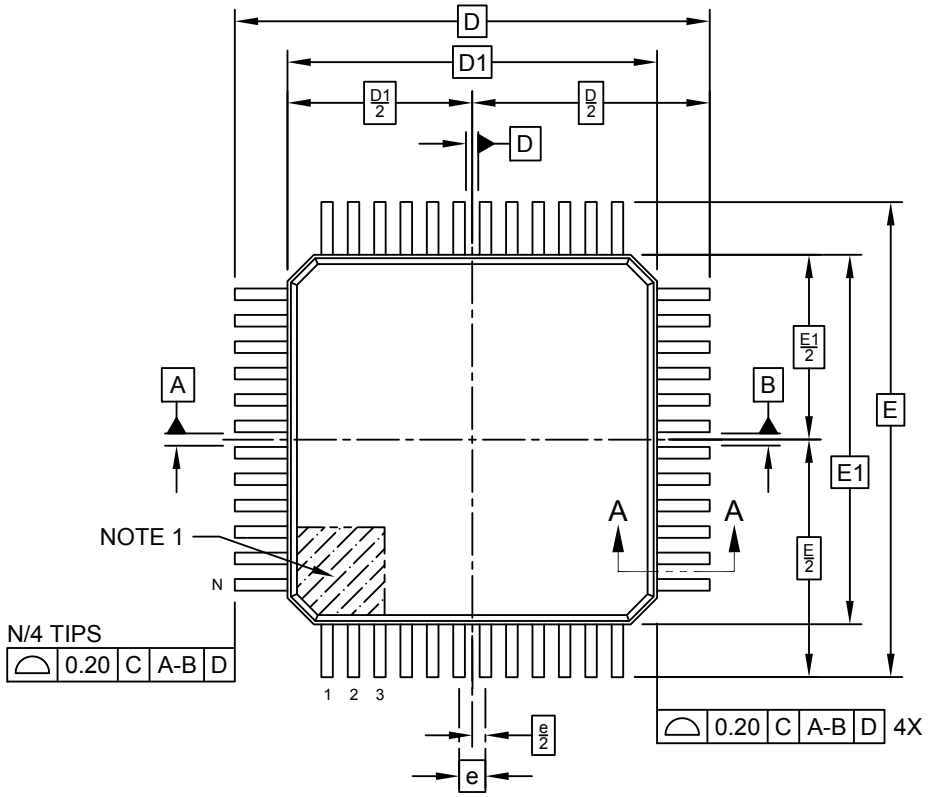
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2494 Rev A

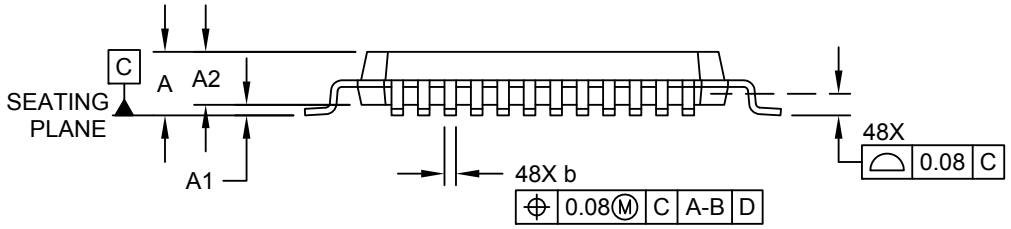
## 42.8 48-Pin TQFP

### 48-Lead Plastic Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




TOP VIEW

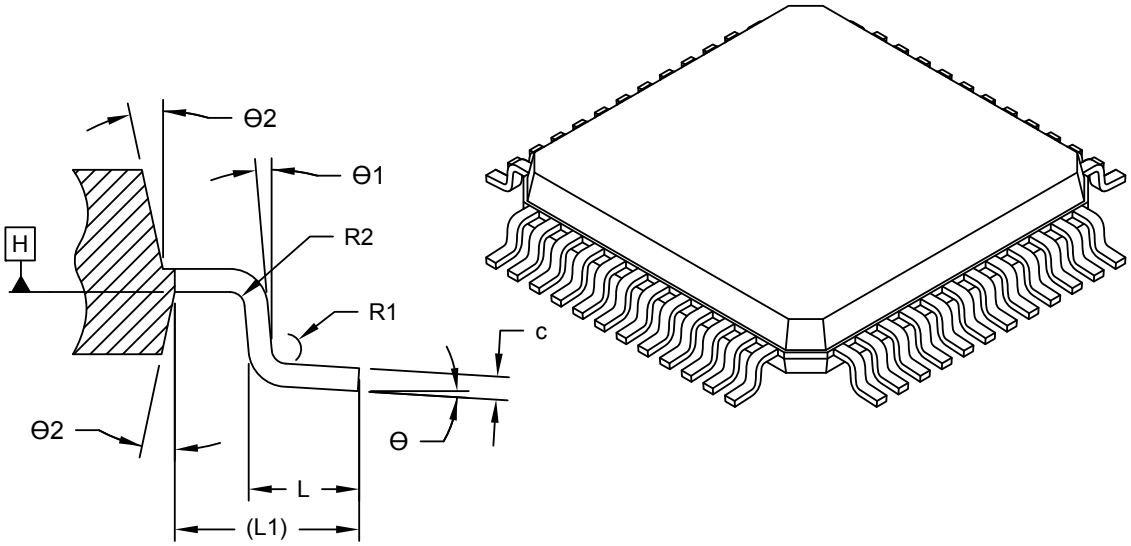



SIDE VIEW

Microchip Technology Drawing C04-300-PT Rev D Sheet 1 of 2

## 48-Lead Plastic Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




SECTION A-A

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | 48 | | |
| Pitch | e | 0.50 BSC | | |
| Overall Height | A | - | - | 1.20 |
| Standoff | A1 | 0.05 | - | 0.15 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Overall Length | D | 9.00 BSC | | |
| Molded Package Length | D1 | 7.00 BSC | | |
| Overall Width | E | 9.00 BSC | | |
| Molded Package Width | E1 | 7.00 BSC | | |
| Terminal Width | b | 0.17 | 0.22 | 0.27 |
| Terminal Thickness | c | 0.09 | - | 0.16 |
| Terminal Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | 1.00 REF | | |
| Lead Bend Radius | R1 | 0.08 | - | - |
| Lead Bend Radius | R2 | 0.08 | - | 0.20 |
| Foot Angle | Θ | 0° | 3.5° | 7° |
| Lead Angle | Θ1 | 0° | - | - |
| Mold Draft Angle | Θ2 | 11° | 12° | 13° |

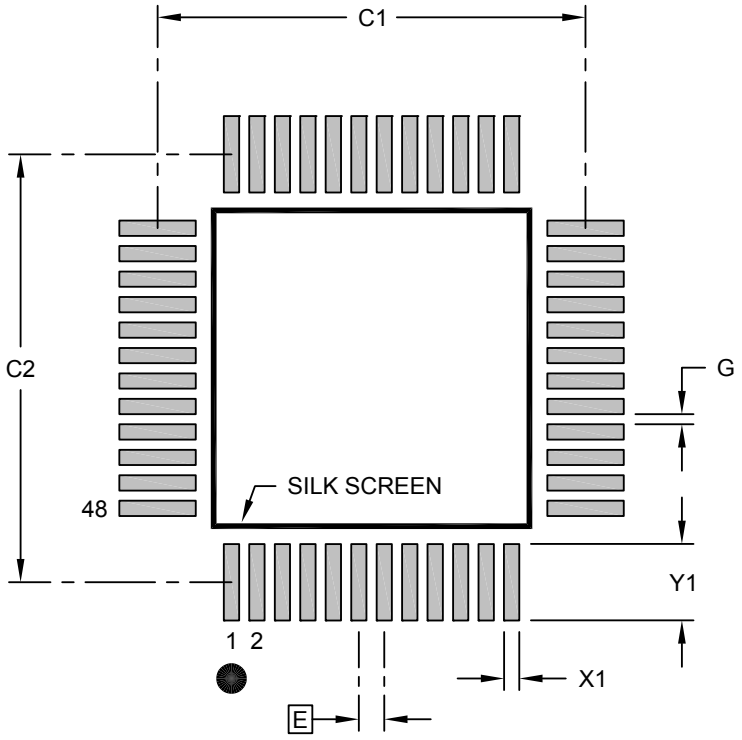
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-300-PT Rev D Sheet 2 of 2

## 48-Lead Plastic Thin Quad Flatpack (PT) - 7x7x1.0 mm Body [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.50 BSC | |
| Contact Pad Spacing | C1 | | 8.40 | |
| Contact Pad Spacing | C2 | | 8.40 | |
| Contact Pad Width (X48) | X1 | | | 0.30 |
| Contact Pad Length (X48) | Y1 | | | 1.50 |
| Distance Between Pads | G | 0.20 | | |

Notes:

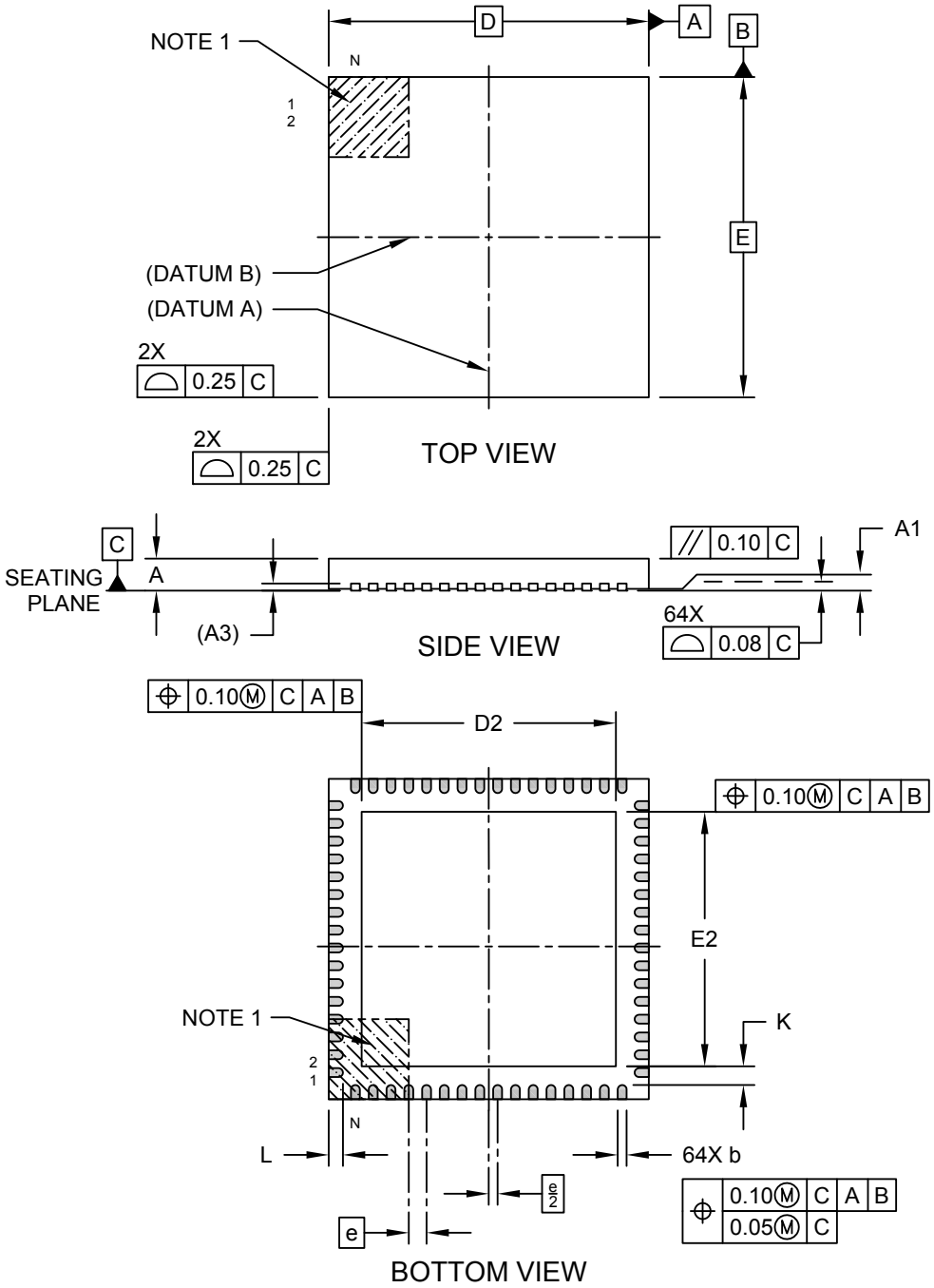
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2300-PT Rev D

## 42.9 64-Pin VQFN

**64-Lead Very Thin Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [VQFN] With 7.15 x 7.15 Exposed Pad [Also called QFN]**

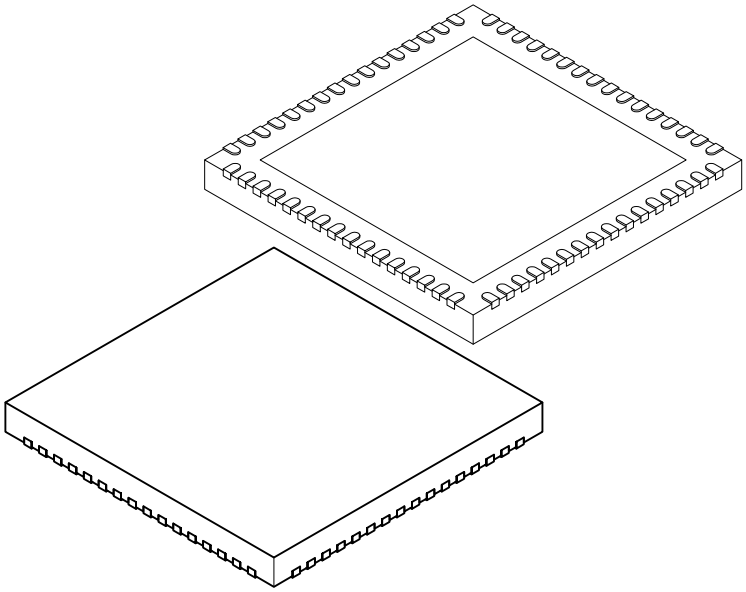
| | |
|---|---|
| **Note:** | For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging |




Microchip Technology Drawing C04-149 [MR] Rev E Sheet 1 of 2

**64-Lead Very Thin Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [VQFN]**
**With 7.15 x 7.15 Exposed Pad [Also called QFN]**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | 64 | | |
| Pitch | e | 0.50 BSC | | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | 0.20 REF | | |
| Overall Width | E | 9.00 BSC | | |
| Exposed Pad Width | E2 | 7.05 | 7.15 | 7.25 |
| Overall Length | D | 9.00 BSC | | |
| Exposed Pad Length | D2 | 7.05 | 7.15 | 7.25 |
| Contact Width | b | 0.18 | 0.25 | 0.30 |
| Contact Length | L | 0.30 | 0.40 | 0.50 |
| Contact-to-Exposed Pad | K | 0.20 | - | - |

Notes:

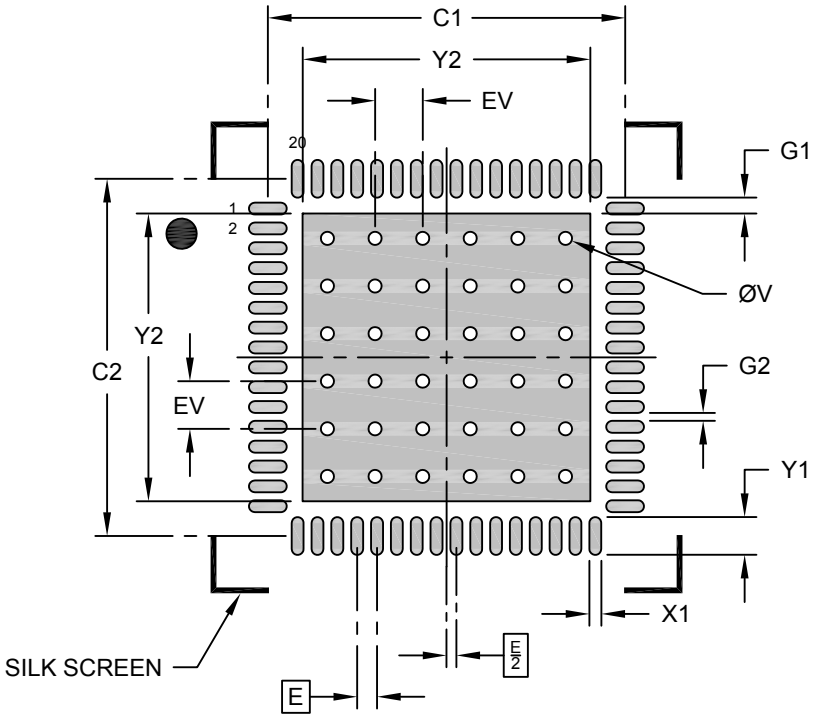
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-149 [MR] Rev E Sheet 2 of 2

**64-Lead Very Thin Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [VQFN] With 7.15 x 7.15 Exposed Pad [Also called QFN]**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.50 BSC | |
| Optional Center Pad Width | X2 | | | 7.25 |
| Optional Center Pad Length | Y2 | | | 7.25 |
| Contact Pad Spacing | C1 | | 9.00 | |
| Contact Pad Spacing | C2 | | 9.00 | |
| Contact Pad Width (X64) | X1 | | | 0.30 |
| Contact Pad Length (X64) | Y1 | | | 0.95 |
| Contact Pad to Center Pad (X64) | G1 | 0.40 | | |
| Spacing Between Contact Pads (X60) | G2 | 0.20 | | |
| Thermal Via Diameter | V | | 0.33 | |
| Thermal Via Pitch | EV | | 1.20 | |

Notes:

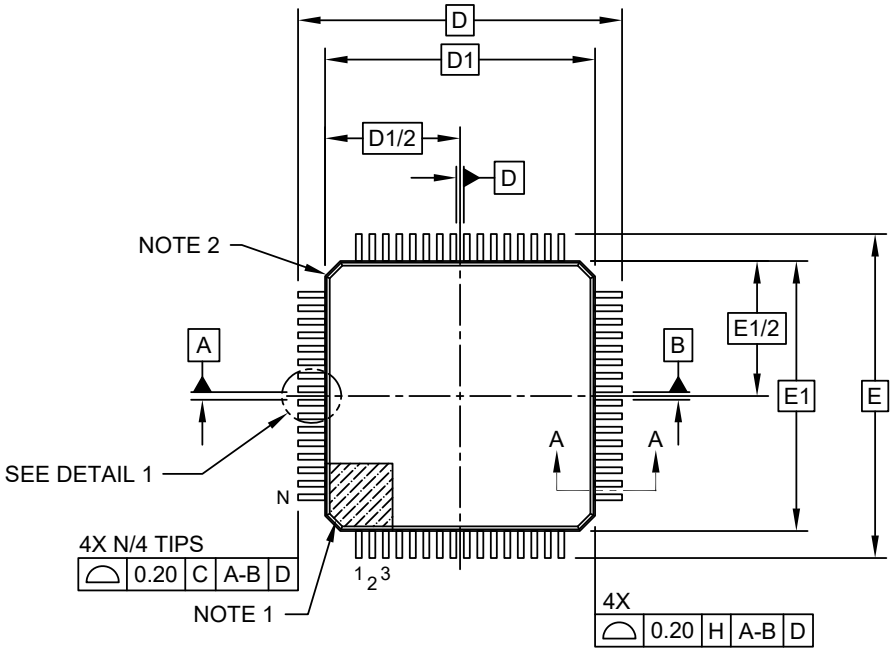
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-149 [MR] Rev E

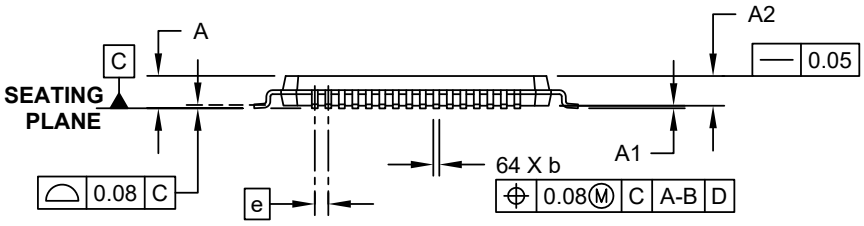
## 42.10 64-Pin TQFP

### 64-Lead Plastic Thin Quad Flatpack (PT)-10x10x1 mm Body, 2.00 mm Footprint [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




TOP VIEW

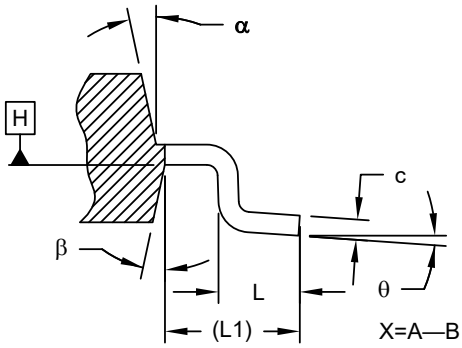



SIDE VIEW

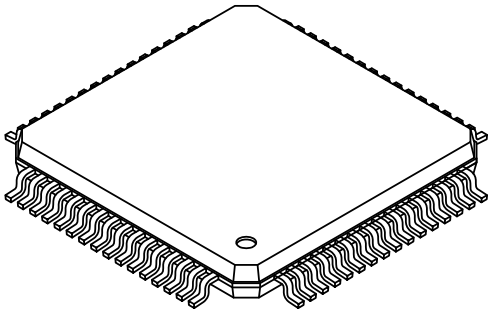
Microchip Technology Drawing C04-085C Sheet 1 of 2

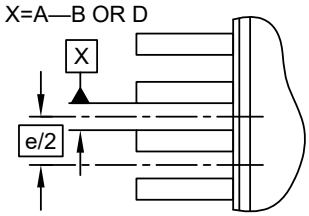
### 64-Lead Plastic Thin Quad Flatpack (PT)-10x10x1 mm Body, 2.00 mm Footprint [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




SECTION A-A






DETAIL 1

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Leads | N | 64 | | |
| Lead Pitch | e | 0.50 BSC | | |
| Overall Height | A | - | - | 1.20 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Standoff | A1 | 0.05 | - | 0.15 |
| Foot Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | 1.00 REF | | |
| Foot Angle | ϕ | 0° | 3.5° | 7° |
| Overall Width | E | 12.00 BSC | | |
| Overall Length | D | 12.00 BSC | | |
| Molded Package Width | E1 | 10.00 BSC | | |
| Molded Package Length | D1 | 10.00 BSC | | |
| Lead Thickness | c | 0.09 | - | 0.20 |
| Lead Width | b | 0.17 | 0.22 | 0.27 |
| Mold Draft Angle Top | α | 11° | 12° | 13° |
| Mold Draft Angle Bottom | β | 11° | 12° | 13° |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

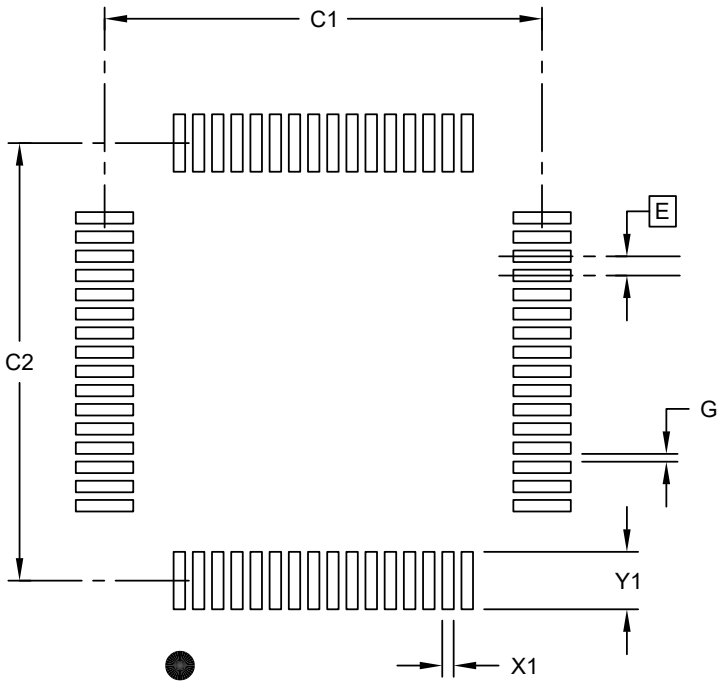
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085C Sheet 2 of 2

### 64-Lead Plastic Thin Quad Flatpack (PT)-10x10x1 mm Body, 2.00 mm Footprint [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.50 BSC | | |
| Contact Pad Spacing | C1 | | 11.40 | |
| Contact Pad Spacing | C2 | | 11.40 | |
| Contact Pad Width (X28) | X1 | | | 0.30 |
| Contact Pad Length (X28) | Y1 | | | 1.50 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-2085B Sheet 1 of 1

# 43. Data Sheet Revision History

**Note:** The data sheet revision is independent of the die revision and the device variant (last letter of the ordering number).

## 43.1 Rev. A - 08/2020

| Section | Description |
|---|---|
| Document | Initial release |

## The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

## Customer Support

Users of Microchip products can receive assistance through several channels:

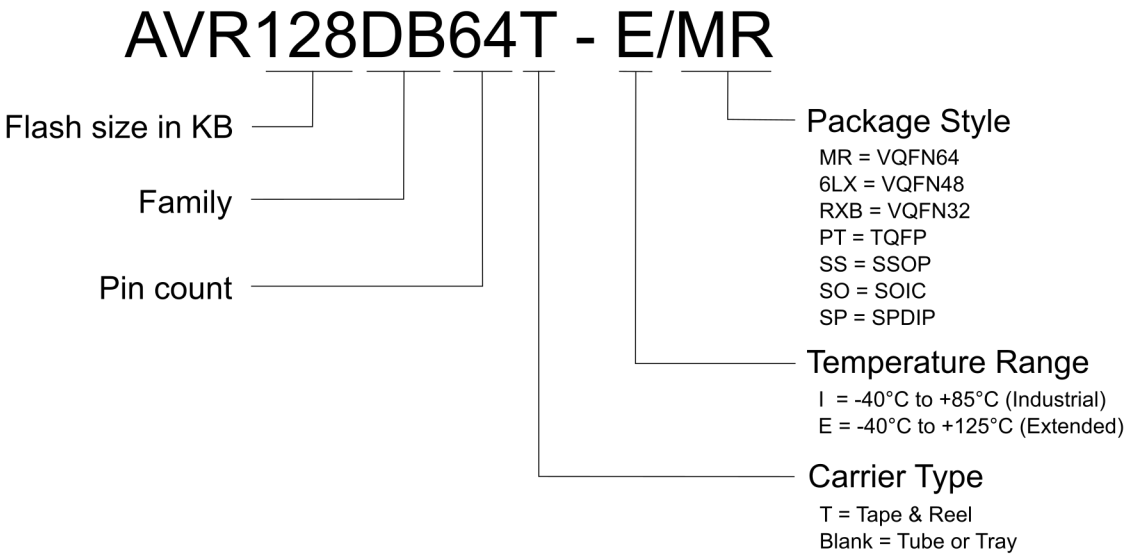
- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

## Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.




**Note:** Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

## Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

## Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2020, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-6597-3

## Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

# Worldwide Sales and Service

| AMERICAS | ASIA/PACIFIC | ASIA/PACIFIC | EUROPE |
|---|---|---|---|
| **Corporate Office**<br>2355 West Chandler Blvd.<br>Chandler, AZ 85224-6199<br>Tel: 480-792-7200<br>Fax: 480-792-7277<br>Technical Support:<br>www.microchip.com/support<br>Web Address:<br>www.microchip.com<br>**Atlanta**<br>Duluth, GA<br>Tel: 678-957-9614<br>Fax: 678-957-1455<br>**Austin, TX**<br>Tel: 512-257-3370<br>**Boston**<br>Westborough, MA<br>Tel: 774-760-0087<br>Fax: 774-760-0088<br>**Chicago**<br>Itasca, IL<br>Tel: 630-285-0071<br>Fax: 630-285-0075<br>**Dallas**<br>Addison, TX<br>Tel: 972-818-7423<br>Fax: 972-818-2924<br>**Detroit**<br>Novi, MI<br>Tel: 248-848-4000<br>**Houston, TX**<br>Tel: 281-894-5983<br>**Indianapolis**<br>Noblesville, IN<br>Tel: 317-773-8323<br>Fax: 317-773-5453<br>Tel: 317-536-2380<br>**Los Angeles**<br>Mission Viejo, CA<br>Tel: 949-462-9523<br>Fax: 949-462-9608<br>Tel: 951-273-7800<br>**Raleigh, NC**<br>Tel: 919-844-7510<br>**New York, NY**<br>Tel: 631-435-6000<br>**San Jose, CA**<br>Tel: 408-735-9110<br>Tel: 408-436-4270<br>**Canada - Toronto**<br>Tel: 905-695-1980<br>Fax: 905-695-2078 | **Australia - Sydney**<br>Tel: 61-2-9868-6733<br>**China - Beijing**<br>Tel: 86-10-8569-7000<br>**China - Chengdu**<br>Tel: 86-28-8665-5511<br>**China - Chongqing**<br>Tel: 86-23-8980-9588<br>**China - Dongguan**<br>Tel: 86-769-8702-9880<br>**China - Guangzhou**<br>Tel: 86-20-8755-8029<br>**China - Hangzhou**<br>Tel: 86-571-8792-8115<br>**China - Hong Kong SAR**<br>Tel: 852-2943-5100<br>**China - Nanjing**<br>Tel: 86-25-8473-2460<br>**China - Qingdao**<br>Tel: 86-532-8502-7355<br>**China - Shanghai**<br>Tel: 86-21-3326-8000<br>**China - Shenyang**<br>Tel: 86-24-2334-2829<br>**China - Shenzhen**<br>Tel: 86-755-8864-2200<br>**China - Suzhou**<br>Tel: 86-186-6233-1526<br>**China - Wuhan**<br>Tel: 86-27-5980-5300<br>**China - Xian**<br>Tel: 86-29-8833-7252<br>**China - Xiamen**<br>Tel: 86-592-2388138<br>**China - Zhuhai**<br>Tel: 86-756-3210040 | **India - Bangalore**<br>Tel: 91-80-3090-4444<br>**India - New Delhi**<br>Tel: 91-11-4160-8631<br>**India - Pune**<br>Tel: 91-20-4121-0141<br>**Japan - Osaka**<br>Tel: 81-6-6152-7160<br>**Japan - Tokyo**<br>Tel: 81-3-6880- 3770<br>**Korea - Daegu**<br>Tel: 82-53-744-4301<br>**Korea - Seoul**<br>Tel: 82-2-554-7200<br>**Malaysia - Kuala Lumpur**<br>Tel: 60-3-7651-7906<br>**Malaysia - Penang**<br>Tel: 60-4-227-8870<br>**Philippines - Manila**<br>Tel: 63-2-634-9065<br>**Singapore**<br>Tel: 65-6334-8870<br>**Taiwan - Hsin Chu**<br>Tel: 886-3-577-8366<br>**Taiwan - Kaohsiung**<br>Tel: 886-7-213-7830<br>**Taiwan - Taipei**<br>Tel: 886-2-2508-8600<br>**Thailand - Bangkok**<br>Tel: 66-2-694-1351<br>**Vietnam - Ho Chi Minh**<br>Tel: 84-28-5448-2100 | **Austria - Wels**<br>Tel: 43-7242-2244-39<br>Fax: 43-7242-2244-393<br>**Denmark - Copenhagen**<br>Tel: 45-4485-5910<br>Fax: 45-4485-2829<br>**Finland - Espoo**<br>Tel: 358-9-4520-820<br>**France - Paris**<br>Tel: 33-1-69-53-63-20<br>Fax: 33-1-69-30-90-79<br>**Germany - Garching**<br>Tel: 49-8931-9700<br>**Germany - Haan**<br>Tel: 49-2129-3766400<br>**Germany - Heilbronn**<br>Tel: 49-7131-72400<br>**Germany - Karlsruhe**<br>Tel: 49-721-625370<br>**Germany - Munich**<br>Tel: 49-89-627-144-0<br>Fax: 49-89-627-144-44<br>**Germany - Rosenheim**<br>Tel: 49-8031-354-560<br>**Israel - Ra'anana**<br>Tel: 972-9-744-7705<br>**Italy - Milan**<br>Tel: 39-0331-742611<br>Fax: 39-0331-466781<br>**Italy - Padova**<br>Tel: 39-049-7625286<br>**Netherlands - Drunen**<br>Tel: 31-416-690399<br>Fax: 31-416-690340<br>**Norway - Trondheim**<br>Tel: 47-72884388<br>**Poland - Warsaw**<br>Tel: 48-22-3325737<br>**Romania - Bucharest**<br>Tel: 40-21-407-87-50<br>**Spain - Madrid**<br>Tel: 34-91-708-08-90<br>Fax: 34-91-708-08-91<br>**Sweden - Gothenberg**<br>Tel: 46-31-704-60-40<br>**Sweden - Stockholm**<br>Tel: 46-8-5090-4654<br>**UK - Wokingham**<br>Tel: 44-118-921-5800<br>Fax: 44-118-921-5820 |